Requirements Engineering
https://doi.org/10.1007/s00766-023-00411-0

ORIGINAL ARTICLE

Advances in automated support for requirements engineering: a systematic literature review

Muhammad Aminu Umar[1,2] · **Kevin Lano**[1]

Received: 28 July 2023 / Accepted: 25 November 2023

Abstract
Requirements Engineering (RE) has undergone several transitions over the years, from traditional methods to agile approaches emphasising increased automation. In many software development projects, requirements are expressed in natural language and embedded within large volumes of text documents. At the same time, RE activities aim to define software systems' functionalities and constraints. However, manually executing these tasks is time-consuming and prone to errors. Numerous research efforts have proposed tools and technologies for automating RE activities to address this challenge, which are documented in published works. This review aims to examine empirical evidence on automated RE and analyse its impact on the RE sub-domain and software development. To achieve our goal, we conducted a Systematic Literature Review (SLR) following established guidelines for conducting SLRs. We aimed to identify, aggregate, and analyse papers on automated RE published between 1996 and 2022. We outlined the output of the support tool, the RE phase covered, levels of automation, development approach, and evaluation approaches. We identified 85 papers that discussed automated RE from various perspectives and methodologies. The results of this review demonstrate the significance of automated RE for the software development community, which has the potential to shorten development cycles and reduce associated costs. The support tools primarily assist in generating UML models (44.7%) and other activities such as omission of steps, consistency checking, and requirement validation. The analysis phase of RE is the most widely automated phase, with 49.53% of automated tools developed for this purpose. Natural language processing technologies, particularly POS tagging and Parser, are widely employed in developing these support tools. Controlled experimental methods are the most frequently used (48.2%) for evaluating automated RE tools, while user studies are the least employed evaluation method (8.2%). This paper contributes to the existing body of knowledge by providing an updated overview of the research literature, enabling a better understanding of trends and state-of-the-art practices in automated RE for researchers and practitioners. It also paves the way for future research directions in automated requirements engineering.

Keywords Requirements engineering · Automated RE · Automation · Support · Systematic literature review

1 Introduction

Software Development is a complex process with ever-changing technologies and requirements, primarily due to business process changes. A critical activity in software development is Requirements Engineering (RE), which focuses on software systems' real-world objectives, functions, and constraints [1]. Requirements engineering has high significance for the quality of the software product. Owing to its inherently complex and interdisciplinary nature, RE is a challenging field in software and systems development; it is crucial for development success [2]. The RE process emphasises the systematic and recursive techniques that ensure system requirements' completeness, consistency, and relevance [3]. This implies that the RE process aims to establish a robust foundation for system development, and the process strives to minimise errors, enhance stakeholder communication, and increase the chances of developing a successful system. Requirements engineering is a crucial process that involves capturing and analysing stakeholder needs, and natural language documents serve as one of the compelling mediums

✉ Muhammad Aminu Umar
aminu.umar@kcl.ac.uk

[1] Department of Informatics, King's College London, Strand, London WC2R 2LS, UK

[2] Department of Computer Science, Ahmadu Bello University, Zaria, Nigeria

Published online: 03 February 2024

for expressing and documenting those requirements over the years.

Natural language (NL) plays a significant role in requirements engineering as it is the primary means of communication between stakeholders and the software development team. This accounts for 79 per cent of all documents [4]. Software requirements are primarily written in natural language [5–7]. Therefore, the role of NL in requirements analysis and documentation cannot be overemphasised, as already highlighted in a study by Ryan in 1993 [8]. Although natural language is inherently object-oriented and descriptive, possessing strong representation power, its syntax and semantics are not formal enough to serve as a programming language; thus, requirements documentation written in NL needs to be reinterpreted into a formal specification language by software engineers [9]. While natural language requirements can present challenges, such as ambiguity or subjectivity, employing systematic approaches and leveraging natural language processing (NLP), techniques can improve the effectiveness and efficiency of the RE process in dealing with natural language artifacts. Natural language processing provides essential techniques for extracting information from descriptions of textual requirements, such as use cases, scenarios, user stories, and transcripts of conversations for requirements elicitation [10]. The applications of NLP for RE are studied in [11], which identifies six categories that encompass the possible activities of NLP in requirements engineering: classification, prioritisation, ambiguity removal, requirements elicitation, requirements assessment, and requirements analysis. To achieve these activities, various automated frameworks and tools have been proposed and developed. Automated requirements engineering aims to reduce the amount of time and labour expenses of RE while maintaining precise and thorough requirements.

Automated requirements engineering refers to using software tools and techniques to support and automate eliciting, analysing, specifying, validating, and managing software requirements. These tools can help streamline and optimise the requirements engineering process, which can be complex and time-consuming. The concept of computer-aided requirements engineering can be traced back to the 1980 s, as early works by Teichroew and Sayani [12] predicted that requirements engineers would eventually need to transition from manual methods to computer-aided approaches, much like how programmers have replaced manual programming with online programming. The goal of automation in requirements engineering is to minimise the time, effort, and cost involved in the RE process, as well as software development in general, while simultaneously enhancing the quality and accuracy of the requirements. This has been demonstrated in several research works in the literature, encompassing various aspects of requirements engineering activities. Automated tools such as *DoMoBoT* [13], *TRAM* [14], *TestMEReq* [15], *ScenarioAmigo* [16], *SUGAR* [17], *CM-Builder* [18], *aTouCan* [19], *Requirements-Collector* [20] among others, have been developed and empirically evaluated. However, it is essential to note that these tools have yet to be widely adopted on an industrial scale. Despite this, industry experts and the research community recognise the need for tool support to automate requirements engineering activities, particularly in automatically generating Unified Modelling Language (UML) diagrams [21]. These tools leverage Artificial Intelligence (AI) capabilities to automate tasks such as requirements elicitation, analysis, validation, and management. Integrating automated tools and AI allows for more efficient and effective handling of complex requirements, improved accuracy, and the potential for innovative approaches in software development.

Considering the vast potential of automated requirements engineering, our objective is to investigate the advancements achieved thus far in various aspects of these tools, including the generated output, RE phases, extent of automation, and tool evaluation methodologies. The motivation for this study arises from the substantial impact of requirements engineering on software quality. Specifically, we aim to examine the influence of automated RE on the requirements engineering process and determine the added value compared to traditional RE approaches. Notably, there has been a significant increase in research dedicated to automated RE recently. Consequently, this paper presents a systematic literature review to identify, evaluate, and analyse existing research on automated requirements engineering. Thus, the contribution of this study is summarised as follows: (i) Comprehensive Investigation: the systematic review explores various dimensions of automated RE tools, including their output, the phases of RE they target, the degree of automation they offer, and the methodologies used to evaluate these tools. (ii) Motivated by Software Quality: the study is motivated by the substantial impact that RE has on software quality. It seeks to understand how automation influences the RE process and assesses its added value compared to traditional RE methods. (iii) Timely and Relevant: given the recent surge in research focused on automated RE, this paper addresses the current state of the field. It systematically analyses existing research, offering valuable insights into the advancements and trends in automated RE. Consequently, it provides updated literature in the field.

The remainder of this paper is organised as follows: Sect. 2 discusses the theoretical background, focusing on traditional versus automated RE and the review of related works. Section 3 describes the methodology employed for the literature analysis. Section 4 presents the results of the literature analysis. Section 5 provides a discussion of the results. Finally, Sect. 6 concludes by explaining the limitations of the study and highlighting prospects for future research and directions.

2 Background

Software requirements describe the structure of a software application's development process and the software's primary goal and objectives for the development team. Thus, providing information that specifies the outlook of a software application. The process of creating, documenting, and maintaining requirements statements is referred to as requirements engineering. It is based on a framework that outlines the key structural components and process elements. It also offers a solid foundation for the process fundamentals, guiding concepts and methods, not software project- or development methodology-specific. The framework proposed by [22], which is shown in Fig. 1, is an example. To build a vision within an existing context, the framework identifies the key structural components of a requirements engineering process. The building block that makes up the framework are System context; The three fundamental requirements engineering activities (Elicitation, documentation, and negotiation); Two cross-sectional activities (Validation and management); and Requirements artefacts (Goals, scenarios, and solution-oriented requirements). Over the years, there has been significant improvement in the methodologies of RE processes. These processes range from sequential (traditional RE) to more iterative and incremental (agile RE). The traditional requirements engineering process's primary objective is creating a system requirements document for knowledge transfer. In contrast, agile techniques emphasise direct interaction between customers and agile teams to achieve a similar goal. Depending on the application area, the individuals involved, and the organisation providing the requirements, several requirements engineering procedures are employed [23]. According to Sommerville, all processes have the following generic activities (phases) in common:

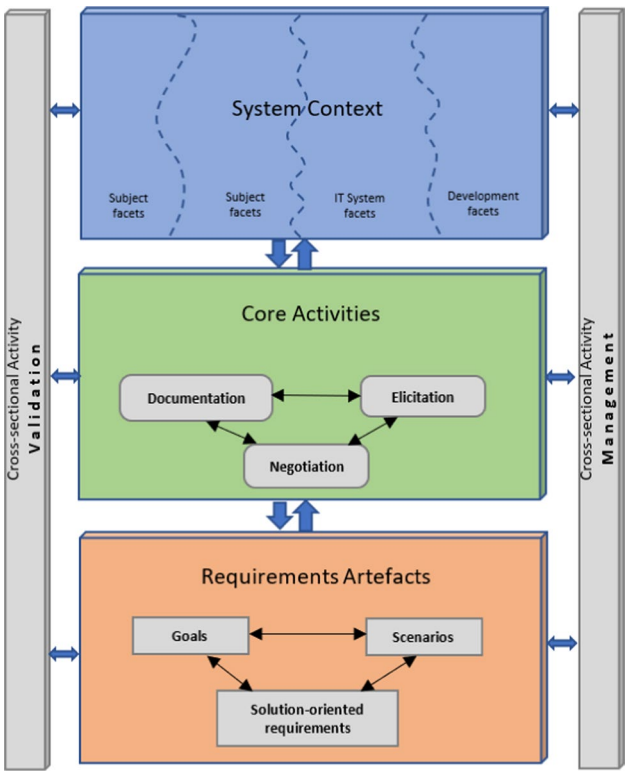

Fig. 1 RE framework [22]

- *Requirements elicitation.* The process of determining and gathering requirements from sources such as stakeholders. Both functional and non-functional requirements are included here.
- *Requirements analysis and specification.* This is the logical decomposition and structuring of the elicitation's process. It involves having a thorough understanding of the requirements and organising both the textual information and the derived requirements into model diagrams and written documentation.
- *Requirements validation.* Ensures that information is gathered accurately and is organised well to fulfil system business goals. This is accomplished by verifying the documents' accuracy, completeness, and correctness and/or models that describe requirements.
- *Requirements management.* This keeps track of changes in requirements and ensures that those changes are made to meet stakeholder's requirements.

Traditional requirements engineering has mostly been based on the idea that requirements exist implicitly in stakeholders' minds and have focused on models and procedures to help identify and document such requirements [24]. However, with the rapid growth and dynamics in the software market, rapid changes in the business processes and increases in customer demands, there is a need to adopt methodologies that will accelerate software development.

2.1 The traditional requirements engineering process

Requirements engineering in traditional software development methodologies, such as the waterfall model, plays a crucial role in the early stages of the development lifecycle. It involves systematically gathering, analysing, validating, and managing software requirements. The waterfall model process activities are performed in a sequence of separate steps where preferably each step is finished before the next one begins [25]. A primary characteristic of this approach is that the software is detailed up-front. Since each project stage must be finished before moving on to the next, this leads to much documentation. The traditional approach starts with gathering and documenting a "complete" set of

requirements, then moves on to architectural and high-level design, development, testing and maintenance [23]. Figure 2 depicts the development phases in the waterfall model.

The RE activities typically constitute an earlier part of the waterfall model before development activities begin, and it is a reference point for subsequent development stages. During RE in the waterfall model, requirements are often gathered through interviews, meetings, and stakeholder discussions [25]. These requirements are then documented in a requirements specification document as a baseline for the entire development process. One of the critical characteristics of requirements engineering in the waterfall model is its linear and sequential nature. Once requirements are finalized and documented, they are expected to stay the same throughout development. Any modifications or additions to requirements may require formal change requests and impact the project timeline and budget.

Requirements engineering is referred to as the first stage of the development process in early waterfall models of software development. However, more recent approaches to software development (such as the Rational Unified Process and Agile, among others) assume that requirements engineering continues across the system's whole life cycle [26]. The plan-driven process nature of the waterfall model made it compulsory to schedule and plan all the process activities before starting work on them. This usually resulted in colossal documentation. The model, however, is appropriate when the requirements are precise and unlikely to change dramatically during system development. The increasing need to address the limitations of the traditional requirements engineering process and software development has given birth to methodologies like agile software development (ASD). Additionally, customers need help explaining their requirements up front clearly. More so, the industry and technology evolve too quickly, and requirements change at rates that overwhelm established traditional methodologies [27]. Agile development is characterized by quick, iterative, and incremental development. Some standard requirements elicitation techniques in agile development are interviews, user stories and rapid feedback. Text mining, an automated technique for generating requirements documents, has gained recent attention [28]. These methodologies have significantly contributed to automating requirements engineering activities and software development.

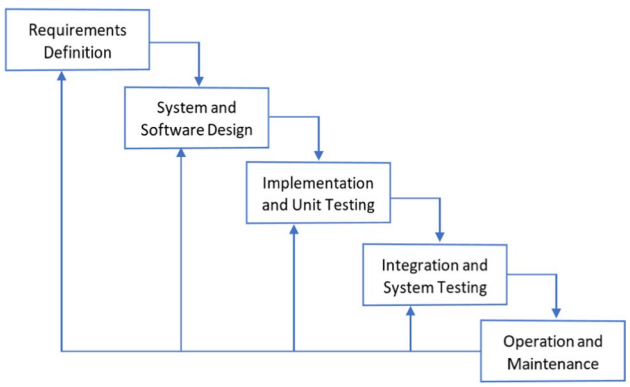

Fig. 2 Waterfall model [23]

2.2 Automation of software engineering processes

Automated Software Engineering represents a critical research area within software user requirements, as it centres on automating software processes to enhance the quality and productivity of software development [5]. Automation of software engineering processes involves using tools and techniques to automate various tasks involved in the software development process. These tasks include requirements analysis, software testing, software maintenance, code review, and code analysis. Automation in software engineering processes has become increasingly popular over the years, as it can help improve the quality of software development processes, reduce development time and costs, and increase the efficiency of software development teams. In recent years, there have been many research efforts in the automation of software engineering processes, and several tools and frameworks have been developed to automate various tasks involved in software development, especially in requirements engineering processes. Most of the tools reported in the selected primary studies are products of research efforts in RE. Additionally, other software engineering aspects have their share of the research outputs. Test data generation [29, 30], code generation [31, 32], code analysis [33], software testing [34, 35], and software maintenance [36] have all been areas of significant study. However, this study focuses on works related to automated requirements engineering.

Automated requirements engineering is a field of software engineering that leverages automation techniques to enhance the effectiveness and efficiency of the RE process. Automated RE encompasses various activities within RE, such as requirements elicitation, analysis, documentation, validation, and management, which can be automated using NLP, machine learning (ML), knowledge-based systems, and other artificial intelligence techniques. These techniques and approaches have provided a new paradigm in the RE subdomain and software development in general. Even though the tools that will help analyse requirements automatically are still evolving through research, especially in using NLP and other AI techniques to improve the quality, efficiency, and consistency of the RE process. The existing commercial graphical CASE tools significantly help document the output of the Analysis and Design phases of software development and aid in identifying incompleteness and inconsistencies within an analysis [18]. Nevertheless, these tools do not contribute to the challenging initial stage of the analysis process,

which involves identifying the object classes, attributes, and relationships utilised to model the problem domain.

Therefore, the systematic literature review (SLR) presented in this work aims to analyse and evaluate the available research on automated RE comprehensively. The findings of this review provide a valuable resource for researchers and practitioners seeking to understand better the state of the art in automated RE and its potential applications in software development. In this study, we consider studies that attempt to automate one activity of RE or the other. Typically, these activities have traditionally been carried out manually by human experts, and instead by automation a tool has been employed to do such activities on behalf of humans—for example, requirements elicitation, classification, modelling, checking requirements inconsistencies and duplication, among others. In automated requirements engineering, support tools are used to facilitate the activities of various components of the requirements engineering framework, which constitute the majority of the activities of RE. Over the years, many research efforts have been automating RE activities from elicitation to requirements management. These automation activities cut across various domains, for example, mobile [37], scientific systems [38], security [39], crowdsourcing [40], open-source applications [41], safety–critical systems [42], and train protection [43]. Accordingly, the automated support comprises both core framework activities and requirements artefacts.

2.3 Application of natural language processing in RE

The services a software system should provide to satisfy the needs and preferences of stakeholders are typically described in software requirements, which are often expressed in natural language [24]. For a long time, natural language has played a significant role in requirements engineering and software development in general. However, natural language is often a poor choice for representing requirements because of its innate propensity for ambiguity, inconsistency, redundancy, etc. [44–46]. Despite these shortcomings, natural language is best for experimentation and communication. It is a tool that human minds have evolved over millennia to use for just that [47]. Natural language processing techniques facilitate text processing which aids RE activities. Requirements elicitation is supported by extracting relevant lexical entries from the vast textual data generated by elicitation techniques (stakeholder interviews, group meetings, protocol analysis or participant observation) [10]. Some NLP technologies and techniques include rule-based techniques, Part-Of-Speech (POS) tagging and Lexico-Syntactic Pattern (LSP), and Speech act-based analysis techniques, among others.

On the other hand, despite the recent intensive efforts to produce formal specifications and automated toolkits, the practical significance of requirements documents expressed in natural language and the demand for user participation throughout the software development life cycle remain. Nevertheless, the ambitious objective of automation of RE activities is promising and has a stake in the future of requirements engineering. A classic example is the selected primary studies reported in this systematic review. The current state of the art of NLP technology has proven it is possible to automate requirements analysis and save significant time analysts spend [48]. Most studies reviewed in this study have applied one NLP technique or the other to demonstrate how RE activities can be automated.

2.4 Summary of related literature reviews

To understand what has been previously researched and avoid duplication of research efforts in the identification, evaluation, and analysis of literature through a systematic literature review, we searched literature reviews and surveys in this subject area. This process helps us gain insights into the current state of research and establish the need for the SLR. Following an extensive search, we discovered publications that have conducted systematic reviews or surveys on various aspects of software development pertaining to requirements engineering. This section presents examples of literature reviews related to automated RE found in software engineering research literature. The summary of these reviews is presented in Table 1. Consequently, we summarise the distinctions between these previous studies and the present one in the final paragraph of this section.

In 2011, Yue et al. [49] systematically reviewed approaches for transforming user requirements into analysis models. The review focused on studies that converted textual specifications into models for analysis, especially from publications from 1996 to 2008. The analyses established 16 techniques from the literature that were employed by numerous publications reviewed. In analysing the findings, they found that no existing methodologies were sufficiently efficient, could automatically or semi-automatically produce a complete and consistent analysis model, or needed acceptable user efforts to specify requirements. They also noted that the majority of methods did not handle traceability. As a result, they advocated for developing a traceability system to establish and maintain relationships between requirements items and analysis model elements. However, the impact of the transformation approaches on RE phases was not mentioned.

In 2012, Carrillo de Gea et al., [50] surveyed to assess the capabilities of requirement engineering tools. The survey's objectives were to gain insights into the extent to which requirements engineering tools support the requirements engineering process. The findings revealed that most tools were delivered under proprietary licenses and were

Table 1 An overview of related work

Reference	Year	Goal	Research questions
Yue et al. [49]	2011	Examine published works that create analysis models from textual requirements	RQ1: What various techniques are employed to transform requirements into analysis models? RQ2: What are the present constraints on these methods? RQ3: What are the unresolved problems that require more research?
Carrillo de Gea et al. [50]	2012	Depict the state-of-the-art of RE tools	Do current Requirements Engineering tools adequately support the RE process?
Meth, Brhel et al. [51]	2013	Review state of the art research in automated requirement elicitation	What is the state of the art in research covering tool support for automated requirements elicitation from natural language documents?
Yang et al. [52]	2014	Review modelling techniques, requirements engineering activities, and domain of application areas	RQ1: How are publications distributed in terms of time, place, research group, and geographic region? RQ2: What modelling techniques and RE activities are being investigated? RQ3: Which application domains and requirements quality attributes are involved? RQ4: Which techniques are more effectively utilized and subject to more thorough evaluation? RQ5: Which RE activities are presented and covered in greater detail? RQ6: What topics can we generalize about based on the content provided the selected studies? RQ7: How do modelling techniques and topics relate to one another?
Abdouli et al. [53]	2016	Review existing approaches and propose other alternatives for Requirement Engineering	To provide an overview of works on requirement analysis, as well as a comparison of these studies
Dawood and Sahraoni [54]	2017	Review the status of using NLP to process software requirements into UML diagrams	To establish the status of the application of NLP in processing software requirements to generate UML diagrams
Schön et al. [55]	2017	Describe the current state of agile RE with a focus on stakeholder and user interaction	RQ1: What approaches are available that involve stakeholders in the RE process and are appropriate for ASD? RQ2: Which agile approaches are available that can show stakeholders the viewpoint of the user? RQ3: What are the typical methods used in ASD for managing requirements?
Ahmed et al. [56]	2022	Review automatic transformation of Natural Language to Unified Modeling Language	RQ1: What are the existing approaches to automate the UML generation? RQ2: How effective are the existing approaches?
Kolahdouz-Rahimi et al. [57]	2023	Examine existing works on requirements formalisation using natural language processing and machine learning	RQ1: What are the most used NLP/ML approaches for automatic/semi-automatic requirement formalisation? RQ2: What are the input and output of RF approaches? RQ3: What are the gaps and deficiencies in existing requirement formalisation work?

typically not free. Furthermore, it was observed that requirements elicitation enjoyed more significant support regarding tools, while requirements modelling and management had the least support. The analysis presented are limited to proprietary tools.

Meth, Brhel, et al., [51] analysed the state of automated requirements elicitation through a systematic review. In the RE sub-domain, the review determined the state of automated requirements elicitation at the time. The review covered works on automated requirements elicitation from 1992 to 2012 by identifying and analysing 36 primary studies. The review conceptualises an analysis framework for works in automated elicitation and formulating future research directions. In contrast, we provide analysis of the development techniques and various RE phases the support tools address.

Through a systematic review, Yang et al. [52] studied requirements modelling and analysis for self-adaptive

systems. The review aimed to investigate past research modelling techniques, RE activities, requirements quality activities, research topics, and application areas. According to the findings, 16 modelling techniques were used in 11 RE activities, and the most frequently stated application domains were online applications and service-based systems. Additionally, they stated that most of the research was conducted in American and European countries, making them very difficult to apply appropriately in the discourse of requirements modelling in developing countries. Our work is not limited to modelling methods but what specific models were generated as output from the support tool.

The purpose of the study by Abdouli et al., 2016 [53] was to survey works that transform requirements into UML diagrams from early manual procedures in 1996 to automated tools in 2015. The research also examined the methods for transforming requirements into models, using five trends of classification: inspection, NLP, heuristics, patterns/graphs, and ontology. Furthermore, the advantages and disadvantages of the reviewed tools were highlighted. The authors concluded that the trend of integrating AI into RE was promising. However, in this report, the analysis of the tools reported are not systematic.

Dawood and Saraoui [54] conducted a survey study of requirements engineering to UML using NLP. The study comprised of a literature review of existing tools to establish their strength and weaknesses; and a survey with a questionnaire distributed among research groups, academia, and practitioners. The results revealed that most users/organisations still manually generated UML diagrams from NL. The researchers drew their conclusion on the understanding that more research in the field of NLP was necessary to construct UML diagrams automatically or semi-automatically efficiently. The analysis of strengths and weaknesses presented in the literature was deduced by the authors and not conducted systematically.

Schön et al. [55] examined agile RE through a systematic literature review. The review focused on agile requirements engineering, which investigated stakeholders' involvement in the process in the existing approaches. The methodologies used to offer user perspective and requirement management procedures were also investigated. The review covered empirical works from 1995 to 2015. The findings showed that ASD has a weak foundation for establishing a shared understanding of the user perspective. The review also revealed problems concerning the direct involvement of users and stakeholders in ASD. As a result, they identified four approaches incorporated into ASD by the selected publication to understand user needs better. The approaches were human-centred design, design thinking, contextual inquiry, and participatory design. The review is limited to agile requirements engineering.

Ahmed et al. [56] studied the automatic transformation of natural language to UML. The study focuses on various approaches used in the transformation by highlighting their pros and cons and metrics. The study also proposed a conceptual framework with further improvement guidelines. The study identifies 70 primary studies covering from 1994 to 2021. However, the review did not take into account the impact of these tools on the RE phase and evaluation approaches for the tools selected in the primary studies.

Through a systematic review, Kolahdouz-Rahimi et al. [57] examine requirements formalisation using NLP and ML. The study reviewed 47 relevant studies and reported that heuristic NLP approaches and deep learning are the commonly used techniques for requirements formalisation using NLP and ML respectively. The study also reported difficulties in comparing the performance of different formalisation approaches due to the absence of standard benchmarks cases for requirements formalisation. The study focused on assessing the evaluation criteria without discussing the analysis methodologies.

While conducting an extensive systematic literature review, our study endeavours to compile and analyse existing evidence concerning automated requirements engineering and its potential implications for future research. In contrast to the previously mentioned studies, the present study delves into the specifics of automated RE activities across distinct phases of the RE process. Unlike previous literature reviews, which offered a more generalized perspective, our study scrutinizes the automation efforts pertaining to key RE phases, including elicitation, analysis and specification, validation, and management. Furthermore, we aim to shed light on the diverse technologies and techniques employed in the automation process, as well as the resultant outcomes produced by supporting tools. As a result, our study seeks to establish connection between automation and the different activities of RE within the realm of software development. Our objectives include exploring the extent to which automation has been integrated into various stages of the software development lifecycle, its significance, identifying best practices, evaluation methods, and assessing the value it has contributed to the RE domain compared to traditional RE approaches.

3 Methodology

Evidence-Based Software Engineering seeks to enhance decision-making concerning software development and maintenance by fusing the most recent research's best evidence with real-world knowledge and ethical principles [58]. The systematic literature review, which focuses on identifying, classifying, and evaluating the extent of research in a particular study topic, is a crucial technique

for evidence-based software engineering. Accordingly, the systematic literature review is a common and frequently used methodology in software engineering [59]. Consequently, this review aims to identify, assess, and interpret all the available research related to automated requirements engineering support. Therefore, we adhered to the recommendations made in [60] for technical relevance and reliability. Thus, we needed to outline the three critical phases of our systematic review: planning, conducting, and documentation. Figure 3 shows a typical research methodology for systematic literature review, which aided our review.

3.1 Research questions

This study investigated the empirical evidence of automated RE support. As a review, it focused more on answering the following research questions and their respective motivation, as provided in Table 2. The SLR is limited to the investigated and reported implemented tools to help in requirements engineering activities rather than the technicalities of the approaches used as published in the primary studies.

3.2 Search strategies

Once the research questions were established, a set of keywords aligned with the research objectives was selected and used for the search. The search strategy was refined to ensure the identification of relevant information that would facilitate an effective and comprehensive investigation of the research questions. Typically, this process involves four steps: selecting appropriate digital libraries, identifying additional search sources, determining relevant search keywords, and establishing the time frame for published articles. For this study, a search was performed across four databases: ACM, Elsevier (Science Direct), IEEE Xplore, and Springer. The decision to utilise these databases was based on their accessibility, extensive coverage, and specific relevance to the topic. These selected databases offer a wide range of scholarly resources that are highly relevant to the research topic.

- ACM digital library (www.dl.acm.org)
- Elsevier (www.sciencedirect.com)
- IEEE Xplore (www.ieeexplore.ieee.org)
- Springer (www.springerlink.com)

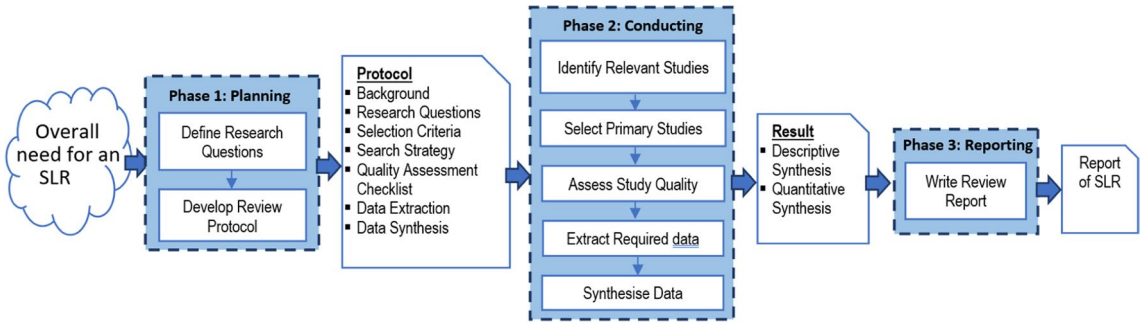

Fig. 3 Research methodology for SLR

Table 2 Research questions

No	Research question	Motivation
RQ1	What type of output or models are generated by the automated tools, and what is the added value according to published empirical studies?	Answering this question helped us to identify and establish the various outputs produced by these support tools to facilitate requirements engineering activities
RQ2	Which requirements engineering phase is mostly automated?	With this question, we were interested in identifying which of the phases of requirements engineering researchers were interested in having an automated support tool
RQ3	To what degree is the automation of the requirements engineering support tool?	Answering this question helped us to identify the extent of automation of the RE activity according to the published empirical studies
RQ4	What are the development techniques/ approaches employed in developing the automated tools?	For this question, we were interested in identifying the development techniques and technologies used in the proposed tools
RQ5	How are these proposed automated tools evaluated?	Answering this question helped us identify the various evaluation methods employed to evaluate the proposed tools

The major goal of the procedure was to find potential primary study candidate papers. Applying search parameters, the title and the entire article text were searched. Inclusion and exclusion criteria were used to filter the articles published between 1996 and 2022, which were the only ones that were found through the search. Details of the inclusion and exclusion criteria are provided in Sect. 3.3.

Search terms/keywords were utilised to create search strings to get the desired result and make the search process more manageable. A comprehensive search was completed in December 2022 using the final combination of search terms interchangeably. *"requirements engineering" OR "requirement elicitation" OR "requirement gathering" OR "software requirements engineering" OR "requirements identification" OR "requirements analysis" OR "requirements specification" OR "requirements modelling" OR "software modelling" OR "modelling" OR "requirements documentation" OR "requirements validation" OR "model extraction" OR "model generation" OR "requirements verification" OR "requirements management", AND (automat* OR Computer Aided OR Computer Supported Software Engineering OR computer-assisted).*

These search strings were applied manually in each of the databases, that is, based on the search functionality provided by the database. Additionally, we treated each database search as a learning and experimenting process. The first author conducted the literature search, data extraction, and analysis, while all authors undertook the review and quality analysis.

3.3 Inclusion and exclusion criteria

After screening, the eligibility for selecting primary studies was determined by applying inclusion and exclusion criteria. Only studies that provided empirical data and findings showcasing the implementation of automated requirements engineering tool assistance for requirements formalization were included to address the research questions.

The inclusion criteria: studies between 1996 and 2022. This date range was defined based on the observation that early work on automated RE emerged in 1995. Further, English language studies; studies relevant to the specified search string; peer-reviewed research, as well as original studies were the other criteria.

While the exclusion criteria included: the primary work that was not published as a chapter, or in conference proceedings or journals; duplicate papers (papers with conference and journal versions). More so, the subject was not directly related to automated RE support; editorials, keynotes, and short papers (less than three pages) were not included.

3.4 Quality assessment

A quality assessment is carried out during the review process to evaluate and validate the primary studies that were previously identified. The fundamental goal of the authors' quality assessment was to ensure (at least to some degree) that our findings would be supported by good-quality empirical research. According to Daun et al. [61] there is no common standard for quality assessment. However, the commonly suggested quantitative approach to quality assessment is by including publications that have been peer reviewed. Therefore, the quality assessment criteria used by [62] were adopted. Accordingly, the quality criteria offer a method for selecting suitable studies that would add to the significance of this research and were critically considered. As shown in Table 3, some of the quality assessment questions related to the primary studies' minimal quality threshold, rigour, credibility, and relevance. The degree to which we could be confident that the results of a particular study could significantly advance the review was measured by these criteria. A dichotomous scale ("yes" or "no") was used to grade the quality assessment criteria, where 1 represents Yes and 0 represents No. As a result, studies with at least one "no" response to the first three questions were disregarded because this review needed to meet a minimum quality threshold. Seven articles were eliminated when the quality assessment criteria were applied. As a result, 85 research in all were chosen as primary studies, accounting for 92.4% of all the studies that had their quality evaluated. Any disagreements were resolved through discussion among the authors.

Table 3 Quality assessment criteria adopted by [62]

Quality threshold	Quality assessment questions
Minimum review quality requirement	1. The reported study is a research paper 2. The stated aim and objectives were crystal clear 3. The setting in which the research was conducted was adequately described
Rigour	4. The research design was suitable for addressing the research's objectives 5. There was an adequate description of the methods for data analysis
Credibility	6. The study offered succinctly expressed findings supported by reliable data
Relevance	7. They contributed to practice or research

3.5 Primary study

After reviewing the studies, we found 85 primary papers ranging from 1996 to 2022. These studies were collected from journal articles, conference proceedings, and book chapters. In the initial stage, 3853 studies were found by applying the search string to the publication databases. The studies were cut down to 425 after title-based selection. Phase 3 saw the short-listed studies reduced to 268 after reading the abstract and eliminating duplicate studies. Following thorough text evaluation, 85 studies were determined to be total. Figure 4 details all four stages of the selection process. Table 9 lists the identified primary studies (Appendix).

3.6 Data collection/extraction

Data collection is one of the crucial stages in a systematic review, and it is performed to extract information from each selected primary study to address our research questions. Data were collected manually from each of the primary studies. The authors went through each paper and manually extracted data into a prepared spreadsheet template. Afterward, the extracted data were collectively reviewed, and any conflicts were discussed and resolved. Subsequently, the data were analysed as a group, and a summary is presented in the results section. Furthermore, we have summarised the collected data, which is presented in Table 9 in Appendix. The data extracted were technically divided into three:

- Foundational details: (title, authors, publication date and location).
- Publication data: Journal, conference, chapter, date (year), publisher, publication title, volumes, issues, pages, keywords, and abstract.
- Review data: tool development approaches/techniques, tool input source, RE phase, model proposed, degree of automation, type of requirements, tool evaluation method.

From the preceding, we therefore, present the report and the findings and statistical analysis of the SLR gathered from the selected primary studies in the next section.

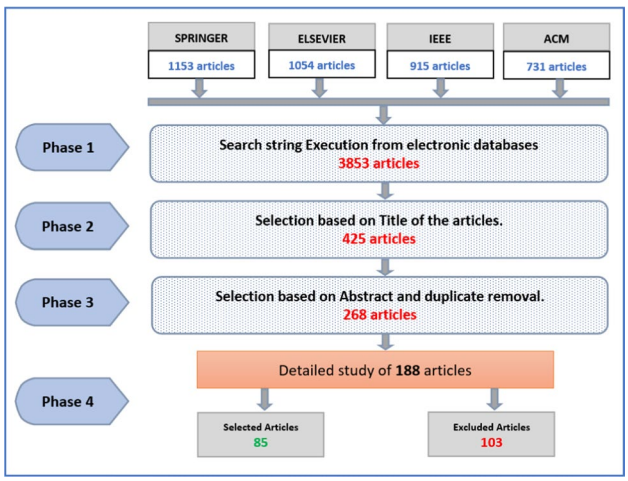

Fig. 4 Phases of selection process

4 Results

This section answers the research questions posed in Sect. 3.1 and highlights the review's findings.

4.1 Overview of the studies

As mentioned earlier, a total of 85 studies were identified for critical evaluation concerning automated RE support. The data extracted is summarized in Table 9 (Appendix). The table summarises each study selected, along with the study ID, author(s), reference, study title, year of publication, publisher, name, and citation type.

With respect to the years of publication, the first publication we reviewed was published in 1998. From other publications, works from 2016 and 2019 recorded the highest number, followed by publications from 2009, 2015 and 2017. The distribution of the reviewed papers spanned 1998 to 2022. The distribution of publications over the given period is shown in Fig. 5. This shows a generally increasing trend in the number of publications over time.

The findings in Table 4 indicate that the studies were published in various venues. Springer accounts for 38.8% (33) as the highest publishing venue for automated RE papers. Then, IEEE and Elsevier account for 32.9% (28) and 17.6% (15), respectively. On the other hand, ACM accounts for 10.6% (9) of the total selected primary studies.

The selected primary studies were mainly journals, conferences, and chapter publications. Of the 85 primary studies selected, 41 appeared in conferences, 33 in journals and 11 in book chapters, as presented in Fig. 6. On journal-specific publications, Requirements Engineering Journal has the highest number of papers associated with automated RE. The number of selected primary studies in each publication venue is shown in Table 5.

Accurately capture the geographical location of authors, we considered each author's location individually, as some of the papers had co-authors from different locations. Notably, most authors were from Europe and Asia, as depicted in Fig. 7. On the other hand, there were relatively few studies on automated requirements engineering from Africa, Australia, Oceania, and South America. Most of these publications are interconnected through citation,

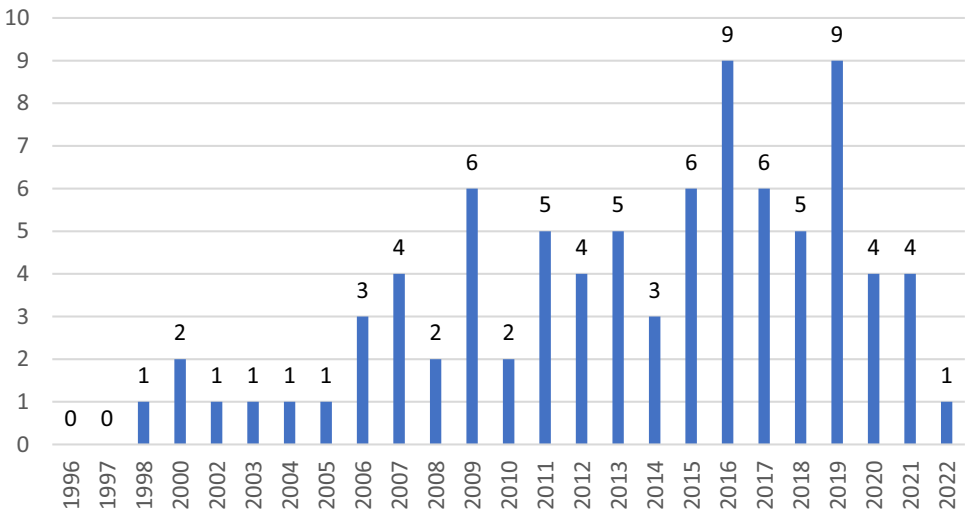

Fig. 5 Distribution of publication over the years

Table 4 Number of publications in each database

Publication	Frequency
ACM	9
Elsevier	15
IEEE	28
Springer	33
Total	85

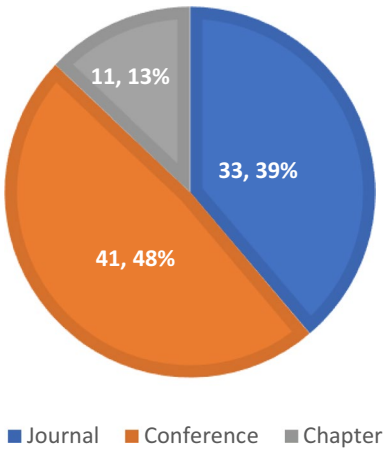

Fig. 6 Distribution of papers by citation type

forming a network of references, as illustrated in Fig. 8. These connections among the articles through citations play a crucial role in establishing the research context, demonstrating scholarly contribution, strengthening validity and reliability, tracing research evolution, and building a knowledge network within the systematic literature review.

4.2 What type of output or models are being generated by the automated tools according to published empirical studies?

One of the communication tools between the RE team and the software development team is the UML models, as they provide a common language and understanding of the requirements. UML models are a valuable tool in the requirements engineering process, as they help to ensure that the requirements are well-defined and accurately capture the needs of the stakeholders. In this study, several UML models were generated as output by the support tools reported in the primary studies. Figure 9 shows some of the models produced by these automated tools. As depicted by the chart, the class diagram (domain model) is the most widely produced model among the tools, followed by the use case diagram and the structured requirements document. However, some studies did not specify any model output, constituting the highest number of primary studies. This category of studies mainly focused on several RE activities, which included improving one RE process or the other. For example, recommendation of omitted steps in use case scenario [16], requirements mining framework [38, 63–65], checking inconsistency of requirements [66, 67], requirements validation and review [15, 43, 68, 69], requirements classification [40, 41, 70–73], audio mining and visualization [74], duplicate requirements [75], automated requirements reuse [76], systematic analysis of NL [42, 47], enhancing the security of RE [37, 39]. In total, UML diagrams generated as output constitute 44.7% (38), while others with no model output made 55.3% (47) of the primary studies.

An automated RE tool receives various inputs to facilitate its process. These inputs are the foundation for the tool to perform its automated functions effectively. Some common inputs to an automated RE tool include natural language requirements; domain knowledge; existing models or

Table 5 Classification of the selected publications

Name	Citation type	Publisher	Total
Requirements engineering	Journal	Springer	11 (12.9%)
Automated software engineering	Journal	Springer	6 (7.1%)
Information and software technology	Journal	Elsevier	5 (5.9%)
The journal of systems and software	Journal	Elsevier	4 (4.7%)
Software and systems modeling	Journal	Springer	3 (3.5%)
IEEE/ACM international conference on automated software engineering	Conference	IEEE/ACM	5 (5.9%)
International requirements engineering conference	Conference	IEEE	4 (4.7%)
India software engineering conference	Conference	ACM	2 (2.4%)
ACM/IEEE international conference on model driven engineering languages and systems	Conference	ACM/IEEE	2 (2.4%)
International conference on information technology	Conference	IEEE	2 (2.4%)
International conference on software reuse	Conference	Springer	2 (2.4%)
International working conference on requirements engineering: foundation for software quality	Chapter	Springer	2 (2.4%)
Others	Journal, conference & chapter		37 (43.5%)
Total			85

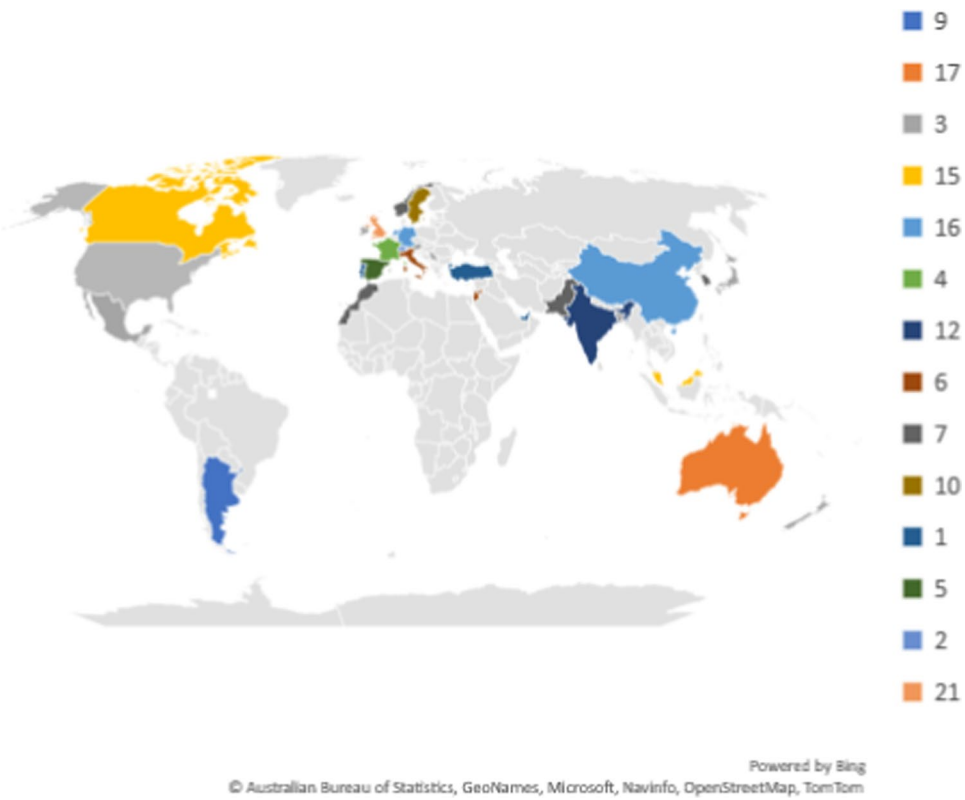

Fig. 7 Distribution of authors geographic locations

artefacts; stakeholder feedback; and metadata. It is important to note that the specific inputs required by an automated RE tool may differ based on its capabilities and functionalities. From our findings, most inputs used by these tools are written in natural language, including unstructured [13, 77, 78], semi-structured text [75, 79, 80]–[82], and structured text [7]. Others are domain knowledge [83], business process models [28] and feedback through voice [74]. In general, natural language text is the most prominent input for most tools, accounting for 94% of their usage. The automated requirements engineering software tools proposed in the primary studies are mostly one shot RE tool and a few editor tools. In conclusion, the value of automated RE generated UML and other outputs lies in improved visualisation, enhanced requirement analysis, early mistake identification, system design and development support, and improved traceability.

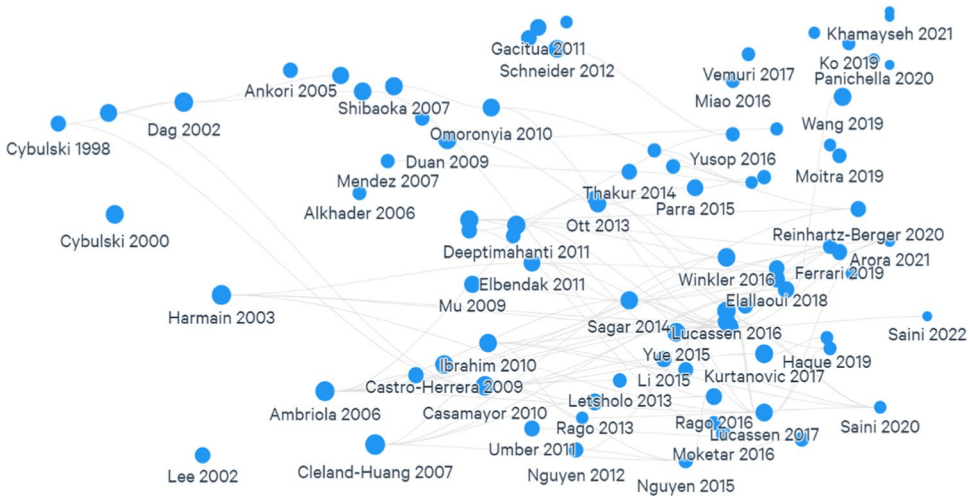

Fig. 8 Citation map of the selected primary studies

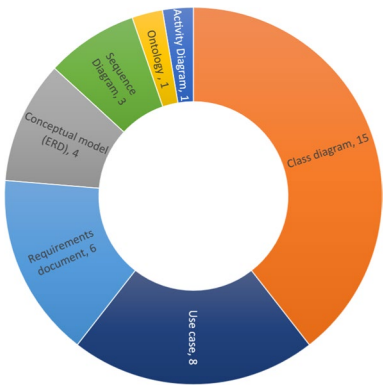

Fig. 9 Distribution of generated output

These advantages contribute to a more efficient, accurate, and collaborative RE process, resulting in better software systems that meets the needs and expectations of stakeholders.

4.3 Which requirements engineering phase is mostly automated?

Requirements Engineering is made of phases, also known as activities. As mentioned earlier and defined by Sommerville [23], the RE phases include requirements elicitation, analysis and specification, validation and management. The tools proposed in the primary studies have automated various tasks within the requirements engineering (RE) process, which can be classified into different RE phases. Although some studies mentioned the specific RE phase their tool supports, we still carefully examine each study for classification. Assigning a primary study to a particular RE phase can be challenging due to the possibility of tool activities spanning multiple phases. The type of the task and how it relates to the broader RE process can be used to categorise an automated RE task into a RE phase. Therefore, we define the following approaches to categorise the primary studies' automated RE tasks into a RE phase:

- Task Focus: determine the automated task focus to ascertain whether it is requirements elicitation, analysis, documentation, validation, or management. This classification aligns the task to the appropriate phase of the RE process.
- Input–Output Mapping: examine the automated task's input and output and relate it to the specific phases of RE. For example, suppose the automated task receives unstructured textual requirements and produces structured use case diagrams. In that case, it can be classified as an analysis task that transforms requirements into a more formal representation.
- Relationship to RE Activities: analyse the relationship between the automated task and the activities typically carried out during the RE phases. For instance, the activity can be classified as an elicitation task if it uses NLP techniques to extract key terms and concepts from requirements documents. This helps to capture and comprehend stakeholder requirements.
- Impact on Phase Objectives: consider the general goals of each RE phase and how the automated task contributes to accomplishing those goals. If the task's goal is to maintain requirement consistency and traceability, it may align with the validation or management phases, which emphasise quality assurance and requirement tracking.
- Integration with RE Processes: examine how the automated task fits into the more extensive RE processes or methodology. Determine whether it is a necessary aspect of a given phase or if it spans multiple phases. For example, if the activity involves automated requirements traceability analysis, it may be related to both the analysis and validation phases.

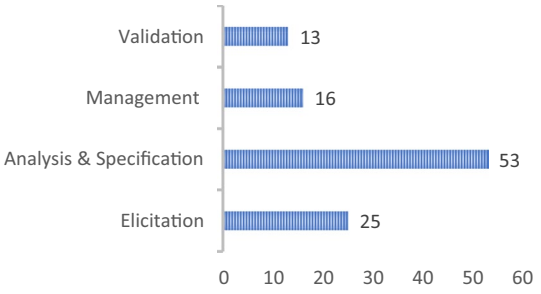

Fig. 10 Most automated RE phase

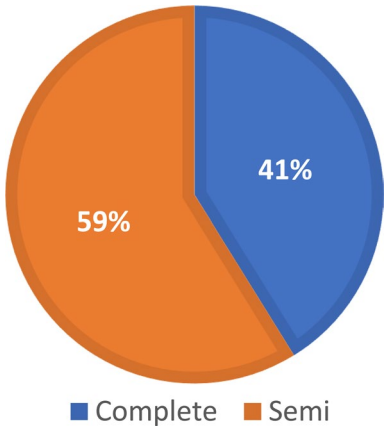

Fig. 11 Degree of tool automation

The approaches described above give a general outline for categorising automated tasks within the context of the RE phases, providing a better understanding of where the task belongs within the entire RE process. Therefore, the activity determined the requirements engineering phase for which each primary work was classified. According to published empirical studies, requirements analysis was the most widely automated phase of the RE activities, accounting for 49.53% (53) of the total selected primary studies. Requirements elicitation and management accounted for 23.36% (25) and 14.95% (16), respectively. On the other hand, requirements validation was the least automated RE phase resulting in 12.15% (13) of the total selected primary studies. Figure 10 shows the extent to which each RE phase has been automated. The analysis phase is critical because it ensures a complete grasp of stakeholder needs, refines requirements, fosters collaboration, establishes project scope, identifies risks, and acts as a basis for later development activities. Therefore, a well-executed analysis phase contributes considerably to the software project's success by providing a firm foundation for the development process and facilitating good communication and alignment among stakeholders.

4.4 To what degree is the automation of the requirements engineering support tool?

Automation is a complete or partial replacement of a task that a human operator previously performed, which suggests that automation can vary along a continuum of levels, from the lowest level of entirely manual performance to the highest level of full automation, rather than being all or nothing [84]. Thus, an automated RE support tool could be semi-automated or complete/full automation. However, this study focused only on automated tools semi or complete. Of the 85 primary studies selected, 50 used semi-automated tools, and 35 had complete automation, as represented in Fig. 11. All the literature reported in this review has implemented their respective proposed tools. The majority, representing 58.8% of the selected studies' artefacts, were semi-automated. In this case, the possible interaction between the human analyst and the tool varied. These variations ranged from input pre-processing, refining data during processing to output interpretation and analysis. In contrast, 41.2% of the primary studies carried out full automation. Despite the authors proposing full automation, none proposed complete replacement requirement engineers. However, the output of automated RE provided the human analyst with an artefact to be verified, thereby reducing development efforts, and saving a great deal of time spent on RE activities. Similarly, it was found that giving analysts an initial version of the requirements model based on informal textual descriptions could significantly speed up the development process because domain and modelling experts could begin modifying the requirements model much earlier [85]. In the primary studies, there is no specific mention of the extent of automation. This suggests that the primary studies needed to provide explicit information about the level or degree of automation achieved in their research.

In addition to classifying support tools reported as either semi-automation or full automation, we have explored the use of the ten levels of automation of decision and action selection and the four-stage model of human information processing proposed in [84]. Based on the system description provided in the primary studies, we have subjected each paper to evaluation to determine the extent of the automation based on the levels of automation design and action selection provided in Table 6. This was done against the four types/stages of automation of information processing, i.e., (i) Information acquisition (ii) Information analysis (iii) Decision selection, and (iv) Action implementation. Figure 12 shows the results of the levels of automation of various support tools reported in the primary studies. From the results, information analysis has a higher degree of automation than the other three stages. In summary, information analysis is a critical stage in information processing, and automation technologies like machine learning and

Table 6 Levels of automation design [84]

High	10	The computer decides everything, act autonomously, ignoring the human
	9	Informs the human only if it, the computer decides to
	8	Informs the human only if asked, or
	7	Executes automatically, then necessarily informs the human, and
	6	Allows the human a restricted time to veto before automatic execution, or
	5	Executes that suggestion if the human approves, or
	4	Suggest one alternative
	3	Narrows the selection down to a few, or
	2	The computer offers a complete set of decision/action alternatives, or
Low	1	The computer offers no assistance: human must take all decisions and actions

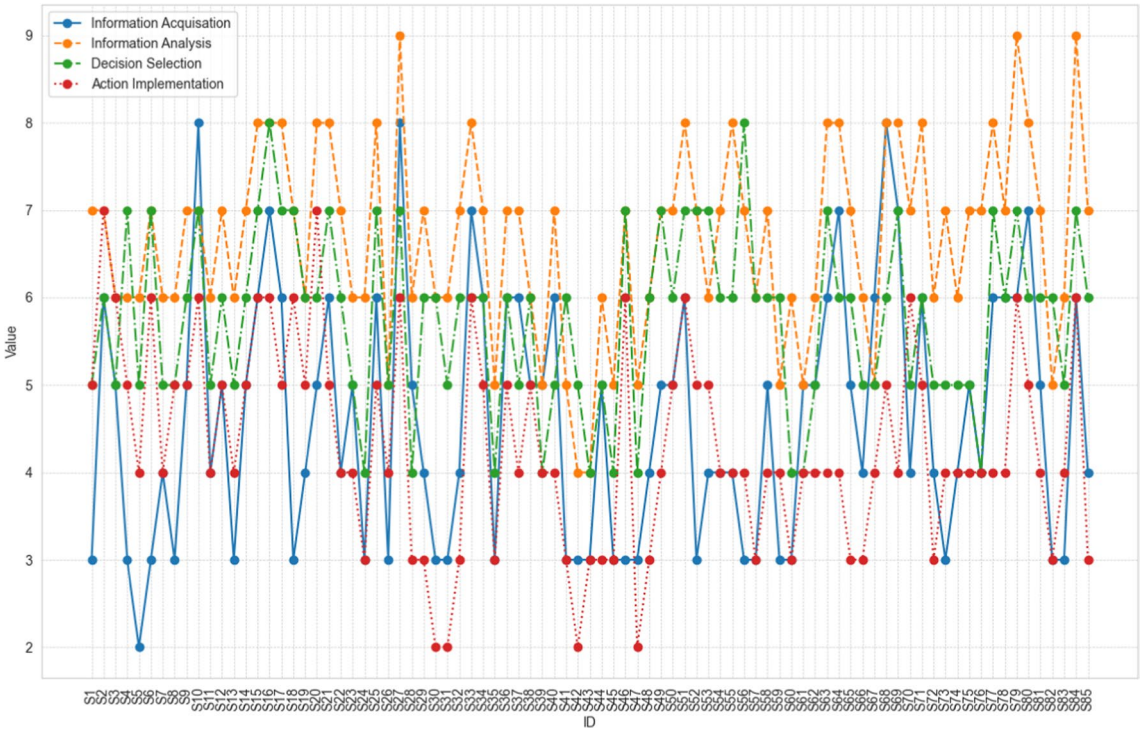

Fig. 12 Level of automation of Degree of tool automation

other AI technologies have advanced significantly in recent years, making it feasible to automate complex analytical tasks. This trend has led to more automation of information analysis than the other stages, enabling the extraction of valuable insights from data efficiently and making informed decisions.

4.5 What are the development techniques/approach employed in the development of the support tools?

In Fig. 13, the results are classified by development techniques or approaches employed by authors. Natural Language Processing techniques are the most widely used,

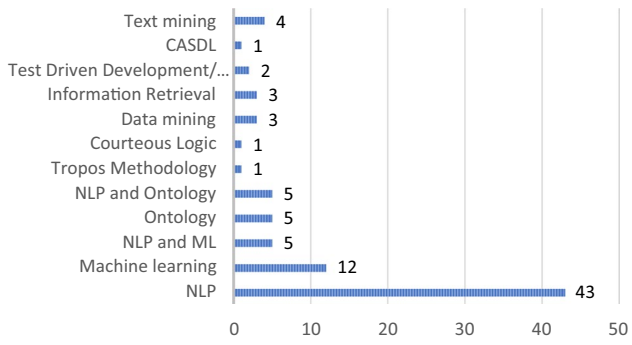

Fig. 13 Distribution of development technique/approach

accounting for 50.6% (43) of the selected primary studies. In the context of automated RE tools, NLP techniques are used to process and interpret natural language requirements, enabling automated understanding, extraction, and analysis of textual information. Machine Learning techniques recorded 14.1% (12) as a development approach, which involves the development of algorithms and models that can learn from data and make predictions or take actions without explicit programming for tasks relating to requirements classification, clustering, or prediction. On the other hand, domain ontology techniques accounted for 5.9% of the selected studies employed to capture and model requirements knowledge. These ontologies capture domain-specific concepts, relationships, and constraints, providing a shared understanding and common vocabulary for representing requirements within a particular domain. Studies that used text mining techniques are four which represent 4.7% of the studies. Some studies combined more than one technique, for example, a combination of NLP and ML; NLP and Ontology had five studies each, accounting for 5.9% each. Other techniques applied were Information retrieval, Data mining, courteous logic, Tropos methodology, and Casco Accurate Description Language (CASDL). These categories supported different levels of abstraction (in terms of techniques, methodology, and logic). In addition, Table 7 shows the specific technologies used to develop the proposed tools. Accordingly, POS tagging, Parsers are the most widely used NLP technologies in tool development. While Naïve Bayes, support vector machines and neural networks are the most frequently used machine learning techniques for classifying and clustering.

4.6 How are these tools evaluated?

This question aimed to identify the evaluation approach(s) employed in evaluating the automated RE tools. Table 8 shows the results categorized by evaluating method and RE phase.

Automated RE tools play a crucial role in analysing and managing requirements. Thus, evaluating these tools is essential to ensure their effectiveness, suitability, and value for the intended purpose. Automated RE tool evaluation involves assessing various aspects of the tool with several factors

Table 7 Technologies employed in the development of tools support

Techniques/technologies	Studies
Natural language processing (NLP)	
POS Tagging	[S7], [S12], [S14], [S19], [S24], [S25], [S27], [S28], [S38], [S44], [S53],[S56], [S59], [S63], [S71], [S73], [S75], [S77], [S82], [S85], [S2], [S8], [S54], [S16], [S79], [S84], [S47], [S51], [S52], [S50], [S21], [S26], [S61], [S10]
Tokenization	[S56], [S59], [S63], [S71], [S77], [S82], [S85], [S54], [S16], [S79], [S84], [S51], [S52], [S39]
Lemmatization	[S78], [S19], [S25], [S56], [S57], [S59], [S16], [S8], [S71]
Stemming	[S78], [S59], [S82], [S16], [S14]
Segmentation	[S56], [S16]
NLP parser/stanford parser	[S7], [S11], [S14], [S15], [S60], [S64], [S70], [S73], [S74], [S75], [S76], [S82], [S85], [S2], [S54], [S21], [S26], [S53], [S59], [S65], [S82]
Parse tree	[S12], [S19], [S38], [S68], [S74], [S2]
Sentence splitting	[S63], [S7], [S77], [S16]
Morphological analysis	[S71], [S77], [S54]
Machine learning (ML)	
Classification	
Naïve bayes (NB)	[S1], [S13], [S36], [S62], [S66], [S69], [S29]
Nearest neighbor (KNN)	[S13], [S36], [S66]
Support vector machine (SVM)	[S58], [S13], [S66], [S69]
Clustering	
Neural networks	[S48], [S54], [S79], [S84]
Regression	
Decision tree	[S35], [S66]
Data mining	
Data mining	[S78], [S37], [S55]
Text mining	[S39], [S10], [S22], [S2]
Information retrieval (IR)	[S20], [S34], [S32]
Ontology	
Ontology (Graph)	[S21], [S40], [S49], [S67], [S42], [S26], [S61], [S51], [S52], [S50], [S23]

Table 8 Evaluation methods

Type	Controlled Experiment	Case study	Prototype/proof of concept	User study	Total
Elicitation	16	3	1	2	22 (25.9%)
Analysis	18	15	9	3	45 (52.9%)
Validation	2	3	1	2	8 (9.4%)
Management	5	4	1	0	10 (11.8%)
Total	41 (48.2%)	25 (29.4%)	12 (14.1%)	7 (8.2%)	85

considered, including functionality, usability, scalability, performance, integration capabilities, support for industry standards, and compatibility with existing tools and processes. To achieve this, several evaluation methods are employed, as reported by the primary studies. The primary studies employ the experimental method, case study evaluation, prototyping/proof of concept and user study. The choice of evaluation method is aligned with the specific objectives of the evaluation. It provides valuable insights into the tool's performance, usability, availability of resources and overall suitability for the intended use.

Most of the primary studies (41) applied a controlled experimental method. This indicates that these studies conducted experiments to evaluate and demonstrate the effectiveness of their approaches. The extent of the experiments varied across studies. Some authors had a single experiment with independent examples or data to explain their proposed method (e.g., studies referenced as [64, 66, 86]). Other studies used numerous in-depth examples based on data from industrial practice, providing a more comprehensive evaluation (e.g., studies referenced as [28, 75, 76]). The case study evaluation method was employed by 25 studies, representing 29.4% of the selected primary studies. Case studies are often conducted on practical live projects or industry applications to assess the proposed approaches' feasibility, effectiveness, and real-world applicability. These studies typically involve in-depth investigations of specific cases or scenarios to provide a more contextualised understanding of the benefits and limitations of the developed tools. An example of industry evaluation using case study evaluation is the study by Arora et al. [7], which conducted an evaluation using four industry requirements documents combined with expert evaluations to extract domain models from textual requirements. The results from the evaluation show that the potential of automated tools in modelling is enormous because their tool achieved approximately 90% accuracy in generating domain model classes. Prototyping, specifically for proof-of-concept purposes, was used in 12 studies. Prototyping involves building a preliminary version or prototype of the proposed system or tool to showcase its functionality, feasibility, and potential applications. These prototypes demonstrate how the methods can be applied in practice. The last evaluation method, user study, was employed by seven studies representing 8.2%. User studies involve gathering feedback and insights directly from end-users to evaluate the proposed tools' applicability, usability, and user experience. Most of the authors evaluated the performance of their artefacts with that of a human expert and compared them against defined/specified functionalities.

5 Discussion

As stated previously, the purpose of this study was to investigate automated RE support and bring to light more discussion supported by extant literature. Eighty-five papers relating to automated RE support were selected for this study. This objective was accomplished through the formulation and answer to research questions. The implications of the findings from Sect. 4 are therefore discussed in this section.

On a general note, it was observed that publications related to automated RE date to 1995, and the state of research has, over the years, improved significantly. The observed improvement in publications on automated RE can be attributed to several factors. Technological advancements in artificial intelligence and machine learning have provided new tools and techniques for automated RE. The increased recognition of the importance of RE in software development has led to more research and development efforts in the field, with a focus on automation. The limitations of traditional manual RE methods have driven the exploration of automated solutions for improved efficiency, accuracy, and scalability. On the other hand, we found that most of the publications focused on functional requirements. This type of requirement is concerned with the functional operations of the system, defining system functionalities and constraints. It is important to note that agile teams in the requirements engineering process are primarily concerned with this type of requirement. Another important aspect of our findings is the geographical location of the authors of our selected studies, as most of the papers were authored by Europeans and Asians. Most of the European contributions came from authors from the United Kingdom, the Netherlands, Germany, Sweden, Norway, Italy, Luxembourg, Spain, Austria, Ireland, Serbia, Switzerland, and Portugal. Asian countries took second place in the contribution,

involving authors from China, Malaysia, India, Palestine, South Korea, Pakistan, Jordan, Singapore, Japan, Israel, Turkey, and UAE. North American countries came third, involving only US, Canada, and Mexico authors. Finally, the remaining continents, Africa, Australia, Oceania, and South America, have contributions from one country, involving Morocco, Australia, New Zealand, and Argentina, respectively. The empirical data from the selected 85 studies could not be generalized due to the uneven distribution of authors across geographic regions. Therefore, there is a need for location-based studies like that of [87] to empirically establish state-of-the-art research in most continents regarding automated RE and software engineering generally. Geographical, psychological, and sociological factors may influence how anything is interpreted and understood when described in natural language [88].

Concerning RQ1, we observed that the class diagram was the most widely generated model. This could result from the class diagram representing high-level models that are useful during the requirements analysis and the initial stages of the system design. Class diagrams are also the most-used UML model in software development in general. This model type structurally captures the domain problem by representing them as classes, attributes, relationships, and association cardinalities. Overall, a class diagram was also used for stakeholder discussions, providing bases for discussion among the development team and providing agile team members with a basic understanding of the system's overall structure. Thus, a class diagram is easy to understand but consumes significant time to construct. Nevertheless, automated tools will go a long way in reducing the time spent generating class diagrams. In addition to class diagrams, use case models, structured requirements documents, and entity-relationship diagrams have also been generated as output from various tools. Automated tool support for automatically constructing an analysis model is critical to model-driven development (MDD) because it enables the MDD lifecycle to move quickly from the coding phase to the specification phase when using precise formal languages [14]. Hence, using automated RE tools to generate class diagrams and other UML diagrams has a promising future in the software RE domain and software development in general. Furthermore, the relationship between the Automated RE-generated UML output and the input is one of translation, interpretation, and representation. The UML output reflects the information and structure present in the input data but in a more organised, standardised, and visual form that facilitates analysis, communication, and further development activities. Therefore, input representation and standardisation, integration and interoperability, ambiguity handling, and support for various input sources are issues related to tool input that might contribute to developing more effective input for automated RE tools. Overall, the research implications surrounding the additional values of Automated RE-generated outputs, especially UML outputs, focus on automated RE technique evaluation, integration with development processes, automation of UML model evolution, context-specific adaptations, human aspects, and performance scalability. Addressing these research implications will allow the area of automated RE to progress, providing more robust and effective methods for automating the production of UML models in the RE process.

The primary RE task is requirements analysis and validation, which includes understanding user requirements, classifying them, and modelling static and dynamic perspectives of software requirements [89]. We have identified the most automated RE phase in response to RQ2: requirement analysis. According to empirical evidence, requirements analysis is the most widely automated RE phase. This could be because the phase allows for analysing the requirements outlined. Alternatively, it also allows for stakeholder deliberations. One of the most essential activities in the RE process is requirement analysis. This activity assists developers and other relevant stakeholders in understanding requirements, their overlaps, and conflicts, as well as in managing conflict and creating a unified set of requirements. Usually, the "Requirement analysis" phase of any software development process is often the most important because it serves as the foundation for all subsequent development work [48]. It is also important to note that, at this stage, various UML models are conceived, and the entire system functionalities are defined and analysed. Thus, this calls for more practical research into the automation of this stage to minimise human intervention to the barest minimum. One of the most significant research issues nowadays is automated software engineering, particularly when analysing requirements and modelling [5]. Thus, extracting requirements artefacts in the analysis stage requires a great deal of skill, and this has been the motivating factor for some of the primary studies to automate the analysis phase. For example, in [48], it was found that before modelling a system, a developer must analyse the use case descriptions to identify actors and functionalities to understand the system's dynamics. Similarly, extracting actions and actors from a linguistic viewpoint requires skill and time and is challenging to master [78]. More specifically, the motivation for the automation of the analysis phase is: saving time and reducing development costs [5, 13, 48, 90], leveraging the powers of NLP tools and technologies [14, 18, 19, 80, 91], avoiding or minimising human mistake in reading and analysing volumes of text written in NL [79, 89, 92] and iterative nature of the analysis phase [48]. Therefore, we propose that future research consider an automated analysis framework that will fit with agile development. Furthermore, the automated analysis that will allow for traceability between the textual requirements and

the formalised models be developed. This way, developers and customers can easily see which original statements from which some formalised requirements are derived. Thus, investigating the research implications of the automated analysis phase in automated RE can help develop the field by providing insights into the effectiveness, accuracy, and consistency of automated RE tools and their adoption, impact, and improvement. It can direct future research and bridge the gap between academic research and practical application in industry.

Accordingly, we have identified the extent of automation of the RE support tools in response to RQ3. As mentioned earlier, most of the tools had semi-automation, which should allow a reasonable part of the activities of a human analyst to be done automatically. Ironically, this does not in any way replace human expertise. Whether complete or semi-complete automation, there is a need for human expert validation since it is the human that will use the output for the remaining phases of development. The automation has proved to be effective in reducing the workload on the human analyst to manually carried out the RE activities such as model generation [77, 80, 81], inconsistencies in requirement [66, 67], checking duplicate requirements [75] and omissions in requirement [16]. Remarkably, there is still the need for researchers in the field to provide more explicit and specific descriptions of the level of automation that will significantly reduce human efforts and development time. This could include information on the automated tasks, processes, or functionalities covered and the extent to which human intervention or manual effort is required. Another important aspect of automation is the human-in-the-loop considerations. At the same time, semi-automation implies a level of human involvement in the RE process, while full automation suggests minimal or no human intervention. Therefore, research can delve into the role of humans in the loop and investigate their contribution, decision-making authority, and interaction with automated tools. This can provide insights into the optimal balance between automation and human expertise, ensuring that automated RE tools effectively leverage human intelligence while reducing manual effort.

We identified Natural Language Processing techniques as the most used in developing various RE automated tools in response to RQ4. As mentioned earlier, NLP plays a significant role in RE activities because most requirements are written in Natural languages. Some natural language techniques employed by the primary studies include linguistic analysis [93, 94], clustering algorithm [16, 95], pattern matching [38], core NLP [78, 94], NL Pasrser [9, 28, 47, 68, 76, 79, 88, 90, 95], text chunking [24, 94], Part of Speech (POS) Tagger [24, 80, 94] to mention but a few. These techniques have been applied in various RE phases, elicitation, analysis, specification, management, and validation. The NLP helps in the text analysis and context meaning to allow for understanding text for onward requirements analysis. Generally, requirements engineering is considered a critical success factor in software development, and most requirements are written in NL, which is challenging to analyse computationally. It has been noted that the need for adequate techniques and tools for computer-assisted processing of early software requirements is one of the major problems in software development [86]. To address these challenges, NLP techniques have been employed. The NLP techniques allow text written in NL to be analysed through major components; lexical analysis (pre-processing, tokenisation, morphology rules, lexicon), syntactic analysis (parsing and mapping), and semantic analysis (drawing, checking, including) [78]. Therefore, pre-processing, classification, and post-processing (mapping relationships) are common processes used in generating most of the analysis models in the literature. In addition to the NLP techniques, machine learning techniques and their application in automated requirements engineering are evolving with many potentials. The proliferation of digitisation has resulted in vast data that allows machine processing. Machine Learning has been employed in some of the primary studies of tasks related to classification [39, 40, 48, 96] and clustering [78, 97]. The combination of NLP and ML techniques has also proven more effective in automated RE support. Since both technologies perform different tasks, while NLP is used mostly for text pre-processing (lexical analysis), machine learning is used for classification and clustering. For example, the outputs of tools like *ReqAligner* [75], which assists analysts in identifying duplicate functionality in textual use cases, and *DoMoBOT*[13], an automated tool for interactive domain modelling, have aided the development of many automated tools. Like other tools reported in the primary studies, these have proven the effective use of NLP and ML techniques in automated RE support. However, the increasing use of NLP and ML approaches in developing automated RE tools has various research implications, including algorithm selection, performance evaluation, data requirements, transfer learning, privacy, human-AI collaboration, and bias considerations. As a result of addressing these implications, researchers can advance the field of automated RE and enhance the capabilities, reliability, and ethical soundness of automated requirements engineering processes.

With respect to RQ5 on how these tools are evaluated, we identified the controlled experimental evaluation method to be the most widely used by authors. The experiment is one of the most widely used evaluation methods in research. In tool development, an experiment has proven effective in evaluating tools, especially at the early stage before industrial case study evaluation. It helps reveal lags and uncovers errors and shortcomings of a tool before subjecting it to an industrial case study. During the evaluation, most of the proposed artefacts are evaluated against the expected functionalities

or the manually generated analysis models. For instance, [13] evaluated the automatically generated class diagram against models manually generated from various problem specifications commonly used in universities. Experiments are also cost-effective, and since most authors are academics, their colleagues and students usually constitute part of the experiment in testing or evaluating such tools with minimal budgets. The fact that experimental evaluation is the most widely used approach highlights its significance in assessing the effectiveness and performance of automated requirements engineering tools. Researchers and practitioners should continue to prioritize rigorous experimental evaluations to provide empirical evidence and quantitative data supporting the claims and benefits of these tools. While case studies and prototypes represent a smaller percentage of evaluation methods, their usage indicates their value in providing deeper insights into the practical application and real-world effectiveness of automated requirements engineering tools. This finding suggests that further exploring case studies and prototypes can contribute to a more comprehensive understanding of tool performance in specific contexts or scenarios. The lower usage of user studies as an evaluation method suggests a need for more attention to the user's perspective. User studies can provide valuable insights into user experiences, satisfaction, and usability aspects of automated requirements engineering tools. The distribution of evaluation methods indicates the importance of adopting a balanced approach that includes a mix of evaluation methods. Combining experimental evaluations, case studies, prototypes, and user studies can provide a more holistic understanding of automated requirements engineering tools' strengths, limitations, and practical implications. In terms of comparative evaluation of different tools against each other, this has not been done, partly because there are no existing established requirements cases on which such comparisons could be performed.

6 Conclusion and future work

6.1 Limitation of the review

The limitations of any systematic review are multifaceted, encompassing potential subjective selection of studies for inclusion. This can arise if the selection process needs more transparency or if personal biases influence the decision-making. Limiting the search to specific databases can also be a limitation, as it may result in incomplete coverage of the literature. A flawed protocol can impact the reliability and validity of the review and introduce bias to the findings. To mitigate these limitations, the guidelines provided in [60] were employed to minimize biases in study selection. However, it is worth noting that during the search process, there is always a chance of overlooking relevant studies, as highlighted by [62]. Moreover, the influence of language on search strings in software engineering further complicates the search for studies. To enhance data gathering accuracy and minimize internal validity threats, an iterative selection process was employed. This approach aligns with the rigour and thoroughness principles underpinning the systematic literature review methodology. By employing this iterative process, we can provide a comprehensive and reliable synthesis of the existing evidence on the chosen topic, making the review a valuable resource for decision-making and further research. By establishing specific inclusion and exclusion criteria, search strategy, data extraction methods, and quality assessment criteria, the selection preference was minimized (at least to some extent), focusing on obtaining the most relevant studies. This approach aimed to produce more generalizable results for the systematic literature review, alleviating potential external validity threats.

6.2 Conclusion and future work

This paper presents a systematic literature review on automated requirements engineering. The review was conducted by searching for and classifying all available studies on automated requirements engineering, following established guidelines for conducting a systematic literature review. Eighty-five papers were selected, reviewed, and systematically analysed based on five specified research questions. These questions aimed to explore the state-of-the-art research in automated requirements engineering, focusing on tool output, automation of requirements engineering phases, level/degree of automation, development approaches/techniques, and tool evaluation methods. As a result, the following responses to the research questions were obtained:

- Researchers increasingly focus on automated requirements engineering, resulting in many publications and various outputs. The most significant output supported by automated tools is UML models (44.7%), while other outputs are primarily related to activities/tasks, such as omission of steps, consistency checking, and requirement validation.
- Most automated tools (49.53) are developed to automate activities in the analysis phase of requirements engineering, making it the most widely automated phase.
- Results indicate that 59% of the tools are semi-automated, requiring some human intervention to perform specified tasks.
- Natural language processing technologies are widely employed in developing automated requirements engineering tools, accounting for 50.6% of the total studies, with POS tagging and Parsers being the most used NLP technologies.

- Controlled experimental methods are most frequently used (48.2%) to evaluate automated requirements engineering tools, while user studies are the least employed evaluation method (8.2%).

Over the past two decades, significant research progress has been made in the academic space regarding automated requirements engineering. However, there is limited application and evaluation of results in the industry, which would provide valuable real-world feedback. This is mainly due to the experimental stage of automated requirements engineering applications. Another notable finding is the need for comparative evaluation analysis among these tools, which may be attributed to the absence of a common dataset. In conclusion, this systematic review endeavours to aggregate empirical findings from publications on automated requirements engineering, shedding light on current practices. These practices are expected to guide researchers and practitioners in applying and evaluating support tools designed to assist requirements engineering processes/activities. Furthermore, this review introduces a new paradigm in requirements engineering, leveraging artificial intelligence and related techniques to facilitate data-driven requirements engineering.

Our future study will determine the extent of traceability among the produced artefacts. There is a need to create a traceability mechanism to maintain links between requirements elements and the created artefacts. Additionally, we will conduct an interview-based study to explore automated requirements engineering practices by industry practitioners to complement the findings of the systematic literature review. On the other hand, there is an increasing potential for using NLP technologies for automated requirements analysis due to the recent advances in artificial intelligence, which enable robust context-sensitive analysis of natural language texts. Therefore, it is also essential to empirically investigate deeper to facilitate a more comprehensive understanding of an optimal process for automated requirements analysis, for example, identifying the data concepts (classes) and features/attributes in the first stage and as a basis for further detailed analysis of functionality. Finally, there is equally the potential for studies on a comparative evaluation of different approaches on the exact requirements of case studies.

Appendix

See Table9

Table 9 Selected primary studies

Study ID	Authors	References	Title	Year	Publisher	Name	Citation type
S1	Vemuri et al	[48]	Automated Use Case Diagram Generation from Textual User Requirement Documents	2017	IEEE	IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)	Conference
S2	Ko et al	[16]	Automatic recommendation to omitted steps in use case specification	2019	Springer	Requirements Engineering	Journal
S3	Kamalrudin	[66]	Automated Software Tool Support for Checking the Inconsistency of Requirements	2009	IEEE	IEEE/ACM International Conference on Automated Software Engineering	Conference
S4	Pinquié et al	[63]	A requirement mining framework to support complex sub-systems supplier	2018	Elsevier	CIRP Design Conference (Procedia CIRP)	Conference
S5	Aysolmaz et al	[28]	A semi-automated approach for generating natural language requirements documents based on business process models	2018	Elsevier	Information and Software Technology	Journal
S6	Moketar et al	[68]	An Automated Collaborative Requirements Engineering Tool for Better Validation of Requirements	2016	ACM	ASE'16	Conference
S7	Deeptimahanti and Babar	[88]	An Automated Tool for Generating UML Models from Natural Language Requirements	2009	IEEE	IEEE/ACM International Conference on Automated Software Engineering	Conference
S8	Guzman et al	[38]	Automated Requirements Extraction for Scientific Software	2015	Elsevier	International Conference on Computational Science (ICCS)	Conference
S9	Parra et al	[70]	A methodology for the classification of quality of requirements using machine learning techniques	2015	Elsevier	Information and Software Technology	Journal
S10	Reddivari et al	[74]	Automated support to capture verbal just-in-time requirements via audio mining and cluster-based visualization	2019	Elsevier	Journal of Industrial Information Integration	Journal
S11	Thakur and Gupta	[79]	Automatic Generation of Sequence Diagram from Use Case Specification	2014	ACM	ISEC'14,	Conference
S12	Elallaoui et al	[80]	Automatic Transformation of User Stories into UML Use Case Diagrams using NLP Techniques	2018	Elsevier	The 8th International Conference on Ambient Systems, Network and Technologies (ANT)	Conference
S13	Li, and Huang et al	[40]	Automatically classifying user requests in crowdsourcing requirements engineering	2018	Elsevier	The Journal of Systems and Software	Journal
S14	Vidya Sagar and Abirami	[77]	Conceptual modeling of natural language functional requirements	2014	Elsevier	The Journal of Systems and Software	Journal
S15	Li and Yue et al	[76]	Enabling automated requirements reuse and configuration	2019	Springer	Software & Systems Modeling	Journal
S16	Rago et al	[75]	Identifying duplicate functionality in textual use cases by aligning semantic actions	2016	Springer	Software & Systems Modeling	Journal
S17	Antinyan and Staron	[69]	Rendex: A method for automated reviews of textual requirements	2017	Elsevier	The Journal of Systems and Software	Journal

Table 9 (continued)

Study ID	Authors	References	Title	Year	Publisher	Name	Citation type
S18	Moketar et al	[15]	TestMEReq: Generating Abstract Tests for Requirements Validation	2016	IEEE	3rd International Workshop on Software Engineering Research and Industrial Practice (SER&IP'16)	Conference
S19	Ambriola and Gervasi	[47]	On the Systematic Analysis of Natural Language Requirements with CIRCE	2006	Springer	Automated Software Engineering	Journal
S20	Cleland-Huang et al	[71]	Automated classification of non-functional requirements	2007	Springer	Requirements Engineering	Journal
S21	Vlas and Robinson	[41]	A Rule-Based Natural Language Technique for Requirements Discovery and Classification in Open-Source Software Development Projects	2011	IEEE	44th Hawaii International Conference on System Sciences	Conference
S22	Cybulski and Reed	[72]	Requirements Classification and Reuse: Crossing Domain Boundaries	2000	Springer	6th International Conference, ICSR-6	Conference
S23	Kiyavitskaya and Zannone	[85]	Requirements model generation to support requirements elicitation: The Secure Tropos experience	2008	Springer	Automated Software Engineering	Journal
S24	Sampaio et al	[98]	EA-Miner: Towards Automation in Aspect-Oriented Requirements Engineering	2007	Springer	Transactions on Aspect-Oriented Software Development III	Chapter
S25	Gacitua et al	[99]	Relevance-based abstraction identification: technique and evaluation	2011	Springer	Requirements Engineering	Journal
S26	Omoronyia et al	[100]	A Domain Ontology Building Process for Guiding Requirements Elicitation	2010	Springer	16th International Working Conference, REFSQ,	Chapter
S27	Ferrari and Esuli	[101]	An NLP approach for cross-domain ambiguity detection in requirements engineering	2019	Springer	Automated Software Engineering	Journal
S28	Alkhader et al	[64]	Experimenting with Extracting Software Requirements Using NLP Approach	2006	IEEE	ICIA, 2006	Conference
S29	Schneider et al	[39]	Enhancing security requirements engineering by organizational learning	2012	Springer	Requirements Engineering	Journal
S30	Seresht et al	[89]	Automatic Conceptual Analysis of User Requirements with the Requirements Engineering Assistance Diagnostic (READ) Tool	2008	IEEE	6th International Conference on Software Engineering Research, Management and Applications (SERA)	Conference
S31	Cybulski & Reed	[86]	Computer-Assisted Analysis and Refinement of Informal Software Requirements Documents	1998	IEEE	Asia Pacific Software Engineering Conference	Conference
S32	Natt och Dag et al	[102]	A Feasibility Study of Automated Natural Language Requirements Analysis in Market-Driven Development	2002	Springer	Requirements Engineering	Journal
S33	Natt och Dag and Thelin et al	[93]	An experiment on linguistic tool support for consolidation of requirements from multiple sources in market-driven product development	2006	Springer	Empirical Software Engineering	Journal
S34	Park et al	[103]	Implementation of an efficient requirements-analysis supporting system using similarity measure techniques	2000	Elsevier	Information and software technology	Journal

Table 9 (continued)

Study ID	Authors	References	Title	Year	Publisher	Name	Citation type
S35	Ankori	[65]	Automatic Requirements Elicitation in Agile Processes	2005	IEEE	IEEE International Conference on Software—Science, Technology & Engineering (SwSTE'05)	Conference
S36	Casamayor et al	[96]	Identification of non-functional requirements in textual specifications: A semi-supervised learning approach	2009	Elsevier	Information and software technology	Journal
S37	Castro-Herrera et al	[104]	A Recommender System for Requirements Elicitation in Large-Scale Software Projects	2009	ACM	SAC'09	Conference
S38	Lee and Bryant	[9]	Automation of Software System Development Using Natural Language Processing and Two-Level Grammar	2004	Springer	9th International Workshop, RISSEF 2002	Chapter
S39	Rago et al	[105]	Uncovering quality-attribute concerns in use case specifications via early aspect mining	2013	Springer	Requirements Engineering	Journal
S40	Shibaoka et al	[106]	GOORE: Goal-Oriented and Ontology Driven Requirements Elicitation Method	2007	Springer	ER 2007Workshops CMLSA, FP-UML, ONISW, QoIS, RIGiM, SeCoGIS	Chapter
S41	Pavlidis et al	[107]	A CASE Tool to Support Automated Modelling and Analysis of Security Requirements, Based on Secure Tropos	2012	Springer	CAiSE Forum 2011	Chapter
S42	Wang et al	[108]	A novel data-driven graph-based requirement elicitation framework in the smart product-service system context	2019	Elsevier	Advanced Engineering Informatics	Journal
S43	Sharma and Biswas	[67]	A Semi-automated Approach towards Handling Inconsistencies in Software Requirements	2013	Springer	ENASE 2012,	Chapter
S44	Robeer et al	[81]	Automated Extraction of Conceptual Models from User Stories via NLP	2016	IEEE	IEEE 24th International Requirements Engineering Conference	Conference
S45	Miao et al	[43]	Automated Requirements Validation for ATP Software via Specification Review and Testing	2016	Springer	ICFEM 2016	Chapter
S46	Yusop et al	[37]	Automated Support to Capture and Validate Security Requirements for Mobile Apps	2016	Springer	APRES 2016	Chapter
S47	Jin et al	[24]	Automated Support to Capture Creative Requirements via Requirements Reuse	2019	Springer	ICSR 2019	Chapter
S48	Winkler and Vogelsang	[73]	Automatic Classification of Requirements Based on Convolutional Neural Networks	2016	IEEE	IEEE 24th International Requirements Engineering Conference	Conference
S49	Moitra et al	[42]	Automating requirements analysis and test case generation	2019	Springer	Requirements Engineering	Journal
S50	Ibrahim & Ahmad	[90]	Class diagram extraction from textual requirements using Natural language processing (NLP) techniques	2010	IEEE	Second International Conference on Computer Research and Development	Conference
S51	Lucassen et al	[109]	Extracting conceptual models from user stories with Visual Narrator	2017	Springer	Requirements Engineering	Journal

Table 9 (continued)

Study ID	Authors	References	Title	Year	Publisher	Name	Citation type
S52	Reinhartz-Berger and Kemelman	[95]	Extracting core requirements for software product lines	2020	Springer	Requirements Engineering	Journal
S53	Lucassen and Dalpiaz et al	[94]	Improving agile requirements: The Quality User Story framework and tool	2016	Springer	Requirements Engineering	Journal
S54	Al-Hroob et al	[78]	The use of artificial neural networks for extracting actions and actors from requirements document	2018	Elsevier	Information and Software Technology	Journal
S55	Duan et al	[110]	Towards automated requirements prioritization and triage	2009	Springer	Requirements Engineering	Journal
S56	Ezzini et al	[44]	Using Domain-specific Corpora for Improved Handling of Ambiguity in Requirements	2021	IEEE	2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)	Conference
S57	Kuk et al	[111]	A Semi-automated generation of Entity-Relationship Diagram based on Morphosyntactic Tagging from the Requirements Written in a Serbian Natural Language	2019	IEEE	IEEE Joint 19th International Symposium on Computational Intelligence and Informatics	Conference
S58	Kurtanovic and Maalej	[112]	Automatically Classifying Functional and Non-Functional Requirements Using Supervised Machine Learning	2017	IEEE	2017 IEEE 25th International Requirements Engineering Conference	Conference
S59	Nassar and Khamayseh	[5]	Constructing Activity Diagrams from Arabic User Requirements using Natural Language Processing Tool	2015	IEEE	2015 6th International Conference on Information and Communication Systems (ICICS)	Conference
S60	Jabbarin and Arman	[113]	Constructing Use Case Models from Arabic User Requirements in a Semi-Automated Approach	2014	IEEE	2014 World Congress on Computer Applications and Information Systems (WCCAIS)	Conference
S61	Jyothilakshmi and Samuel	[6]	Domain Ontology Based Class Diagram Generation from Functional Requirements	2012	IEEE	2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)	Conference
S62	Sardinha et al	[114]	EA-Analyzer: automating conflict detection in a large set of textual aspect-oriented requirements	2013	IEEE	Automated Software Engineering	Journal
S63	Arora et al	[7]	Extracting Domain Models from Natural-Language Requirements: Approach and Industrial Evaluation	2016	ACM	ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems (MODELS '16)	Conference
S64	Sharma et al	[21]	From Natural Language Requirements to UML Class Diagrams	2015	IEEE	2015 IEEE Second International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)	Conference
S65	Alami et al	[115]	A Semi–automated Approach for Generating Sequence Diagrams from Arabic User Requirements Using a Natural Language Processing Tool	2017	IEEE	2017 8th International Conference on Information Technology (ICIT)	Conference
S66	Haque et al	[45]	Non-Functional Requirements Classification with Feature Extraction and Machine Learning: An Empirical Study	2019	IEEE	1st International Conference on Advances in Science, Engineering and Robotics Technology 2019 (ICASERT 2019)	Conference

Table 9 (continued)

Study ID	Authors	References	Title	Year	Publisher	Name	Citation type
S67	Nguyen et al	[83]	REInDetector: A Framework for Knowledge-Based Requirements Engineering	2012	IEEE	2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering	Conference
S68	Yue et al	[19]	aToucan: An Automated Framework to Derive UML Analysis Models from Use Case Models	2015	ACM	ACM Transactions on Software Engineering and Methodology	Journal
S69	Ott	[116]	Automatic Requirement Categorization of Large Natural Language Specifications at Mercedes-Benz for Review Improvements	2013	Springer	REFSQ 2013	Chapter
S70	Mu et al	[91]	Extracting Software Functional Requirements from Free Text Documents	2009	IEEE	2009 International Conference on Information and Multimedia Technology	Conference
S71	Umber et al	[46]	NL-Based Automated Software Requirements Elicitation and Specification	2011	Springer	ACC 2011	Chapter
S72	Elbendak et al	[117]	Parsed use case descriptions as a basis for object-oriented class model generation	2011	Elsevier	The Journal of Systems and Software	Journal
S73	Nguyen and Grundy et al	[118]	Rule-Based Extraction of Goal-Use Case Models from Text	2015	ACM	ESEC/FSE'15	Conference
S74	Deeptimahanti and Sanyal	[92]	Semi-automatic Generation of UML Models from Natural Language Requirements	2015	ACM	ISEC '11	Conference
S75	Letsholo et al	[14]	TRAM: A Tool for Transforming Textual Requirements into Analysis Models	2013	IEEE	ASE 2013	Conference
S76	Segundo et al	[119]	UML Sequence Diagram Generator System from Use Case Description Using Natural Language	2007	IEEE	Fourth Congress of Electronics, Robotics and Automotive Mechanics	Conference
S77	Harmain and Gaizauskas	[18]	CM-Builder: A Natural Language-Based CASE Tool for Object-Oriented Analysis	2003	Springer	Automated Software Engineering	Journal
S78	Qureshi et al	[120]	Capturing Users Requirements Using a Data Mining Approach	2021	IEEE	International Conference on Communication Technologies (ComTech)	Conference
S79	Saini et al	[13]	DoMoBOT: A Bot for Automated and Interactive Domain Modelling	2020	ACM	International Conference on Model Driven Engineering Languages and Systems (MODELS'20)	Conference
S80	Kamalrudin et al	[121]	MaramaAIC: tool support for consistency management and validation of requirements	2017	Springer	Automated Software Engineering	Journal
S81	Panichella and Ruiz	[20]	Requirements-Collector: Automating Requirements Specification from Elicitation Sessions and User Feedback	2020	IEEE	2020 IEEE 28th International Requirements Engineering Conference (RE)	Conference
S82	Shehadeh et al	[122]	Semi-Automated Classification of Arabic User Requirements into Functional and Non-Functional Requirements using NLP Tools	2021	IEEE	2021 International Conference on Information Technology (ICIT)	Conference

Table 9 (continued)

Study ID	Authors	References	Title	Year	Publisher	Name	Citation type
S83	Mustaffa et al	[123]	Semi-Automated Software Requirements Specification (SRS) Document Generator: The Guide to Novice Analyst	2021	IEEE	2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information management (ICSECS-ICOCSIM)	Conference
S84	Saini et al	[124]	Automated, interactive, and traceable domain modelling empowered by artificial intelligence	2022	Springer	Software and Systems Modeling	Journal
S85	Nasiri et al	[82]	Towards a Generation of Class Diagram from User Stories in Agile Methods	2020	Elsevier	International Workshops on the Advancements in Model Driven Engineering (AMDE 2020)	Conference

Acknowledgements Muhammad Aminu Umar acknowledges the funding support of the Petroleum Technology Development Fund (PTDF), the Federal Government of Nigeria.

References

1. Zava P (1995) Classification of research efforts in requirements engineering. In: Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95)
2. Nuseibeh B, Easterbrook S (2000) Requirements engineering: a roadmap. In: Proceedings of the conference on the future of software engineering (ICSE '00), pp 35–46. [Online]. https://doi.org/10.1145/336512.336523
3. Geogy M, Dharani A (2016) A scrutiny of the software requirement engineering process. Procedia Technol 25:405–410. https://doi.org/10.1016/j.protcy.2016.08.125
4. Mich L, Franch M, Novi IP (2004) Market research for requirements analysis using linguistic tools. Requirements Eng 9:40–56
5. Nassar IN, Khamayseh FT (2015) Constructing activity diagrams from Arabic user requirements using Natural Language Processing tool. In: 2015 6th International conference on information and communication systems (ICICS), Amman, Jordan: IEEE, pp 50–54. https://doi.org/10.1109/IACS.2015.7103200
6. Jyothilakshmi MS, Samuel P (2012) Domain ontology based class diagram generation from functional requirements. In: 2012 12th International conference on intelligent systems design and applications (ISDA), Kochi, India: IEEE, pp 380–385. https://doi.org/10.1109/ISDA.2012.6416568
7. Arora C, Sabetzadeh M, Briand L, Zimmer F (2016) Extracting domain models from natural-language requirements: approach and industrial evaluation. In: Proceedings of the ACM/IEEE 19th International conference on model driven engineering languages and systems, Saint-malo France: ACM, pp 250–260. https://doi.org/10.1145/2976767.2976769
8. Ryan K (1993) The role of natural language in requirements engineering. In: IEEE International symposium on requirements engineering, IEEE, pp 240–242. https://doi.org/10.1109/ISRE.1993.324852
9. Lee BS, Bryant BR (2022) Automation of Software System Development Using Natural Language Processing and Two-Level Grammar. In: Radical Innovations of Software and Systems Engineering in the Future. RISSEF 2002, Springer, Berlin, Heidelberg, pp 219–233
10. Flores JJG (2004) Linguistic processing of natural language requirements: the contextual exploration approach. In: Proceedings of the 10th international workshop on requirements engineering: foundation for software quality (REFSQ'04), pp 99–112
11. Nazir F, Butt WH, Anwar MW, Khattak MAK (2017) The applications of natural language processing (NLP) for software requirement engineering: a systematic literature review. In:

International conference on information science and applications, pp 485–493
12. Teichroew D, Sayani H (1980) Computer-Aided Requirements Engineering. In: ACM National Conference Proceedings (ACM '80), ACM, pp 369–381
13. Saini R, Mussbacher G, Guo JLC, Kienzle J (2020) DoMoBOT: a bot for automated and interactive domain modelling. In: Proceedings of the 23rd ACM/IEEE international conference on model driven engineering languages and systems: companion proceedings, Virtual Event Canada: ACM, pp 1–10. https://doi.org/10.1145/3417990.3421385
14. Letsholo KJ, Zhao L, Chioasca EV (2013) TRAM: A tool for transforming textual requirements into analysis models. In: 2013 28th IEEE/ACM International conference on automated software engineering (ASE), Silicon Valley, CA, USA: IEEE, pp 738–741. https://doi.org/10.1109/ASE.2013.6693146
15. Moketar NA, Kamalrudin M, Sidek S, Robinson M, Grundy J (2016) TestMEReq: generating abstract tests for requirements validation. In: Proceedings of the 3rd international workshop on software engineering research and industrial practice - SER&IP '16, Austin, Texas: ACM Press, pp 39–45. https://doi.org/10.1145/2897022.2897031
16. Ko D, Kim S, Park S (2019) Automatic recommendation to omitted steps in use case specification. Requirements Eng 24(4):431–458. https://doi.org/10.1007/s00766-018-0288-z
17. Kumar DD, Sanyal R (2008) Static UML Model Generator from Analysis of Requirements (SUGAR). In: 2008 Advanced Software Engineering and Its Applications, Hainan, China: IEEE, pp 77–84. https://doi.org/10.1109/ASEA.2008.25
18. Harmain HM, Gaizauskas R (2003) CM-builder: a natural language-based CASE tool for object-oriented analysis. Autom Softw Eng 10:157–181. https://doi.org/10.1023/A:10229160289
19. Yue T, Briand LC, Labiche Y (2015) aToucan: an automated framework to derive UML analysis models from use case models. ACM Trans Softw Eng Methodol 24(3):1–52. https://doi.org/10.1145/2699697
20. Panichella S Ruiz M (2020) Requirements-Collector: Automating Requirements Specification from Elicitation Sessions and User Feedback. In: 2020 IEEE 28th International requirements engineering conference (RE), Zurich, Switzerland: IEEE, pp 404–407. https://doi.org/10.1109/RE48521.2020.00057
21. Sharma R, Srivastava PK, Biswas KK (2015) From natural language requirements to UML class diagrams. In: 2015 IEEE Second international workshop on artificial intelligence for requirements engineering (AIRE), Ottawa, ON: IEEE, pp 1–8. https://doi.org/10.1109/AIRE.2015.7337625
22. Pohl K (2010) Requirements engineering: fundamentals, principles, and techniques, 1st edn. Springer, Berlin
23. Sommerville I (2011) Software Engineering, 9th ed. Person Education, Inc.
24. Do QA, Chekuri SR, Bhowmik T (2019) Automated Support to Capture Creative Requirements via Requirements Reuse. In: Reuse in the Big Data Era, Peng X, Ampatzoglou A, Bhowmik T, Eds., in Lecture Notes in Computer Science, Springer, Cham, pp 47–63. https://doi.org/10.1007/978-3-030-22888-0_4
25. Curcio K, Navarro T, Malucelli A, Reinehr S (2018) Requirements engineering: a systematic mapping study in agile software development. J Syst Softw 139:32–50
26. Jin Z (2018) Requirements and requirements engineering. Environ Model-Based Requir Eng Softw Intensive Syst. https://doi.org/10.1016/B978-0-12-801954-2.00001-7
27. De Lucia A, Qusef A (2010) Requirements engineering in agile software development. J Emerg Techn Web Intell 2(3):212–220
28. Aysolmaz B, Leopold H, Reijers HA, Demirörs O (2018) A semi-automated approach for generating natural language requirements documents based on business process models. Inf Softw Technol 93:14–29. https://doi.org/10.1016/j.infsof.2017.08.009
29. Mehmood MA, Khan MNA, Afzal W (2018) Automating Test Data Generation for Testing Context-Aware Applications. In: 2018 IEEE 9th International conference on software engineering and service science (ICSESS), Beijing, China: IEEE, pp 104–108. https://doi.org/10.1109/ICSESS.2018.8663920
30. Turner DA, Park M, Kim J, Chae J (2008) An Automated Test Code Generation Method for Web Applications using Activity Oriented Approach. In: 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, L'Aquila, Italy: IEEE, pp 411–414. https://doi.org/10.1109/ASE.2008.61
31. Anil Kumar S (2020) Enhancing the Scope for Automated Code Generation and Parallelism by Optimizing Loops through Loop Unrolling. In: 2020 Fourth International Conference on Inventive Systems and Control (ICISC), Coimbatore, India: IEEE, pp 790–795. https://doi.org/10.1109/ICISC47916.2020.9171081
32. Nakatoh T, Uchida S, Ishita E, Oga T (2016) Automated Generation of Coding Rules: Text-Mining Approach to ISO 26000. In: 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), Kumamoto, Japan: IEEE, pp 154–158. https://doi.org/10.1109/IIAI-AAI.2016.210
33. Xie S, Yang J, Lu S (2021) Automated Code Refactoring upon Database-Schema Changes in Web Applications. In: 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), Melbourne, Australia: IEEE, pp 1262–1265. https://doi.org/10.1109/ASE51524.2021.9678934
34. Musthafa FN, Mansur S, Wibawanto A (2020) Automated Software Testing on Mobile Applications: A Review with Special Focus on Android Platform. In: 2020 20th International Conference on Advances in ICT for Emerging Regions (ICTer), Colombo, Sri Lanka: IEEE, pp 292–293. https://doi.org/10.1109/ICTer51097.2020.9325445
35. Aguirre N (2017) Efficient SAT-Based Software Analysis: From Automated Testing to Automated Verification and Repair. In: 2017 IEEE/ACM 5th International FME Workshop on Formal Methods in Software Engineering (FormaliSE), Buenos Aires, Argentina: IEEE, pp 2–2. https://doi.org/10.1109/FormaliSE.2017.21
36. Wirotyakun A, Netisopakul P (2012) Improving software maintenance size metrics A case study: Automated report generation system for particle monitoring in Hard Disk Drive Industry. In: 2012 Ninth International Conference on Computer Science and Software Engineering (JCSSE), Bangkok, Thailand: IEEE, pp 334–339. https://doi.org/10.1109/JCSSE.2012.6261975
37. Yusop N, Kamalrudin M, Sidek S, Grundy J (2016) Automated Support to Capture and Validate Security Requirements for Mobile Apps. In: Requirements Engineering Toward Sustainable World, vol. 671, S.-W. Lee and T. Nakatani, Eds., in Communications in Computer and Information Science, Singapore: Springer Singapore, pp 97–112. https://doi.org/10.1007/978-981-10-3256-1_7
38. Li Y, Guzman E, Tsiamoura K, Schneider F, Bruegge B (2015) Automated requirements extraction for scientific software. Procedia Comput Sci 51:582–591. https://doi.org/10.1016/j.procs.2015.05.326
39. Schneider K, Knauss E, Houmb S, Islam S, Jürjens J (2012) Enhancing security requirements engineering by organizational learning. Requirements Eng 17(1):35–56. https://doi.org/10.1007/s00766-011-0141-0
40. Li C, Huang L, Ge J, Luo B, Ng V (2018) Automatically classifying user requests in crowdsourcing requirements engineering. J Syst Softw 138:108–123. https://doi.org/10.1016/j.jss.2017.12.028
41. Vlas R, Robinson WN (2011) A Rule-Based Natural Language Technique for Requirements Discovery and Classification in

Open-Source Software Development Projects. In: 2011 44th Hawaii International Conference on System Sciences, Kauai, HI: IEEE, pp 1–10. https://doi.org/10.1109/HICSS.2011.28
42. Moitra A et al (2019) Automating requirements analysis and test case generation. Requirements Eng 24(3):341–364. https://doi.org/10.1007/s00766-019-00316-x
43. Miao W et al. (2016) Automated Requirements Validation for ATP Software via Specification Review and Testing. In: Formal Methods and Software Engineering, K. Ogata, M. Lawford, and S. Liu, Eds., in Lecture Notes in Computer Science, Cham: Springer International Publishing, pp 26–40. https://doi.org/10.1007/978-3-319-47846-3_3
44. Ezzini S, Abualhaija S, Arora C, Sabetzadeh M, Briand LC (2021) Using Domain-Specific Corpora for Improved Handling of Ambiguity in Requirements. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), Madrid, ES: IEEE, pp 1485–1497. https://doi.org/10.1109/ICSE43902.2021.00133
45. Md Haque A, Abdur Rahman Md, Siddik MS (2019) Non-Functional Requirements Classification with Feature Extraction and Machine Learning: An Empirical Study. In: 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), Dhaka, Bangladesh: IEEE, pp 1–5. https://doi.org/10.1109/ICASERT.2019.8934499
46. Umber A, Bajwa IS, Asif Naeem M (2011) NL-Based Automated Software Requirements Elicitation and Specification. In: Advances in Computing and Communications, A. Abraham, J. Lloret Mauri, J. F. Buford, J. Suzuki, and S. M. Thampi, Eds., in Communications in Computer and Information Science, Berlin, Heidelberg: Springer Berlin Heidelberg, pp 30–39. https://doi.org/10.1007/978-3-642-22714-1_4
47. Ambriola V, Gervasi V (2006) On the systematic analysis of natural language requirements with CIRCE. Autom Software Eng 13(1):107–167. https://doi.org/10.1007/s10515-006-5468-2
48. Vemuri S, Chala S, Fathi M (2017) Automated use case diagram generation from textual user requirement documents. In: 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), Windsor, ON: IEEE, pp 1–4. https://doi.org/10.1109/CCECE.2017.7946792
49. Yue T, Briand LC, Labiche Y (2011) A systematic review of transformation approaches between user requirements and analysis models. Requirements Eng 16(2):75–99. https://doi.org/10.1007/s00766-010-0111-y
50. de Carrillo Gea JM, Nicolás J, Fernández Alemán JL, Toval A, Ebert C, Vizcaíno A (2012) Requirements engineering tools: capabilities, survey and assessment. Inf Softw Technol 54(10):1142–1157. https://doi.org/10.1016/j.infsof.2012.04.005
51. Meth H, Brhel M, Maedche A (2013) The state of the art in automated requirements elicitation. Inf Softw Technol 55(10):1695–1709. https://doi.org/10.1016/j.infsof.2013.03.008
52. Yang Z, Li Z, Jin Z, Chen Y (2014) A Systematic Literature Review of Requirements Modeling and Analysis for Self-adaptive Systems. In: Requirements Engineering: Foundation for Software Quality, vol. 8396, C. Salinesi and I. van de Weerd, Eds., in Lecture Notes in Computer Science, vol. 8396. , Cham: Springer International Publishing, pp 55–71. https://doi.org/10.1007/978-3-319-05843-6_5
53. Abdouli M, Karaa WBA, Ghezala HB (2016) Survey of works that transform requirements into UML diagrams. In: 2016 IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA), Towson, MD, USA: IEEE, pp 117–123. https://doi.org/10.1109/SERA.2016.7516136
54. Dawood OS, Sahraoui A-E-K (2017) From requirements engineering to uml using natural language processing—survey study. EJERS 2(1):44. https://doi.org/10.24018/ejers.2017.2.1.236
55. Schön E-M, Thomaschewski J, Escalona MJ (2017) Agile requirements engineering: a systematic literature review. Comput Stand Interfaces 49:79–91. https://doi.org/10.1016/j.csi.2016.08.011
56. Ahmed S, Ahmed A, Eisty NU (2022) Automatic Transformation of Natural to Unified Modeling Language: A Systematic Review. In: 2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA), Las Vegas, NV, USA: IEEE, pp 112–119. https://doi.org/10.1109/SERA54885.2022.9806783
57. Kolahdouz-Rahimi S, Lano K, Lin C (2023) Requirement Formalisation using Natural Language Processing and Machine Learning: A Systematic Review. In: presented at the 11th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2023), Lisbon, Portugal
58. Dybå T, Kitchenham B, Jørgensen M (2005) Evidence-based Software engineering for practitioners. IEEE Softw 22(1):58–65
59. Kitchenham B (2004) Procedures for performing systematic reviews. Keele University, Keele
60. Kitchenham B, Charters S (2007) Guidelines for performing systematic literature reviews in software engineering. Keele University and Durham University, Keele
61. Daun M, Grubb AM, Stenkova V, Tenbergen B (2023) A systematic literature review of requirements engineering education. Requirements Eng 28(2):145–175. https://doi.org/10.1007/s00766-022-00381-9
62. Dybå T, Dingsøyr T (2008) Empirical studies of agile software development: a systematic review. Inf Softw Technol 50(9–10):833–859
63. Pinquié R, Véron P, Segonds F, Croué N (2018) A requirement mining framework to support complex sub-systems suppliers. Procedia CIRP 70:410–415. https://doi.org/10.1016/j.procir.2018.03.228
64. Alkhader Y, Hudaib A, Hammo B (2006) Experimenting With Extracting Software Requirements Using NLP Approach. In: 2006 International Conference on Information and Automation, Colombo, Sri Lanka: IEEE, pp 349–354. https://doi.org/10.1109/ICINFA.2006.374136
65. Ankori R (2005) Automatic requirements elicitation in agile processes. In: IEEE International Conference on Software - Science, Technology & Engineering (SwSTE'05), Herzlia, Israel: IEEE, pp 101–109. https://doi.org/10.1109/SWSTE.2005.8
66. Kamalrudin M (2009) Automated Software Tool Support for Checking the Inconsistency of Requirements. In: 2009 IEEE/ACM International Conference on Automated Software Engineering, Auckland, New Zealand: IEEE, pp 693–697. https://doi.org/10.1109/ASE.2009.38
67. Sharma R, Biswas KK (2013) A Semi-automated Approach towards Handling Inconsistencies in Software Requirements. In: Evaluation of Novel Approaches to Software Engineering, vol. 410, L. A. Maciaszek and J. Filipe, Eds., in Communications in Computer and Information Science, vol. 410. , Berlin, Heidelberg: Springer Berlin Heidelberg, pp 142–156. https://doi.org/10.1007/978-3-642-45422-6_10
68. Moketar NA, Kamalrudin M, Sidek S, Robinson M, Grundy J (2016) An automated collaborative requirements engineering tool for better validation of requirements. In: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering - ASE 2016, Singapore, Singapore: ACM Press, pp 864–869. https://doi.org/10.1145/2970276.2970295

69. Antinyan V, Staron M (2017) Rendex: a method for automated reviews of textual requirements. J Syst Softw 131:63–77. https://doi.org/10.1016/j.jss.2017.05.079
70. Parra E, Dimou C, Llorens J, Moreno V, Fraga A (2015) A methodology for the classification of quality of requirements using machine learning techniques. Inf Softw Technol 67:180–195. https://doi.org/10.1016/j.infsof.2015.07.006
71. Cleland-Huang J, Settimi R, Zou X, Solc P (2007) Automated classification of non-functional requirements. Requirements Eng 12(2):103–120. https://doi.org/10.1007/s00766-007-0045-1
72. Cybulski JL, Reed K (2000) Requirements Classification and Reuse: Crossing Domain Boundaries. In: International Conference on Software Reuse (ICSR 2000), pp 190–210
73. Winkler J, Vogelsang A (2016) Automatic Classification of Requirements Based on Convolutional Neural Networks. In: 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW), Beijing, China: IEEE, pp 39–45. https://doi.org/10.1109/REW.2016.021
74. Reddivari S, Bhowmik T, Hollis C (2019) Automated support to capture verbal just-in-time requirements via audio mining and cluster-based visualization. J Ind Inf Integr 14:41–49. https://doi.org/10.1016/j.jii.2018.06.001
75. Rago A, Marcos C, Diaz-Pace JA (2016) Identifying duplicate functionality in textual use cases by aligning semantic actions. Softw Syst Model 15(2):579–603. https://doi.org/10.1007/s10270-014-0431-3
76. Li Y, Yue T, Ali S, Zhang L (2019) Enabling automated requirements reuse and configuration. Softw Syst Model 18(3):2177–2211. https://doi.org/10.1007/s10270-017-0641-6
77. Vidya Sagar VBR, Abirami S (2014) Conceptual modeling of natural language functional requirements. J Syst Softw 88:25–41. https://doi.org/10.1016/j.jss.2013.08.036
78. Al-Hroob A, Imam AT, Al-Heisa R (2018) The use of artificial neural networks for extracting actions and actors from requirements document. Inf Softw Technol 101:1–15. https://doi.org/10.1016/j.infsof.2018.04.010
79. Thakur JS, Gupta A (2014) Automatic generation of sequence diagram from use case specification. In: Proceedings of the 7th India Software Engineering Conference on - ISEC '14, Chennai, India: ACM Press, pp 1–6. doi: https://doi.org/10.1145/2590748.2590768
80. Elallaoui M, Nafil K, Touahni R (2018) Automatic transformation of user stories into UML use case diagrams using NLP techniques. Procedia Comput Sci 130:42–49. https://doi.org/10.1016/j.procs.2018.04.010
81. Robeer M, Lucassen G, van der Werf JMEM, Dalpiaz F, Brinkkemper S (2016) Automated Extraction of Conceptual Models from User Stories via NLP. In: 2016 IEEE 24th International Requirements Engineering Conference (RE), Beijing: IEEE, pp 196–205. https://doi.org/10.1109/RE.2016.40
82. Nasiri S, Rhazali Y, Lahmer M, Chenfour N (2020) Towards a generation of class diagram from user stories in agile methods. Procedia Comput Sci 170:831–837. https://doi.org/10.1016/j.procs.2020.03.148
83. Nguyen TH, Vo BQ, Lumpe M, Grundy J (2012) REInDetector: a framework for knowledge-based requirements engineering. In: Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering - ASE 2012, Essen, Germany: ACM Press, p 386. https://doi.org/10.1145/2351676.2351754
84. Parasuraman R, Sheridan TB, Wickens CD (2000) A model for types and levels of human interaction with automation. IEEE Trans Syst Man Cybern A 30(3):286–297. https://doi.org/10.1109/3468.844354
85. Kiyavitskaya N, Zannone N (2008) Requirements model generation to support requirements elicitation: the Secure Tropos experience. Autom Softw Eng 15(2):149–173. https://doi.org/10.1007/s10515-008-0028-6
86. Cybulski JL, Reed K (1998) Computer-assisted analysis and refinement of informal software requirements documents. In: Proceedings 1998 Asia Pacific Software Engineering Conference (Cat. No.98EX240), Taipei, Taiwan: IEEE Comput. Soc, pp 128–135. https://doi.org/10.1109/APSEC.1998.733606
87. Méndez Fernández D, Wagner S (2015) Naming the pain in requirements engineering: a design for a global family of surveys and first results from Germany. Inf Softw Technol 57:616–643. https://doi.org/10.1016/j.infsof.2014.05.008
88. Deeptimahanti DK, Babar MA (2009) An Automated Tool for Generating UML Models from Natural Language Requirements. In: 2009 IEEE/ACM International Conference on Automated Software Engineering, Auckland, New Zealand: IEEE, pp 680–682. https://doi.org/10.1109/ASE.2009.48
89. Seresht SM, Ormandjieva O, Sabra S (2008) Automatic Conceptual Analysis of User Requirements with the Requirements Engineering Assistance Diagnostic (READ) Tool. In: 2008 Sixth International Conference on Software Engineering Research, Management and Applications, Prague, Czech Republic: IEEE, pp 133–142. https://doi.org/10.1109/SERA.2008.34
90. Ibrahim M, Ahmad R (2010) Class Diagram Extraction from Textual Requirements Using Natural Language Processing (NLP) Techniques. In: 2010 Second International Conference on Computer Research and Development, Kuala Lumpur, Malaysia: IEEE, pp 200–204. https://doi.org/10.1109/ICCRD.2010.71
91. Mu Y, Wang Y, Guo J (2009) Extracting Software Functional Requirements from Free Text Documents. In: 2009 International Conference on Information and Multimedia Technology, Jeju Island, Korea (South): IEEE, pp 194–198. https://doi.org/10.1109/ICIMT.2009.47
92. Deeptimahanti DK, Sanyal R (2011) Semi-automatic generation of UML models from natural language requirements. In: Proceedings of the 4th India Software Engineering Conference on - ISEC '11, Thiruvananthapuram, Kerala, India: ACM Press, pp 165–174. https://doi.org/10.1145/1953355.1953378
93. Nattoch Dag J, Thelin T, Regnell B (2006) An experiment on linguistic tool support for consolidation of requirements from multiple sources in market-driven product development. Empir Softw Eng 11(2):303–329. https://doi.org/10.1007/s10664-006-6405-5
94. Lucassen G, Dalpiaz F, van der Werf JMEM, Brinkkemper S (2016) Improving agile requirements: the quality user story framework and tool. Requirements Eng 21(3):383–403. https://doi.org/10.1007/s00766-016-0250-x
95. Reinhartz-Berger I, Kemelman M (2020) Extracting core requirements for software product lines. Requirements Eng 25(1):47–65. https://doi.org/10.1007/s00766-018-0307-0
96. Casamayor A, Godoy D, Campo M (2010) Identification of non-functional requirements in textual specifications: a semi-supervised learning approach. Inf Softw Technol 52(4):436–445. https://doi.org/10.1016/j.infsof.2009.10.010
97. Burgueno L, Cabot J, Gerard S (2019) An LSTM-Based Neural Network Architecture for Model Transformations. In: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS), Munich, Germany: IEEE, pp 294–299. https://doi.org/10.1109/MODELS.2019.00013
98. Sampaio A, Rashid A, Chitchyan R, Rayson P (2007) EA-miner: towards automation in aspect-oriented requirements engineering. In: Transactions on Aspect-Oriented Software Development III, Springer, Berlin, Heidelberg, pp 4–39
99. Gacitua R, Sawyer P, Gervasi V (2011) Relevance-based abstraction identification: technique and evaluation. Requirements Eng 16(3):251–265. https://doi.org/10.1007/s00766-011-0122-3
100. Omoronyia I, Sindre G, Stålhane T, Biffl S, Moser T, Sunindyo W (2010) A domain ontology building process for guiding requirements

elicitation. In: Requirements Engineering: Foundation for Software Quality. REFSQ 2010., Springer, Berlin, Heidelberg, pp 188–202
101. Ferrari A, Esuli A (2019) An NLP approach for cross-domain ambiguity detection in requirements engineering. Autom Softw Eng 26(3):559–598. https://doi.org/10.1007/s10515-019-00261-7
102. Nattoch Dag J, Regnell B, Carlshamre P, Andersson M, Karlsson J (2002) A feasibility study of automated natural language requirements analysis in market-driven development. Requirements Eng 7(1):20–33. https://doi.org/10.1007/s007660200002
103. Park S, Kim H, Ko Y, Seo J (2000) Implementation of an efficient requirements-analysis supporting system using similarity measure techniques. Inf Softw Technol 42(6):429–438. https://doi.org/10.1016/S0950-5849(99)00102-0
104. Castro-Herrera C, Duan C, Cleland-Huang J, Mobasher B (2009) A recommender system for requirements elicitation in large-scale software projects. In: Proceedings of the 2009 ACM symposium on Applied Computing - SAC '09, Honolulu, Hawaii: ACM Press, p 1419. https://doi.org/10.1145/1529282.1529601
105. Rago A, Marcos C, Diaz-Pace JA (2013) Uncovering quality-attribute concerns in use case specifications via early aspect mining. Requirements Eng 18(1):67–84. https://doi.org/10.1007/s00766-011-0142-z
106. Shibaoka M, Kaiya H, Saeki M (2007) GOORE: Goal-Oriented and Ontology Driven Requirements Elicitation Method. In: Advances in Conceptual Modeling – Foundations and Applications. ER 2007, Springer, Berlin, Heidelberg, pp 225–234
107. Pavlidis M, Islam S, Mouratidis H (2012) A CASE Tool to Support Automated Modelling and Analysis of Security Requirements, Based on Secure Tropos. In: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, vol. 8827, Bayro-Corrochano E, Hancock E, Eds., in Lecture Notes in Computer Science, vol. 8827. , Cham: Springer International Publishing, pp 95–109. https://doi.org/10.1007/978-3-642-29749-6_7
108. Wang Z, Chen C-H, Zheng P, Li X, Khoo LP (2019) A novel data-driven graph-based requirement elicitation framework in the smart product-service system context. Adv Eng Inform 42:100983. https://doi.org/10.1016/j.aei.2019.100983
109. Lucassen G, Robeer M, Dalpiaz F, van der Werf JMEM, Brinkkemper S (2017) Extracting conceptual models from user stories with Visual Narrator. Requirements Eng 22(3):339–358. https://doi.org/10.1007/s00766-017-0270-1
110. Duan C, Laurent P, Cleland-Huang J, Kwiatkowski C (2009) Towards automated requirements prioritization and triage. Requirements Eng 14(2):73–89. https://doi.org/10.1007/s00766-009-0079-7
111. Kuk K, Angeleski M, Popovic B (2019) A Semi-automated generation of Entity-Relationship Diagram based on Morphosyntactic Tagging from the Requirements Written in a Serbian Natural Language. In: 2019 IEEE 19th International Symposium on Computational Intelligence and Informatics and 7th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics (CINTI-MACRo), Szeged, Hungary: IEEE, pp 000085–000092. https://doi.org/10.1109/CINTI-MACRo49179.2019.9105162
112. Kurtanovic Z, Maalej W (2017) Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning. In: 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal: IEEE, pp 490–495. https://doi.org/10.1109/RE.2017.82
113. Jabbarin S, Arman N (2014) Constructing use case models from Arabic user requirements in a semi-automated approach. In: 2014 World Congress on Computer Applications and Information Systems (WCCAIS), Hammamet, Tunisia: IEEE, pp 1–4. https://doi.org/10.1109/WCCAIS.2014.6916558
114. Sardinha A, Chitchyan R, Weston N, Greenwood P, Rashid A (2013) EA-Analyzer: automating conflict detection in a large set of textual aspect-oriented requirements. Autom Softw Eng 20(1):111–135. https://doi.org/10.1007/s10515-012-0106-7
115. Alami N, Arman N, Khamyseh F (2017) A semi-automated approach for generating sequence diagrams from Arabic user requirements using a natural language processing tool. In: 2017 8th International Conference on Information Technology (ICIT), Amman, Jordan: IEEE, pp 309–314. https://doi.org/10.1109/ICITECH.2017.8080018
116. Ott D (2013) Automatic Requirement Categorization of Large Natural Language Specifications at Mercedes-Benz for Review Improvements. In: Requirements Engineering: Foundation for Software Quality, vol. 7830, J. Doerr and A. L. Opdahl, Eds., in Lecture Notes in Computer Science, vol. 7830. , Berlin, Heidelberg: Springer Berlin Heidelberg, pp 50–64. https://doi.org/10.1007/978-3-642-37422-7_4
117. Elbendak M, Vickers P, Rossiter N (2011) Parsed use case descriptions as a basis for object-oriented class model generation. J Syst Softw 84(7):1209–1223. https://doi.org/10.1016/j.jss.2011.02.025
118. Nguyen TH, Grundy J, Almorsy M (2015) Rule-based extraction of goal-use case models from text. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, Bergamo Italy: ACM, pp 591–601. https://doi.org/10.1145/2786805.2786876
119. Segundo LM, Herrera RR, Herrera KYP (2007) UML Sequence Diagram Generator System from Use Case Description Using Natural Language. In: Electronics, Robotics and Automotive Mechanics Conference (CERMA 2007), Morelos: IEEE, pp 360–363. https://doi.org/10.1109/CERMA.2007.4367713
120. Qureshi MZ, Azhar A, Abubakar Q, Rana TA, Maqbool A (2021) Capturing Users Requirements Using a Data Mining Approach. In: 2021 International Conference on Communication Technologies (ComTech), Rawalpindi, Pakistan: IEEE, pp 49–54. https://doi.org/10.1109/ComTech52583.2021.9616939
121. Kamalrudin M, Hosking J, Grundy J (2017) MaramaAIC: tool support for consistency management and validation of requirements. Autom Softw Eng 24(1):1–45. https://doi.org/10.1007/s10515-016-0192-z
122. Shehadeh K, Arman N, Khamayseh F (2021) Semi-Automated Classification of Arabic User Requirements into Functional and Non-Functional Requirements using NLP Tools. In: 2021 International Conference on Information Technology (ICIT), Amman, Jordan: IEEE, pp 527–532. https://doi.org/10.1109/ICIT52682.2021.9491698
123. Binti Mustaffa NFN, Bin Sallim J, Binti Mohamed R (2021) Semi – Automated Software Requirement Specification (SRS) Document Generator: The Guideline to Novice System Analyst. In: 2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM), Pekan, Malaysia: IEEE, pp 80–85. https://doi.org/10.1109/ICSECS52883.2021.00022
124. Saini R, Mussbacher G, Guo JLC, Kienzle J (2022) Automated, interactive, and traceable domain modelling empowered by artificial intelligence. Softw Syst Model 21(3):1015–1045. https://doi.org/10.1007/s10270-021-00942-6