IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 29 December 2024, accepted 28 January 2025, date of publication 5 February 2025, date of current version 24 February 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3539357

SURVEY

Enhancing Agile Software Development: A Systematic Literature Review of Requirement Prioritization and Reprioritization Techniques

NOSHIN TASNEEM, HAZURA BINTI ZULZALIL, AND SA'ADAH HASSAN

Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang 43400, Malaysia

Corresponding author: Hazura Binti Zulzalil (hazura@upm.edu.my)

This work was supported in part by Universiti Putra Malaysia (UPM).

ABSTRACT Agile software development places great importance on requirement prioritization and reprioritization, which allow teams to concentrate on providing the most beneficial features to satisfy stakeholders. Most systematic review articles on prioritization techniques focus on traditional methods and ignore recent approaches that use artificial intelligence (AI). Additionally, there is a notable scarcity of literature addressing the neglected domain of reprioritisation and a lack of review papers analyzing semi-automated approaches. To fill this gap, this systematic literature review includes an in-depth review of newly developed AI-based and semi-automated techniques, in addition to widely used traditional prioritization methods. This study analyzed 76 primary studies from five credible electronic databases (Springer Link, IEEE Xplore, Scopus, Science Direct, and ACM Digital Library) to address six selected research questions. This literature review paper is unique in that it covers conventional approaches, reprioritization techniques, and AI-based and semi-automated techniques in a single review, which has not been done in previous papers. The findings highlight the strength and weakness of each technique. This review also identifies the most commonly used prioritization techniques in agile software development and the key challenges in requirement prioritization. Future research opportunities in the field of reprioritization are revealed by the gaps identified in the literature. This research contributes to enhancing the agility and effectiveness of agile software development (ASD).

INDEX TERMS Agile software development, AI-based approaches, fuzzy logic, machine learning, optimization, prioritization techniques, requirement prioritization, reprioritization, semi-automated approaches, systematic review.

I. INTRODUCTION

Requirement Engineering (RE) is one of the most important components of software engineering and involves systematic elicitation, analysis, prioritization, and documentation of stakeholder requirements. It is aligned with the client's expectations while dealing with contextual factors and managing changing requirements efficiently [1], [2], [3]. RE consists of several processes, including requirement prioritization (RP), which helps to perform informed decision making to help software teams meet the needs of stakeholders within time, cost, and technological constraints [4], [5], [6]. RP plays an important role in delivering valuable features, with better resource allocation and project success [7], [8].

Agile methodologies have gained popularity in software engineering owing to their emphasis on client collaboration, incremental delivery, and responsiveness to change [9], [10]. Scrum and extreme programming (XP) have become common methods because they successfully integrate requirement prioritization into development

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

processes [11], [12], [13]. Researchers and practitioners greatly appreciate RP in agile requirement engineering, which is motivated by business value and active customer interaction [14], [15].

RP is an important process in agile software development, but it has hurdles. Managing various stakeholder expectations and project timelines to align priorities to business objectives is a perennial challenge. While business value is frequently used as the primary criterion, studies recommend combining additional criteria, such as developer perspectives, cost, and requirement interdependencies, to improve decision making [16], [17], [18].

Another challenge in requirement prioritization (RP) is choosing the best technique from a variety of possibilities, which is determined by the project context, stakeholder understanding, and the relevance of criteria at different phases [19], [20], [21], [22]. The literature identifies a wide range of conventional techniques for improving the accuracy, efficiency, dependability, and conflict resolution in requirement prioritization (RP). Commonly used methods include The analytical hierarchy process (AHP), MoSCoW, cost-value ranking, cumulative voting, planning game (PG), kano model, numerical assignment, binary search tree (BST), bubble sort, value-oriented prioritization (VOP), quality functional deployment (QFD), and wieger's method [21], [23], [24], [25], [26]. Each of these techniques has advantages and disadvantages, making it appropriate for a variety of situations. However, no single approach is always advantageous because technique selection is significantly influenced by the project's environment and requirements [8], [23], [27]. While conventional techniques, such as AHP and CV are trustworthy and widely used, they lack scalability and become inefficient with increasing requirements. Techniques such as the Kano model, cost value ranking, planning game, and pairwise comparisons also face scalability issues [28]. Dot voting, planning poker, and QFD require low effort [24], [29]. Cumulative voting and planning game are less complex. Binary search trees, minimal spanning trees, and priority groups are reliable, fault-tolerant, and user-friendly [30]. Current requirement prioritization techniques still suffer from various issues such as limited scalability, ambiguity, uncertainty, complexities in application, high computational costs, and absence of automation [2], [29], [31].

A pressing issue in requirements engineering is the inherently dynamic nature of the requirements, which are frequently altered, added, or postponed. Although reprioritisation is an important aspect of dealing with this kind of change, it is still relatively underexplored in agile contexts [2], [19]. Current research and industry practices have revealed significant inefficiencies in requirement reprioritization techniques, resulting in poor resource reallocation, dissatisfied customers, project delays, and interruptions. This emphasizes the urgent need to develop dynamic reprioritisation techniques that can respond to changing project requirements [32], [33], [34].

Artificial intelligence (AI)-based techniques, such as fuzzy logic, machine learning, and optimization, can help overcome some constraints of requirement prioritization [2], [31], [35]. Although such techniques provide enhanced scalability and efficiency, no single AI-based technique can resolve all the complex challenges found in RP. Fuzzy logic is known for its scalability, but requires accuracy improvements, whereas optimization approaches are precise but slow for large requirement sets. Machine learning offers scalable solutions, but may lack accuracy in minimizing the disagreements [2].

The majority of software requirement prioritization is done manually, which is both resource-intensive and unproductive, particularly for large projects. This lack of automation causes delays, greater expenses, and increased risk [36], [37]. Reliance on professional expertise frequently leads to biased evaluations, compromising the quality of prioritization and overall project performance. Semi-automated methods use both manual and automated processes to improve requirement prioritization by considering aspects such as importance, time, cost, and risk [38], [39], [40].

This study is motivated by the growing acceptance of agile approaches in the software industry as well as the importance of effective RP in iterative development, continuing feedback, and flexibility [21], [24], [41]. Existing systematic studies on prioritization techniques generally concentrate on traditional methods, ignoring newer AI-based strategies and the critical features of reprioritization. Additionally, there is a scarcity of studies on semi-automated approaches. This review study attempts to fill these gaps by conducting an in-depth review of traditional, AI-based, and semi-automated prioritization techniques as well as reprioritization strategies. It also highlights the most commonly used techniques in agile software development and addresses the prevalent challenges in requirement prioritization. This study highlights the need for dynamic reprioritization strategies to react efficiently to changing project conditions while addressing stakeholder needs.

The remainder of the paper is organized as follows: Section II describes the research methodology; Section III presents the research findings; Section IV explains the implications of the findings for research and practice; Section V clarifies threats to validity; Section VI shares directions for future work; and Section VII concludes the study.

II. RESEARCH METHODOLOGY

In accordance with Kitchenham and Charters' guidance, this study uses a systematic literature review methodology to examine requirement prioritization and reprioritization techniques in agile software development (ASD) [42]. It is important to conduct a well-organized study that adheres to norms, rules, and systematic methodologies [43].

The SLR process comprises three fundamental components, which are regarded as essential principles for conducting a dependable research endeavor: (1) planning process, (2) conducting process, and (3) reporting process. Each

component included additional nested activities, as delineated in the accompanying diagram (Fig. 1).

Research questions were formulated to guide this review. Keywords were selected to find relevant papers, and the inclusion, exclusion, and quality criteria ensured the selection of high-standard primary studies. Data from these studies were extracted and synthesized to address these questions.

This research methodology acts as a framework for answering the selected research questions. The components of the systematic literature review (SLR) protocol process shown in Fig. 1 are described in the following subsections.

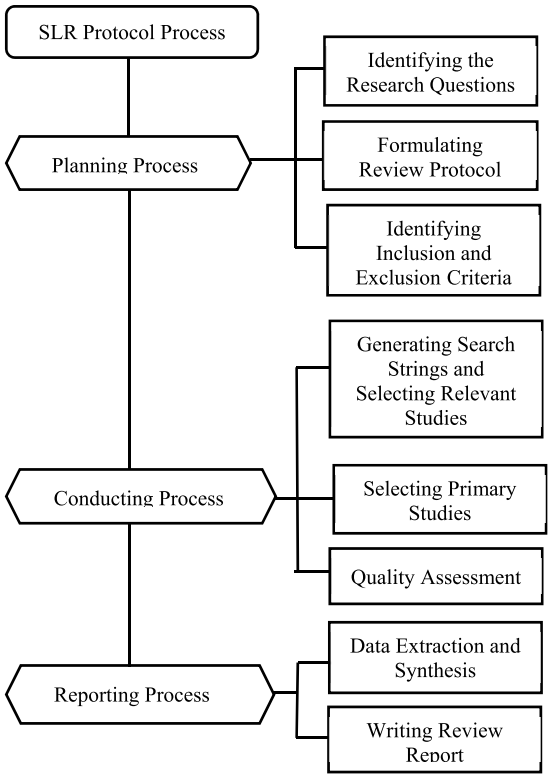

FIGURE 1. SLR protocol process.

The next section consists of the planning, conducting research, and reporting processes.

A. PLANNING PROCESS

The first phase and the most important phase is the planning phase, which included formulating the research questions, establishing a review protocol, and determining the inclusion and exclusion criteria. The PICOC strategy was used to structure the research questions and develop a protocol.

1) RESEARCH QUESTIONS

This research explores techniques for requirement prioritization, including reprioritization and AI-based and semi-automated approaches. The goal of this study was to determine which agile prioritization techniques are most frequently utilized, as well as the difficulties that arise from handling requirement changes and the current issues with requirement prioritization in agile software development. In addition, this study addressed the need for reprioritization techniques. To accomplish this goal, six research questions were developed. In this review article, we applied our protocol to identify the maximum number of relevant studies to appropriately respond to the research questions. Table 1 lists these six research questions.

TABLE 1. Research questions.

No.	Research Question
RQ1	Which requirement prioritization techniques are the most commonly used in agile software development?
RQ2	What are the existing challenges associated with requirement prioritization in agile software development?
RQ3	What are the AI based approaches in requirement prioritization?
RQ4	Which semi-automated approaches are currently available for requirement prioritization?
RQ5	Why reprioritization is important within agile software development?
RQ6	What are the available reprioritization techniques?

2) FORMULATING REVIEW PROTOCOL

To reduce study bias, Kitchenham and Charters [42], along with Kitchenham [44], developed a protocol for conducting systematic reviews. These protocols ensure consistency and transparency in research, reduce omissions, and enhance the reliability.

The steps of the review protocol were clarified through the following processes: (i) developing the research questions, (ii) formulating a search strategy, (iii) developing standards and guidelines for choosing studies, (iv) developing protocols for assessing quality, (v) carrying out a plan for data extraction, and (vi) synthesizing the extracted data. The review procedure is illustrated in Fig. 2.

3) INCLUSION AND EXCLUSION CRITERIA

It is essential to use inclusion and exclusion criteria to ensure that the chosen studies are pertinent to the research questions and fulfil the goals of the systematic literature review. Numerous research publications including books, journals, conference papers, workshop proceedings, symposium papers, and other research items were found during the search process. Using this approach, the review was guaranteed to contain only relevant quality studies.

Research articles in English, including books, conference proceedings, and journals that dealt with requirement prioritization and reprioritization in agile software development (ASD), were included in the search. Additionally, articles published between 2000 and 2024 were included to capture the evolution of requirement prioritization from Agile Manifesto (2001) [9] to modern AI-driven methods. Excluded studies were those that were less than four pages in length, published in languages other than English, or unrelated to

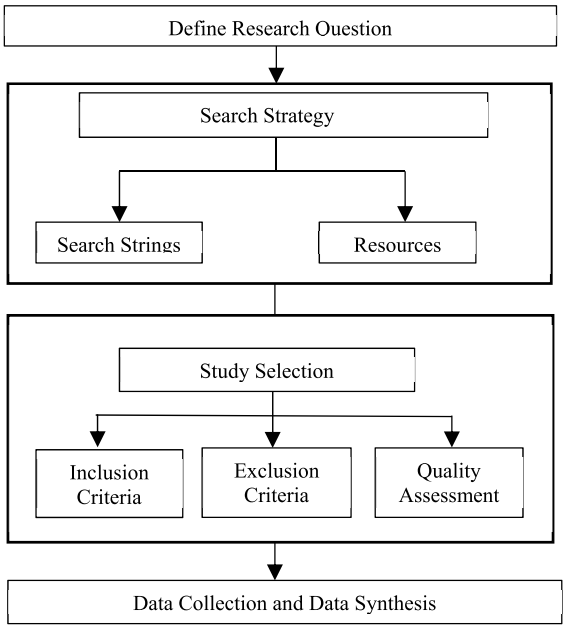

FIGURE 2. Review protocol stages.

the subject. Table 2 summarizes the inclusion and exclusion criteria.

TABLE 2. Inclusion and exclusion criteria.

Inclusion Criteria	Exclusion Criteria
The articles were required to be in English and published	Articles not published in the English language
Articles discussing requirements prioritization techniques	Articles not discussing requirement prioritization techniques
Articles published from 2000 to 2024 to capture the evolution from Agile to AI-based techniques	Articles below 2000
Articles that aligns to the research questions	Unrelated articles to the research questions
The articles that are four pages or longer	The article less than 4 pages

B. CONDUCTING PROCESS

After the planning phase, the next most important phase of the SLR protocol is the conducting process. It includes several stages to ensure the unbiased gathering of relevant studies. Following the establishment of the protocol during the planning stage, the conducting step began with a review. This stage included establishing the search strategy, conducting the research selection process, selecting primary studies, and evaluating the quality of these studies. The following subsection describes all stages of the conducting process.

1) SEARCH STRATEGY

Developing and implementing a successful search strategy are critical steps in conducting systematic literature reviews. This requires careful planning and control because it is time-consuming and prone to errors. Therefore, a systematic approach is needed to design, impose, and test suitable search strategies that allow efficient retrieval of relevant literature from digital databases [45]. Numerous sources, including books, journals, and conference papers, were found by employing search strings and key terms specific to the research question. To ensure the selection of reliable and unbiased studies, the findings were further filtered using precise inclusion and exclusion criteria. This approach provides a strong foundation for detailed analysis and synthesis.

a: SEARCH STRING

The correct identification of search strings is a critical stage that determines the effectiveness of gathering all potentially relevant research in our field. We followed the guidelines given by Kitchenham and Charters [42] and the procedures by Kitchenham [44] for creating search strings. The methods they provide involve identifying the most important terms from the research questions and searching through acronyms and alternative spellings for each key term. We then apply the "AND" and "OR" Boolean operators. The operators "AND" and "OR" are used to associate important phrases and synonyms, abbreviations, and alternative terms, respectively.

Key terms must be incorporated to obtain the final search strings. First, we decided which terms from our research questions were important for inclusion in our search string. Next, we collected synonyms and related terms that are frequently used in requirement prioritization and agile software development. We considered other names for the algorithms to ensure that all pertinent AI strategies were covered. We then used Boolean operators, such as "AND" and "OR," to combine these terms into search strings so that a more in-depth analysis could be performed.

The following terms are the identified key terms for our study:[("Software Requirements" OR "User Story Requirements") AND ("Prioritization Techniques" OR "Prioritization Approaches" OR "Ranking Strategy") AND ("Agile Software Development" OR "ASD") AND ("Challenges" OR "Issues" OR "Limitations") AND (Semi-automated Approaches" OR "Semi-automated Techniques") AND ("AI Based Approaches" OR "Fuzzy Logic" OR "Machine Learning" OR "Optimization" OR "Deep Learning" OR "Neural Network") AND ("Reprioritization Techniques" OR "Reprioritization Approaches") AND ("Requirement Change" OR "Change Management")].

These are possible combinations of keywords used to generate the search strings. These were useful for collecting all possible relevant literature to answer the research questions.

b: RESEARCH RESOURCES

Choosing the right databases is vital for efficient SLR. Researchers want to locate resources that are relevant to their topic, publicly available, and have maximum studies. It is essential to note that a few databases may require precise search techniques owing to their precise search engine

structure and grammar. For example, the search on ScienceDirect is limited to eight search operators (such as ''OR'' or ''AND''). Search phrases were split into separate searches if they had more than eight operators.

We adopted a two-step approach to search for relevant studies. First, we conducted an automated search of five online databases: IEEE Xplore, Springer Link, Scopus, Science Direct, and the ACM Digital Library. The databases used are listed in Table 3.

TABLE 3. Online databases.

Database Name	URL
IEEE Xplore	https://ieeexplore.ieee.org
Springer Link	https://link.springer.com
Science Direct	https://www.sciencedirect.com
Scopus	https://www.scopus.com
ACM Digital Library	https://dl.acm.org

Second, we performed a manual search using a backward-forward search method to identify additional relevant studies based on references referred to in the preliminary set of studies [46]. We also used Google Scholar to identify studies that referred to the selected primary studies.

2) STUDY SELECTION PROCESS

The preliminary search yielded a preliminary listing of 4317 studies from five online databases. We only considered journals and conference papers in the selection process. Multiple filtering and depuration stages were applied to 4317 studies. We followed the guidelines of Kitchenham and Charters [42] and the procedures of Kitchenham [44] to select relevant studies. First, we applied the inclusion and exclusion criteria illustrated in Table 2 and obtained 2985 filtered studies. Second, we eliminated duplicate studies that were discovered in numerous databases throughout the search process using the mendeley reference manager and obtained 2766 studies. Third, 530 studies were obtained after considering and assessing the titles and abstracts and whether they were relevant to the research questions. In the fourth step, to obtain the expected set of primary studies, full-text screening was conducted. Finally, 67 primary studies were included. To bring our research to date, a journal article with two duplicates must be included, instead of a conference paper. Next, to identify any missing publications, we used a snowballing search approach to follow the citations and references of a subset of research. This resulted in the inclusion of 13 articles in the first iteration. The 13 articles were examined again in the second iteration to determine if there were any more missing articles. Therefore, 80 primary studies were included in this analysis. Ultimately, four papers were eliminated after assessing the quality of the chosen primary studies. The research selection procedure is illustrated in Fig. 3.

The final number of primary studies included 76. Appendix A presents the selected primary studies.

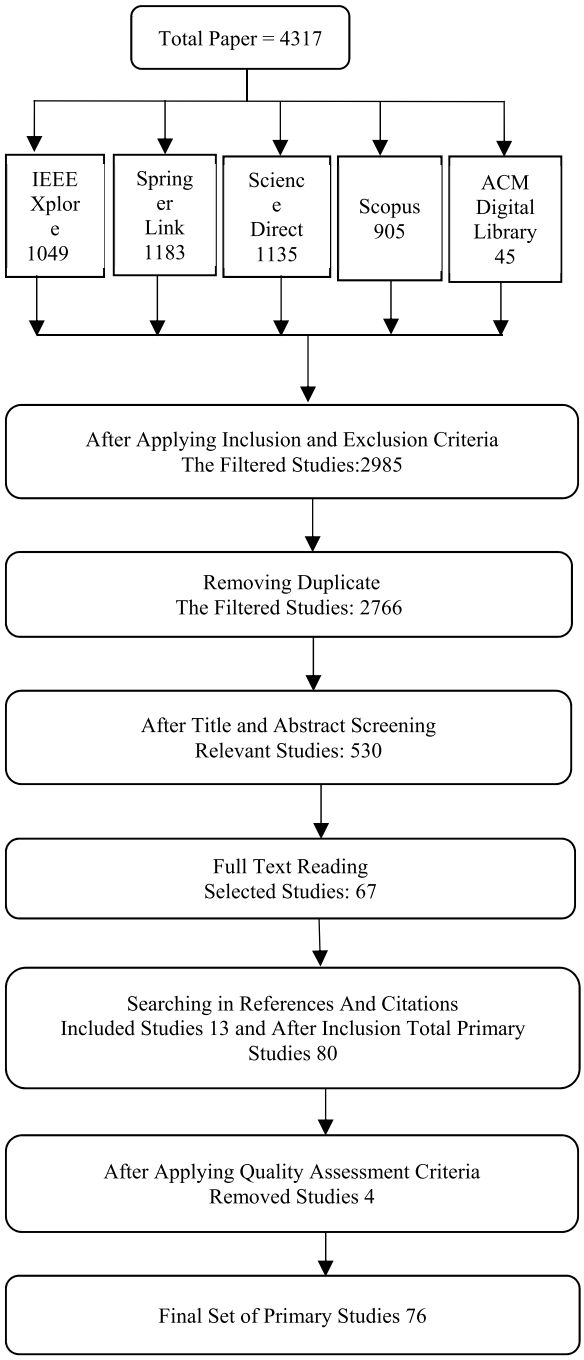

FIGURE 3. Study selection process.

3) QUALITY ASSESSMENT

Quality assessment is an important part of conducting a systematic literature review (SLR) to ensure the reliability and validity of the findings. Several key criteria were used to assess the quality of the studies. We followed the guidelines of Kitchenham and Charters [42] to establish the quality assessment checklist, as shown in Table 4. Three levels were used to measure the study quality: high, medium, and low. Subsequently, answers to each question were divided into

three categories. Number 1 in the first category denotes a complete and unambiguous fulfilment of the quality requirements. Number 0 in the second category denotes that the item does not satisfy any of the specified quality assessment criteria. Number 0.5 in the last category denotes only partially meets the specified conditions.

TABLE 4. Quality assessment checklist.

QA ID	Quality checklist questions
QA 1	Are the research aims and objectives seem to be well-defined?
QA 2	Is the research design well-defined?
QA 3	Does the study concentrate on the related domains of RQs?
QA 4	Does the researcher present sufficient evidence to support their findings and conclusions?
QA 5	Does the publication source have a high impact factor or reputation in the field?

The scores for the five criteria were gathered for quality evaluation after applying the assessment criteria. The quality evaluation established a range of three levels. A high level is indicated by a range of scores between 4.5 and 5.0, a medium level by a range of scores between 3.5 and 4.0, and a low level by a range of scores between 2.5 and 3.0. The majority of the scores were high, while four studies were disqualified for failing to meet the requirements. The distribution of the selected 76 primary studies is shown in Fig. 4, following the application of the quality criteria for the three levels.

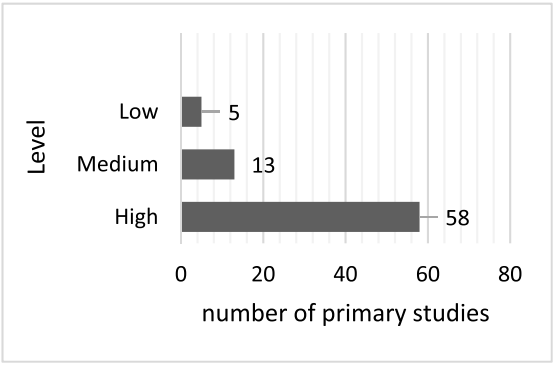

FIGURE 4. Study quality assessment levels.

The findings of the primary study quality assessment criteria are explained in Appendix B.

C. REPORTING PROCESS

The final phase of this systematic review was the reporting process. To conduct a thorough study, this involves extracting and synthesizing data that are in line with the research questions. The findings were compared and combined to uncover the trends, patterns, and gaps. From selection to synthesis, each step of the procedure was recorded to guarantee reliability and credibility.

1) DATA EXTRACTION

The data extraction phase of this systematic literature review focused on systematically gathering relevant information from 76 primary studies. The purpose was to answer the research questions properly and obtain insights into requirement prioritization and reprioritization approaches in agile software development. The extracted items, with small descriptions, are listed in Table 5.

TABLE 5. Data extracted items of the primary studies.

Data extracted item	Description
Study ID	A unique identifier for each study
Title	The title of the study
Authors	Authors name
Year	The year of publication.
Publication type	The name of the journal or conference where the study was published
Research type	Indicates whether the study is empirical (based on experiments or observations) or theoretical.
Techniques	The specific techniques used for requirement prioritization or reprioritization in the study
Key findings	The main findings of the study
Advantages and challenges	Advantages or benefits of the technique and challenges or limitations of the technique
Tools or methods used	Any specific tools or methods used in the study for prioritization or reprioritization
Evaluation metrics	Metrics used to evaluate the techniques or methods
Study quality	A score reflecting the quality of the study based on predefined criteria.
Notes	Any additional notes or comments relevant to the study.

These phases were assessed according to guidelines. We thoroughly assessed every study to extract specific information such as publication year, authors, journal or conference, and title. Furthermore, to understand the nature of the study, we sorted the research according to the theoretical or empirical data. Techniques for prioritizing requirements were reviewed for their applicability, noting both their benefits and drawbacks. To guarantee a thorough assessment, each study was graded according to the quality criteria. This phase identified trends, patterns, and research gaps, offering insights into the evolution of prioritization techniques. By documenting and analyzing the data, we provided actionable recommendations to enhance the agile software development processes.

2) DATA SYNTHESIS

This systematic literature review synthesizes data on requirement prioritization and reprioritization techniques for agile software development. It categorizes 76 primary studies into manual, AI-based, and semi-automated approaches, and

analyzes popular methods. Documenting instances from the research, the synthesis also identifies current challenges related to requirement prioritization in agile. AI-based techniques, focusing on fuzzy logic, optimization, and machine learning, were evaluated alongside semi-automated systems that combine algorithms with human judgement. The importance of reprioritization was identified, and the available reprioritization techniques were documented. Finally, all findings are synthesized into a coherent narrative, key insights are summarized, and gaps in the current research that point to future study directions are highlighted.

III. RESULTS

This section provides the outcomes of the review aimed at addressing the research questions outlined in Table 1. The final 76 primary studies used to answer these questions were narrowed down after applying the stated general and quality criteria, marking the end of the research process, followed by the synthesis and reporting of the final data.

A. RQ1) WHICH REQUIREMENT PRIORITIZATION TECHNIQUES ARE THE MOST COMMONLY USED IN AGILE SOFTWARE DEVELOPMENT?

The aim of this research was to identify the most used requirement prioritization techniques for ASD among the vast number of available techniques. An in-depth literature review was conducted to identify relevant studies and work aiding in the identification of commonly used requirement prioritization techniques in agile.

A study involving a focus group of knowledge experts and industry practitioners found that business value is the most popular factor for prioritizing agile requirements. The MoSCoW method is the most commonly used prioritization strategy, followed by the Kano Model and Cost-Value Ranking, according to that study [S3]. Analytical hierarchy process (AHP), cumulative voting (CV), planning game (PG), quality functional deployment (QFD), binary search tree (BST), and cost value approach (CVA) are techniques often used in agile requirement prioritization [S17]. Compared with other methods, AHP is the most frequently used and cited method. Because AHP can accurately estimate requirement matches, it can provide more reliable priority results. AHP, QFD, and BST were slow priorities, whereas kano, CV, MoSCoW, VOP, and PG were the fastest. Less effortful methods include QFD, planning pokers, and dot voting [S7], [S24]. Table 6 lists the most commonly used requirement prioritization techniques for agile software development (ASD). Each technique is listed along with the references that have cited or mentioned them. The techniques included in the table are widely recognized within the agile community for their effectiveness in managing and prioritizing requirements. Fig. 5 graphically represents the number of references or frequencies for each technique mentioned in Table 6, providing a visual overview of their popularity and usage trends.

From the above table, it is clear that MoSCoW is the most commonly used technique and obtains the highest citation over the other techniques, indicating its wide use. The second most cited technique is the analytic hierarchy process (AHP). It is a highly regarded pairwise comparison procedure that is well-suited for intricate decision-making. It is also the most reliable technique; however, because of its complexity, slow calculations, and scalability issues, MoSCoW has become the most popular. Cumulative Voting (CV) and the Planning Game (PG) were also highly referenced, each appearing in eight studies. The next most notable techniques are cost-value ranking, pairwise analysis, and binary search trees. Other techniques, such as the kano model, value-oriented prioritization, wieger prioritization, and numerical assignment, have moderate citations with five to six references each. Quality Functional Deployment (QFD) is less frequently

TABLE 6. Most used user story requirement prioritization technique in ASD.

Techniques	Study ID
MoSCoW	S3, S7, S8, S12, S13, S14, S24, S28, S40, S69.
AHP	S7, S8, S12, S13, S17, S24, S27, S28, S29
Cumulative Voting	S3, S7, S12, S13, S17, S24, S27, S28
Kano Model	S3, S14, S17, S24, S40
Cost-Value Ranking	S3, S8, S12, S13, S24, S27, S64
The planning game	S8, S12, S13, S24, S27, S29, S64, S69
Pair Wise Analysis	S7, S12, S13, S24, S27, S28, S69
Value Oriented Prioritization	S8, S12, S13, S17, S24
Wiegers Prioritization	S8, S12, S13, S27, S69
The numerical assignment	S12, S13, S14, S28, S29, S64
Binary Search Tree	S12, S13, S24, S28, S29, S50, S69
Quality Functional Deployment	S7, S24, S69

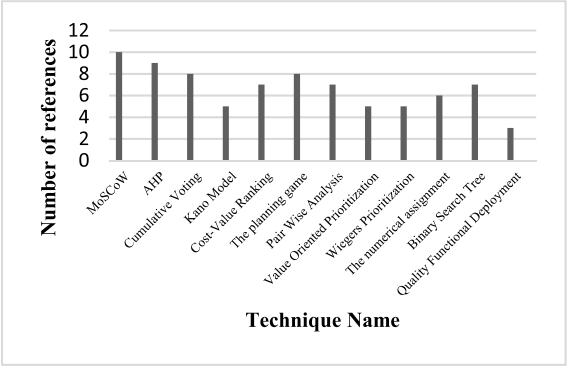

FIGURE 5. Number of references for the most used requirement prioritization techniques in ASD.

mentioned, although it is still known for its organized and scenario-specific applications. This distribution demonstrates the differing popularity and adoption of these techniques in agile prioritization.

Now, we will discuss each technique with a focus on its practical implementation, especially in agile environments, if found in the literature, the results achieved, and its benefits and drawbacks. After a brief description, a comparison of these techniques is presented in Table 7. We also attempted to demonstrate the strengths and weaknesses of the most commonly used techniques, as shown in Table 8.

1) MoSCoW

MoSCoW is widely used in agile software development approaches, such as scrum, rapid application development (RAD), and dynamic system development methods (DSDM) [47]. According to a survey, MoSCoW is the most widely used prioritization technique among agile practitioners and experts in Malaysia [28]. One study employed the MoSCoW method to prioritize important modules in ERP system implementation to increase operational efficiency and ensure the timely delivery of vital features, despite the fact that dependency management and priority conflicts are problematic [48]. While pointing out its shortcomings in terms of reacting to dynamic project changes, another study showed how it may be used in agile frameworks for organized requirements prioritizing and successfully coordinating project outputs with business objectives [49]. According to a case study, the MoSCoW method outperformed AHP for prioritization tasks in a fuzzy environment [50].

2) AHP

Several Agile environments have adopted the analytic hierarchy process (AHP) to methodically handle prioritizing issues. In global agile projects, vendor reliability is prioritized by aligning factors, such as quality and cost, thereby enhancing project outcomes [51]. Fuzzy AHP optimizes cost overhead variables for resource efficiency in dynamic agile environments [52]. It also addressed communication and dependency management challenges in distributed agile teams, improved coordination, and focused on high-value deliverables [53]. Compared to alternative prioritization techniques, AHP has been shown to yield the most trustworthy results, providing dependable frameworks for decision-making in agile projects [54]. AHP reduces bias and clarifies decisions; however, it has scalability problems and finds it difficult to adjust to rapid changes in agile projects [52], [54].

3) CUMULATIVE VOTING

In Agile, Cumulative Voting (CV) provides a simple and effective way to prioritize requirements, which makes it perfect for making decisions quickly throughout the iterations [28]. Its practical application in prioritizing criteria for a student finance system is illustrated through a case study, which highlights advantages such as speed and user-friendliness. There are still issues such as the possibility of tactical voting and the complexity of handling larger sets of requirements [55].

4) KANO MODEL

The Kano model works especially well in agile software development when customer preferences are the only thing taken into account [56]. A practical implementation involves visualizing Kano results using TreeMaps to aid understanding and decision-making [57]. Better–worse coefficients obtained from the Kano surveys were used in a different approach to measure customer satisfaction [58]. New service development is another real-world application, but it is not agile [59]. The principles of this application can be adapted to agile contexts. The Kano model successfully classifies requirements according to how they affect customer satisfaction; it may lack precise quantitative prioritization, potentially requiring integration with other methods [29], [58], [60]. Its combination with Analytic Hierarchy Process provides a useful application, resolving Kano's limitation of lacking precise quantitative prioritization [61].

5) COST-VALUE RANKING

Cost-value ranking is a good fit for agile because it maintains a balance between cost and value. When taking consumer preferences, time, and business value into account, cost value ranking is beneficial [56]. In a web development project, OurRank, a practical implementation, successfully collected and mapped benefit and cost criteria in a corporate setting using a structured five-step method to combine cost and value into requirement prioritization [62]. A review highlights how cost-value ranking addresses customer value and implementation costs [63]. Cost-value ranking balances cost and value in agile, fostering stakeholder alignment but struggles with scalability, interdependencies, and subjective estimations [56].

6) PLANNING GAME

In the context of ASD, the PG approach is mostly used in software-planning processes [64]. The Planning Game, which is an agile software development methodology, is a key practice in Extreme Programming [12]. Requirements were prioritized through sorting and ranking in controlled tests as part of a real-world application of the Planning Game (PG). The findings indicated that PG was less time-consuming and more user-friendly than pairwise comparisons; however, accuracy did not significantly differ [65]. However, it has scalability issues, and can be affected by subjective customer feedback [56], [65].

7) PAIR WISE ANALYSIS

This method focuses on comparing requirements in pairs to determine their relative importance, enabling product owners and stakeholders to make informed decisions regarding feature prioritization [60]. Pairwise comparisons were evaluated in conjunction with various prioritization techniques, including planning game and tool-supported pairwise comparisons,

in a real-world application. The findings demonstrated that tool-supported pairwise comparisons were the quickest and easiest to use, comparable to the planning game, whereas manual pairwise comparisons were the least user-friendly and time-consuming [65].

8) VALUE ORIENTED PRIORITIZATION

In agile terms, VOP is a favourable choice for practitioners because of its focus on business value and customer preferences [56]. In a case study, a small company implemented VOP, improving communication, and aligning development with business values [66]. A study comparing six prioritization techniques ranked VOP as the best based on criteria, such as accuracy and ease of use [67]. VOP is useful for coordinating development with business objectives; however, because of its potential complexity and the possibility of neglecting requirement dependencies, it may not be appropriate for larger projects [29], [66].

9) WIEGERS PRIORITIZATION

Agile's emphasis on value-effort balance is in line with Wiegers' approach, which ranks requirements according to value, cost, and risk [68], [69]. A real-world implementation in a Finnish company's transportation systems development project demonstrated its use in systematically prioritizing change requests but faced challenges such as unclear criteria and subjective inputs [70]. Using fuzzy logic, another study suggested improving these methods by translating the evaluation criteria into language variables [71]. Although the results of both examples demonstrated improved decision-making support, their shortcomings include their complexity and dependence on subjective judgement [70], [71].

10) THE NUMERICAL ASSIGNMENT

Numerical Assignment is a simple and fast requirement prioritization technique suited for medium-sized datasets [27]. In a comparative study, the Numerical Assignment Technique was evaluated using a multi-criteria decision-making approach. The Numerical Assignment Technique is simple and accessible to stakeholders with varying expertise; however, its lack of accuracy and scalability makes it less effective for complex projects [67]. An enhanced version, the Extensive Numerical Assignment (ENA), addresses uncertainty using probability distributions and grade intervals. A case study on an examination system showed improved prioritization accuracy and stakeholder satisfaction, but added complexity and relied on subjective input [72].

11) BINARY SEARCH TREE

A Binary Search Tree (BST) is more effective for managing large datasets that need to be ranked and sorted with fewer comparisons [37], [73]. A real-world implementation of BST was found in an agile-based e-governance project to prioritize software requirements by importance and cost using multi-voting. Although effective for systematic prioritization, it struggles with scalability and manages dependencies [74]. Another practical use of BST is to prioritize geographically distributed stakeholder (GDS) requirements, addressing regional needs such as power resilience, but facing challenges of complexity and bias [75].

12) QUALITY FUNCTIONAL DEPLOYMENT

QFD is particularly well suited for agile development when meeting client preferences is the main priority [56]. The practical implementation of QFD in an agile project was found in the development of the TimeTrak app. The QFD tracked functionality, optimized resource allocation, and prioritized user stories. Although controlling task correlations is still a problem, the results demonstrated improved sprint planning [76]. In banking Enterprise Resource Planning (ERP) projects, QFD is also practically used in agile RP to formulate client needs and prioritize agility enablers and success factors. Although it had time and complexity issues, it enhanced organizational and IT infrastructure prioritization [77]. Overall, QFD improves decision-making efficiency by coordinating development with customer needs; however, when used in large systems, it encounters scalability and complexity issues [56].

After understanding all the techniques mentioned in the literature, we compared them using different factors, such as scalability, complexity, and time consumption. The comparison is presented in Table 7. We also determined the strengths and weaknesses of the techniques mentioned in the literature, and summarized them in Table 8.

In Tables 7 and 8, we compare the most used requirement prioritization techniques in ASD, as well as their strengths and weaknesses. From the table, it is clear that MoSCoW is scalable and simple, making it easy to understand and implement. However, one of its limitations is that it does not grade within different requirement categories, which makes it difficult to assess the relative significance of different elements. Another popular method, AHP, produces reliable results and for being able resolves conflicting problems. However, it is time-consuming, particularly for projects with many requirements, and is not scalable. Although methods such as cumulative voting and the kano model have been praised for their speed and user-friendliness, they are criticized for their limited application scenarios and inability to manage various criteria. Additionally, techniques such as cost value ranking and wieger prioritization are noteworthy for merging cost and business value assessments; however, their limitations arise from their complexity and challenges in controlling internal dependencies.

The planning game is easy to use and fast, balancing business value, development cost, and technical risk. However, it cannot handle a large number of requirements. Value-oriented prioritization and pairwise analysis involve making more subjective decisions. While value-oriented prioritization considers business value but ignores interdependencies and is not appropriate for larger projects, pairwise prioritization is based on expert opinion, which makes it quick but inconsistent and unscalable. Quality functional deployment (QFD) and the Kano model focus on customer preference.

TABLE 7. Comparison among the most used requirement prioritization technique in ASD using different factors.

Technique	Scalability	Difficulty Level	Measurement Scale	Time Required	Focus Area
MoSCoW	Scalable	Very simple	Nominal	Faster	Business benefits
AHP	Non Scalable	Highly Complex	Ratio	Very Slow	Penalties,Strategic Importance,
Cumulative Voting	Non Scalable	Simple	Ratio	Faster	Customer importance
Kano Model	Non Scalable	Simple	N/A	Faster	Customer preferences
Cost value Ranking	Non Scalable	Moderately Complex	Ordinal	Faster	Customer importance
Planning Game	Non Scalable	Simple	Ordinal	Faster	Development cost, Business value, Technical risk
Pair Wise Analysis	Non Scalable	Moderately Complex	Ordinal	Faster	Expert judgements
Value Oriented Prioritization	Non Scalable	Moderately Complex	Ratio	Faster	Business value
Wiegers Prioritization	Scalable	Moderately Complex	Ratio	Moderate	Business value, Delay costs
Numerical Assignment	Scalable	Simple	Ordinal	Faster	Customer importance
Binary Search Tree	Scalable	Simple	Ordinal	Slow	N/A
Quality Functional Deployment	Non Scalable	Moderately Complex	Ordinal	Slow	Voice of the customer

Whereas QFD effectively captures the "voice of the customer," it is not scalable or consistent. The kano model, on the other hand, is fast and concentrates on "Trustworthiness," but it is confined to effect analysis. Numerical assignment is scalable and straightforward; it also focuses on customer importance but lacks adaptability to internal failures. The binary search tree is scalable and easy, offering quick responses even for numerous requirements. However, it only provides an ordinal ranking without assigning priority values. This informs which requirement is important, but the degree of importance cannot be obtained from the BST.

Each technique has unique strengths and weaknesses, which make it suitable for different scales and circumstances, highlighting a balance between ease of use, scalability, and depth of prioritization. A team can choose the best technique according to the requirements of a particular project by having a clear understanding of the uniqueness of these techniques.

B. RQ2) WHAT ARE THE EXISTING CHALLENGES ASSOCIATED WITH REQUIREMENT PRIORITIZATION IN AGILE SOFTWARE DEVELOPMENT?

Agile software development has a high success rate because of its advantages and potential, particularly for smaller projects. Requirement engineering (RE) is important because "Requirements" are an essential part of any software development life cycle. Agile offers advantages to customers' business demands, changing requirements, and interactions. Nevertheless, it also presents problems, many of which are tied to requirement-related issues [78]. Although a satisfactory number of studies have been conducted on agile development and adoption, there is a lack of research on the challenges encountered in RP when using agile software. After identifying this gap, we chose this research question to identify the challenges of requirement prioritization in agile. After analyzing all available literature relevant to this research question, we identified six major challenges, which are presented in Table 9 with the references where these challenges are mentioned, and Fig. 6 visualizes these challenges with the number of citations.

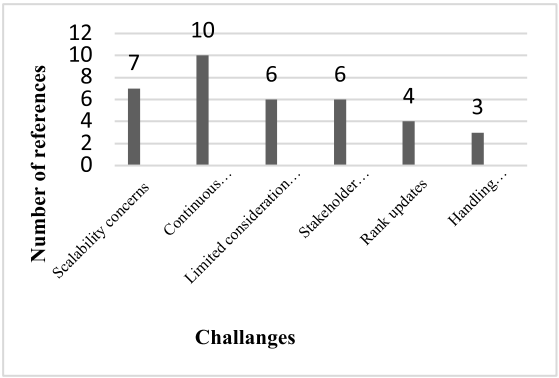

FIGURE 6. Visual representation of challenges associated with requirement prioritization in ASD.

The given challenges are the most mentioned in different studies on agile RP. From Table 9 and Figure 6, we can

TABLE 8. Strength and weakness of the most used requirement prioritization technique in ASD.

Technique	Strength	Weakness
MoSCoW	It is reliable, easier to understand, and requires less effort. It can handle a lot of various circumstances, easily adaptable and also able to accommodate a wide range of requirements	Lack of grading within different categories is the main issue. It is challenging to determine which COULD or SHOULD requirements be more crucial than others. Better suited for a product with a lesser customer base
AHP	Capability of resolving conflicting objectives and offer a trustworthy outcome	Time-consuming when there are more requirements and lack of scalability makes larger projects challenging
Cumulative Voting	Easy to use	Not suitable for large number of requirements and cannot evaluate how different the requirements are in terms of relative priority.
Kano Model	Kano is more focused on what the client requires in terms of "Trustworthiness." Kano model is the fastest approach to prioritize requirements	It is limited to application in effect analysis. It isn't for making suggestions for new product features, which can be challenging to implement
Cost Value Ranking	Ability to integrate the evaluations of the requirements' cost and value for implementation	Managing interdependencies becomes more computationally complex as the number of requirements rises and also not scalable. Lack of flexibility and inability to maintain communication amongst many partners in order to maintain coordination
Planning Game	It is a better modification of numerical computation. It is easy to use and consume less time to prioritization process	Cannot handle large number of requirements
Pair Wise Analysis	Expert judgement is necessary to match the criteria. The comparison criteria for the options can stay informal, allowing participants' experiences to serve as the foundation for decisions	It is possible to unpair particular requirements. Time-consuming, not scalable, complex, and provide inconsistent outcomes
Value Oriented Prioritization	The prioritization process takes the organization's business value into consideration	Ignores the dependence of requirements ,unfit for larger projects. The computation method ignores necessary conditions
Wiegers prioritization	It offers a comprehensive approach aligned with business value	Complex to use and overlook internal dependencies
Numerical Assignment	Easy to use, minimal complexity combined with quick speed .Can handle requirements of large scale	Low level of adaptation to internal failure
Binary Search Tree	BST could respond quickly even while handling lots of requirements.	BST just ranks requirements in a straightforward manner without assigning any priority values. BST indicates which requirement is more beneficial, but it is impossible to determine how essential that requirement is, therefore the comparison is just ordinal
Quality Functional Deployment	QFD is an organised approach for gathering customer demands and known as "voice of the customer"	Mostly used in small systems, limited in terms of scalability and consistency

see that the most cited challenges are continuous requirement changes. The change-accepting nature of agile makes it difficult to maintain a stable prioritization list, disturbing the development process, and the second most cited challenge is scalability. However, most existing techniques suffer from these problems when dealing with an increasing number of requirements. The limited consideration of non-functional requirements (NFR) and stakeholder collaboration and communication are the next most mentioned challenges. Non-functional requirements frequently receive less attention than functional requirements, lowering the overall software quality, and misaligned stakeholder perspectives can lead to miscommunication and disagreements. Rank updates are the next challenge that needs to be addressed in ASD requirement prioritization. Priorities in the Agile must be regularly updated to reflect evolving project objectives and stakeholder demands. Agile requirement prioritization also involves managing dependencies and interdependencies

TABLE 9. Challenges associated with requirement prioritization in ASD.

Challenges	Study ID
Scalability Concerns	S15, S17, S24, S25, S27, S41, S50
Continuous Requirements Change	S15, S17, S24, S27, S30, S41, S44, S45, S51, S67
Limited Consideration of Non-Functional Requirements (NFR)	S1, S8, S9, S17, S41, S45
Stakeholder Collaboration and Communication	S17, S31, S41, S45, S50, S52
Rank Updates	S15, S46, S50, S56
Handling Dependencies and Interdependencies	S15, S17, S57

among requirements, as changes to one can significantly impact others.

For better understanding these challenges are discussed in following subsection.

1) SCALABILITY CONCERNS

Scalability problems are caused by an inability to handle a large number of requirements. Scalability is currently one of the primary issues in most RP approaches [60]. Scalability is an important consideration in the prioritization process, because most industrial projects now have a large number of requirements [60], [79]. A study on 108 existing Requirement Prioritization (RP) techniques revealed that **93%** of these techniques poorly handle the scalability issue, while only **7%** are successful in addressing it effectively [79]. Numerous factors, such as the volume of comparisons, absence of automation, and total reliance on human labor, can lead to scalability issues [37], [60], [79]. In the majority of the currently available techniques, including bubble sorting, AHP, and pairwise comparison techniques, one of the typical methods for conducting the RP process is a comparison of the relative priority between pairs of requirements, resulting in a total of n(n-1)/2 comparisons [80]. As requirements grow, the workload increases exponentially, making prioritization inefficient and prone to human error [60]. The majority of RP techniques currently in use rely on human intervention to prioritize the process. This dependency not only makes the process time-consuming but also risks inaccuracies, especially in large-scale projects [36]. Without automation and intelligence, the current RP approaches struggle to handle hundreds or thousands of criteria effectively.

2) CONTINIOUS REQUIREMENT CHANGE

In Agile, requirements often change or arise unexpectedly, making adaptability essential. Frequent changes can complicate testing, delay iterations, and extend deadlines even for experienced teams [32], [81], [82]. From Table 9, we can clearly see that the most mentioned challenge in ASD is handling requirement changes. The literature only partially addresses requirement changes in agile development, emphasizing the importance of academic and industry collaboration [83]. Existing prioritization techniques may not fully capture the agility required to accommodate frequent changes in requirements and priorities throughout the project lifecycle. The literature lacks prioritization methods that can handle continuous requirement changes. Case studies conducted with software development organizations highlight a challenging scenario within the software industry. These findings underscore a situation in which a limited number of individuals within an industry comprehend the significance of changing priorities [84]. However, there is a lack of established techniques and processes for effectively managing and adapting to these changing priorities. Dealing with these constant changes requires an ongoing and adaptive approach to ensure that the project remains aligned with the evolving priorities.

3) LIMITED CONSIDERATION OF NON-FUNCTIONAL REQUIREMENTS(NFR)

Non-functional requirements (NFRs) are frequently noted as a challenge in agile projects, as agile techniques tend to prioritize the incremental delivery of functional features while ignoring NFRs. However, little research has been conducted, especially on requirements engineering approaches in agile development that handle NFRs [85].

Non-functional requirements (NFRs) are frequently overlooked in agile development because of the emphasis on providing functional features in fewer iterations. Both developers and customers value functionality over NFRs, and Agile practitioners frequently underestimate their significance. Ignoring NFRs from the beginning can result in errors, project failures, and increased costs in terms of time, resources, and maintenance [86].

Although NFR-related issues are often discussed, the findings indicate that practitioners acknowledge the value of NFRs in agile projects and that the success rate of agile projects is higher than that of plan-driven initiatives. This suggests that practitioners in real-world agile projects should set up appropriate agile procedures that allow them to successfully capture NFRs and subsequently produce software that satisfies these requirements [87].

4) STAKEHOLDER COLLABORATION AND COMMUNICATION

Prioritizing user stories in agile development necessitates thorough stakeholder analysis to determine their impact, significance, and overall value. Agile teams use in-person informal communication to align goals, share experiences, and plan actions. Unlike traditional development, agile promotes continuous communication and interaction between stakeholders and the development team. Effective stakeholder selection is essential for prioritizing requirements, ensuring that they understand customer desires, possess agile knowledge, and engage in open communication and

collaboration [64]. Most prioritization strategies fail to enable effective communication and collaboration among stakeholders, which is essential for goal alignment and proper requirement prioritization in agile development [60].

Agile requirement prioritization poses significant challenges for stakeholder collaboration and communication because different stakeholders bring different and sometimes contradictory opinions to the discussion. This may result in costly delays, overspending, or unsatisfactory project completion. However, in the real world of project work, involving different stakeholders, conflicts are bound to arise occasionally [88]. Resolving these challenges is essential to aligning stakeholder expectations and achieving successful project outcomes.

5) RANK UPDATES

A rank update refers to the continuous prioritization capability of a technique, allowing it to automatically adjust ranks whenever a new requirement is added or an existing requirement is removed from the list. [89]. A change in requirements is the cause of this situation. Consequently, at any time a requirement is added to or removed from the rank list, the prioritization methods that are currently in use are unable to update or reflect the rank status. This is crucial because the procedures for selection and decision making cannot function without iterations. Therefore, a prioritization approach that allows rank updates is required. This limitation appears to have affected most of the techniques currently used.

6) HANDLING DEPENDENCIES AND INTERDEPENDENCIES

Requirement interdependencies are significant challenges in agile requirement prioritization. Dependencies are formed when one requirement has an impact on another, resulting in sequential linkages that must be handled carefully. Interdependencies involve several interconnected requirements, adding complexity, because most requirements cannot be addressed alone and have a considerable influence on one another [90]. Requirement interdependencies have an impact on software development because they influence decisions and actions, and perhaps generate unexpected consequences for other requirements. This can lead to a considerable increase in implementation costs [91], [92].

Agile software development often overlooks the management of requirement interdependencies, primarily because of project simplicity. However, as projects grow and their requirements are executed in stages, resolving interdependencies becomes increasingly important. Ignoring them threatens a project's success and imposes unexpected expenses. Current Agile approaches do not provide significant assistance for managing interdependencies, underlining the need for additional studies to enhance this area [93].

The above mentioned challenges in requirement prioritization can reduce the effectiveness and efficiency of agile software development. Larger or distributed teams find it more difficult to manage requirements owing to scalability issues, and ongoing requirement changes necessitate an adaptable and flexible technique for prioritizing requirements. A lack of attention has been paid to non-functional requirements that reduce the quality of software products. For continued success, strong stakeholder collaboration and communication are essential but difficult to maintain. Adjusting the priority rankings in response to project modifications and managing interdependencies and dependencies introduces additional layers of complexity. Researchers may focus on improving the requirement prioritization techniques of agile software development by addressing these challenges, which will ultimately result in better software products and more successful project results for ASD.

C. RQ3) WHAT ARE THE AI-BASED APPROACHES IN REQUIREMENT PRIORITIZATION?

AI is already in use across a wide range of industries, most notably computer engineering. AI approaches have been increasingly used in RE in the past few years to automate and enhance the requirements engineering process [94]. The main advantages of AI techniques are that they expedite decision-making, solve complex problems, improve overall performance, automate a wide range of tasks, and provide value prediction by creating models based on past data [31].

Artificial Intelligence (AI) techniques have experienced exponential development in their utilization in recent years. Requirement prioritization has been revolutionized by artificial intelligence (AI). Current techniques for prioritizing requirements involve considerable human effort and have several limitations, including overlapping results and issues with scalability, time consumption, and inaccuracy. Some of these issues can be resolved using artificial intelligence techniques and algorithms [2]. Several AI-based techniques for prioritizing requirements have been developed using fuzzy logic, machine learning, and optimization-based algorithms [2], [3], [31].

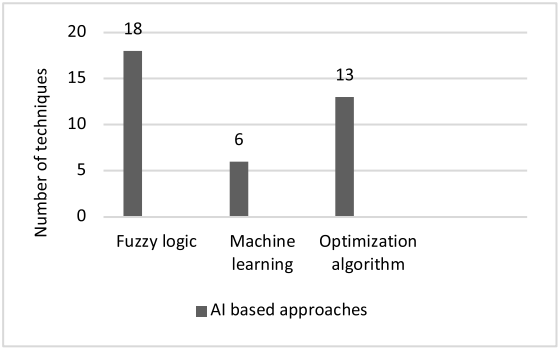

FIGURE 7. AI based approaches in requirement prioritization.

In this research question, we explored the available AI-based approaches for RP. We divide these approaches into three categories. These approaches are based on fuzzy logic, RP techniques that use machine learning, and optimization-based approaches. The following subsection

explores all the available techniques under these categories. The number of approaches for each category is shown in Fig. 7.

After applying our search strategies to determine the requirement prioritization techniques under these three categories, we found 37 primary studies. Among them, 18 techniques are based on fuzzy logic,6 techniques are based on machine learning, and the remaining 13 techniques are based on optimization algorithms.

1) FUZZY LOGIC-BASED APPROACHES

Fuzzy logic was first proposed by Zadeh in 1965 [95]. Fuzzy logic is a scientific method to deal with uncertainty by converting confusing concepts into mathematical solutions. It solves ambiguous problems and enables exact definitions of phrases such as low cost, high quality, and high progress [96]. Fuzzy logic simplifies the control theory and addresses complex real-world problems using computational intelligence. The application of fuzzy logic in software engineering spans effort estimation, project similarity, development, evaluation, evolution, and requirement engineering [97].

The techniques we find based on fuzzy based approaches are represent in Table 10.

In the following subsections, we summarize each technique after analyzing the primary studies mentioned in Table 10.

a: URPCalc[S18]

URPCalc is a requirement prioritization technique that uses fuzzy approaches for time series forecasting. This method can assist product managers in choosing requirements that will be important to users in the future for upcoming product releases. It incorporates data from several sources, including helpdesk systems and user reviews, to predict changes in requirement relevance, reduce workload, and improve customer satisfaction. Although it reduces human mistakes and constantly adjusts to new data, its dependency on input quality and complicated external system integration are notable limitations.

b: VOLATILE REQUIREMENT PRIORITIZATION: A FUZZY-BASED APPROACH [S19]

This article suggests a fuzzy-based requirement prioritisation strategy for managing changing requirements across the software development lifecycle. The authors defined that detectability, dependability, and changeability were the three most important criteria for rating the volatile requirements. The approach, which was implemented in MATLAB, was evaluated using a case study of a web-based MIS project, showing that it can handle frequent changes in requirements more accurately and adaptably than the conventional methods. This strategy has the advantage of being adaptable when dealing with volatility and has a systematic approach to handling changing requirements. However, its execution is

TABLE 10. Fuzzy based approaches for requirement prioritization.

Study ID	Technique Name	Description
S18	URPCalc	The proposed techniques is designed for automating the user requirements prioritization and uses fuzzy logic
S19	Volatile Requirement Prioritization: A Fuzzy Based Approach	This study develop a method for prioritizing requirements while considering their changeability utilizing fuzzy approach
S20	Prioritization And Partial Selection (PAPS)	A fuzzy system to improve the accuracy of prioritization and selection while reducing the negative effects of disregarding security requirements
S32	A Fuzzy Set-Based Approach for the Prioritization of Stakeholders	The goal of this proposed approach is to prioritize the stakeholders based on the importance of software requirements using fuzzy set based approach.
S33	Fuzzy_MoSCoW	A software requirement prioritization method using fuzzy based MoSCoW .
S34	Adaptive Fuzzy Hierarchical Cumulative Voting	The objective of Adaptive Fuzzy Hierarchical Cumulative Voting (AFHCV) is to expand the coverage of possible events at runtime by integrating an adaptive mechanism with the current Fuzzy Hierarchical Cumulative Voting (FHCV) technique
S35	Fuzzy Approach to Prioritize Usability Requirements Conflicts	This study offers an original framework that emphasises using fuzzy logic to map the qualities of the usability requirements to the linguistic assessment provided by the users
S42	Approach to Prioritize the Requirements Using Fuzzy Logic	The study uses fuzzy rules to give an entirely new multilevel quality based insightful prerequisites requirement elicitation technique.
S46	Fuzzy multi-criteria decision making approach	The fuzzy multicriteria decision-making (FMCDM) approach is used in this study to provide an improved method for prioritising requirements for software projects with stakeholders.

TABLE 10. *(Continued.)* **Fuzzy based approaches for requirement prioritization.**

S47	PHandler	Priority Handler (PHandler) is an expert system that is based on the value-based intelligent requirement prioritization technique, neural network and analytical hierarchical process
S48	An Approach for Prioritizing NFRs According to Their Relationship with FRs	The objective of this research is to propose a quantitative method for performing the process of prioritising non-functional requirements using alpha cut approach as well as fuzzy logic
S53	Fuzzy preference relation for requirements prioritization	An approach to the PRFGORE process is presented in this study. To make choices when the available data is fuzzy and imprecise, and to improve the goal-oriented requirements elicitation process
S54	A Neuro-Fuzzy based Approach	A neuro-fuzzy based approach is proposed in this study for software requirement prioritization
S58	Multi-person decision-making using Fuzzy AHP	A novel approach is proposed to prioritize the conflicting requirements using Fuzzy AHP and an alpha cut
S60	A Simulation-based Fuzzy Multi-attribute Decision Making approach	This study is proposed a novel approach for prioritising software requirements that is based on formulating a fuzzy multiple-attribute decision-making issue
S61	Software requirement prioritization using Fuzzy Multi-attribute Decision Making	The article introduces a novel method for arranging criteria in order of importance using fuzzy numbers that represent an attributes predicted value
S63	Value Based Intelligent Requirement Prioritization (VIRP)	A novel value based intelligent requirement prioritization techniques is proposed using fuzzy logic.
S68	Value Based Fuzzy Requirement Prioritization	This study propose an intelligent fuzzy logic based technique as well as a framework for evaluation.

highly dependent on the quality of the input data, precise fuzzy membership functions, and proficiency of tools such as MATLAB.

c: PRIORITIZATION AND PARTIAL SELECTION (PAPS) [S20]

This study proposes a fuzzy framework called Prioritization And Partial Selection to address the challenge of resource limitations in software projects that prevents the full implementation of security requirements. PAPS optimizes requirement prioritization and selection by utilizing a fuzzy inference method that assesses impact, cost, and technical feasibility. It was tested on an online banking system and showed better prioritization with linear complexity and scalability. While its shortcomings include the complex nature of implementing fuzzy systems and the possibility of subjectivity while establishing fuzzy rules, its strengths include improving the accuracy and lowering overlooked security requirements.

d: A FUZZY SET-BASED APPROACHES FOR THE PRIORITIZATION OF STAKEHOLDERS [S32]

This study proposes a technique that uses a fuzzy-based mechanism to prioritize stakeholders according to the importance of requirements. Its goal is to identify and prioritize stakeholders whose requirements should be considered for various software releases. Stakeholder analysis using the extent fuzzy AHP, requirement classification, and stakeholder role specification are the three steps in this process. Its usefulness and efficacy in stakeholder prioritization were illustrated through a case study of an Institute Examination System (IES). Better decision making and support for inaccurate fuzzy data are two advantages of this method. Larger projects can encounter scalability issues because they depend on expert judgement to define the fuzzy inputs.

e: FUZZY_MoSCoW [S33]

The authors expanded the MoSCoW approach to fuzzy environments. The objective of this study is to introduce a fuzzy-based MoSCoW method for prioritizing software requirements. Identifying requirements, establishing triangle fuzzy numbers, gathering fuzzy assessments from decision-makers, using graded mean integration, and calculating requirement rankings are the five steps in this process. The technique was tested on a Library Management System (LMS) and successfully prioritized both functional and non-functional factors. Its strength lies in its ability to handle uncertainty and improve prioritization accuracy; however, its reliance on expert input for fuzzy definitions and computational complexity are potential limitations.

f: ADAPTIVE FUZZY HIERARCHICAL CUMULATIVE VOTING (AFHCV) [S34]

The AFHCV approach offers a way to add requirements to the current dataset or modify the priority of requirements at runtime. The Fuzzy Hierarchical Cumulative Voting(FHCV) is not appropriate for requirements that require a runtime. Considering the dynamic nature of stakeholders, adaptivity is

necessary. The proposed AFHCV technique offers an effective means of appending new requirements to the current requirement set and permits the modification of the priority of existing requirements during runtime. By reprioritizing this process, the AFHCV also enhanced the FHCV results. Reprioritization is applied to the final requirement set or when a new requirement set is added. The experimental results demonstrate that the AFHCV is more successful than the FHCV in adding additional requirements at runtime and produces prioritization results that are closer to the actual priority. Because each stakeholder's requirement is manually assigned a priority, the AFHCV is limited in terms of the amount of time it can consume. The requirements were prioritized by the AFHCV using the FHCV.

g: FUZZY APPROACH TO PRIORITIZE USABILITY REQUIREMENTS CONFLICTS [S35]

This study proposes a methodology for prioritizing usability objectives that conflict using fuzzy logic. Finding and prioritizing usability attribute conflicts is the goal of facilitating improved decision-making. This method quantifies disputes using triangular fuzzy numbers, fuzzy logic, and Mamdani's method. When used in a scenario including an Electronic Healthcare System (EHCS), it was successful in recognizing and ranking conflicts. The advantages of the framework include handling inaccurate, fuzzy data, automation, and a reduction in human labor. Although some processes are automated by the framework, the vast number of rules and expert inputs can make computations more complex and time-consuming, especially in large-scale systems.

h: APPROACH TO PRIORITIZE THE REQUIREMENTS USING FUZZY LOGIC [S42]

In this study, the authors concentrated on the requirement completeness and predictability of understandability during the RE phase of the software development process. Fuzzy logic serves as the multilevel foundation for this innovative requirement prioritization technique. This method predicts the completeness and understandability of requirements using stakeholders, experts, and a fuzzy logic-based system. Their results demonstrated improved requirement prioritization in various environments. The manner in which they manage the inherent unpredictability of software development is one of their main advantages. However, a possible drawback is the dependence on subjective expert assessments. Prioritize requirements and differentiate between criteria that are negotiable and those that are not two tasks for future research.

i: FUZZY MULTI-CRITERIA DECISION-MAKING APPROACH [S46]

The FMCDM technique was used in this study. Several models and formulas were developed to improve the practicality of the proposed method. The proposed methodology was evaluated using relevant project datasets. Significant limitations in the prioritization techniques currently in use have been overcome by using the suggested strategy. Overall, the proposed method performed better in terms of the above metrics. Software engineers can use this information to distinguish between the requirements of the relevant stakeholders, which are the most and least valued. This will help them plan for software release and prevent agreements, contracts, or trust breaches during the software development process. They intend to create a concurrent hybridization of evolutionary algorithms and FMCDM in the future to address the requirements prioritization issue.

j: PHandler [S47]

The proposed PHandler system is a highly knowledge-based system with a hybridized product as the end result. The proposed PHandler system combines the strengths of the three primary techniques: AHP, BPNN, and VIRP. A BPNN was utilized to determine the software requirement value for prioritization. Because expert judgements are biased, the process of prioritizing software requirements is highly non-linear. As a result, the RV is forecasted using the BPNN. Finally, AHP was performed to eliminate any ambiguity. Exceptions are made to prioritize the requirements for which the RV is the same. Large RV-based clusters of requirements can be divided into smaller sub-clusters by the exceptions, and AHP is used to rank the requirements of the smaller sub-clusters. The findings demonstrate that the proposed PHandler expert decision support system effectively addresses the issue of requirement scalability and solves the prioritization problem in projects with numerous software requirements.

k: AN APPROACH FOR PRIORITIZING NFRS ACCORDING TO THEIR RELATIONSHIP WITH FRS [S48]

This study tackles the challenge of managing non-functional requirements (NFRs) in software development. The authors propose a systematic approach that accounts for the relationship between NFRs and functional requirements (FRs). Their methodology involves eliciting both FRs and NFRs, establishing their linkages in a decision matrix, and employing techniques such as Triangular Fuzzy Numbers and Alpha Cuts to aggregate the value of each NFR in relation to all FRs. This leads to the creation of a prioritized list of NFRs, allowing developers to focus on those that are most crucial to a project's success.

l: FUZZY PREFERENCE RELATION FOR REQUIREMENTS PRIORITIZATION [S53]

In this article, the author describes a method for prioritizing requirements in a goal-oriented requirements elicitation process by combining an extended fuzzy analytic hierarchy process with an a-level weighted F-preference relationship in a group decision-making process. A prioritized list of requirements is then obtained using the binary tree sorting method. Lastly, they implement the suggested technique for the Institute Examination System's (IES)RP. However, by considering a far wider number of stakeholders in the group decision-making process as well as more sub-goals, requirements, and criteria, the approach covered in this study

can be further optimized. In the future, they plan to propose a fuzzy attributed goal-oriented requirements elicitation and analysis (FAGOREA) approach by utilizing fuzzy contribution values and a fuzzy preference matrix, and creating a tool that will aid FAGOREA's decision-making procedures.

m: A NEURO-FUZZY-BASED APPROACH [S54]

This study suggests a revolutionary method based on a neuro-fuzzy system that can deliver the ideal combination of high-quality components required to produce products that satisfy customers. The goal of this research was to prioritize software requirements. They used a fuzzy membership function in a neuro fuzzy system. This type of matrix establishes a link between the quality attributes and client requests. The proposed approach exemplifies quality criteria, such as speed, task completion time, readability, traceability, correctness, and compatibility. The results of the study indicated that the network computation and output production times were approximately 8% faster than those of the QFD ranking approach. Scientists believe that a neuro-fuzzy system can yield more accurate responses than those used in earlier research.

n: MULTI-PERSON DECISION-MAKING USING FUZZY AHP [S58]

This study suggests a straightforward and manageable prioritization technique for multi-criteria, multi-person decision making in a fuzzy setting. FAHP is used when stakeholder requirements are ambiguous or unclear. Stakeholder weights were determined using eigenvalues. Stakeholder expectations are reflected in the integrated matrix; however, systemic constraints can prevent all stakeholders from being completely satisfied. The final consensus is complete ordering, which is achieved by prioritizing criteria using an alpha cut on the matrix. Stakeholders agreed with the technique when it was attempted on a website for travel management planning. Although it is computationally complex and requires expert input for fuzzy rule formulations, its key benefits include handling many stakeholders, managing uncertainty, and producing a unified requirement list.

o: A SIMULATION-BASED FUZZY MULTI-ATTRIBUTE DECISION MAKING APPROACH [S60]

The novel method of prioritizing the requirements presented in this study uses fuzzy numbers to rate the importance of each attribute in a requirement based on its predicted value. To determine this ranking, it is necessary to approximate these expected values through simulations by selecting random values from the credibility distributions of these numbers. This method combines several criteria and aspects to provide a broad picture of quality standards. The method was validated using a case study, which demonstrated that it could effectively prioritize requirements. Among its many benefits are its ability to manage imprecise data, incorporate several attributes with ease, and be readily modified for decision making. However, it requires a significant amount of computational work for the simulation and is dependent on the quality of the fuzzy input definitions.

p: SOFTWARE REQUIREMENTS PRIORITIZATION FUZZY MULTI-ATTRIBUTE DECISION MAKING APPROACH [S61]

In this study, the requirements were prioritized as a fuzzy multi-attribute decision problem and the options in the issue plan were ranked using the anticipated value operator. The proposed method can handle erroneous data by displaying different quality attributes as grayscale values. This method is easy to use and can be applied to analyze hypothetical situations in which different traits are assigned different weights. The proposed method is particularly helpful because of its adaptability, low cost, speed, and simplicity of usage. Nevertheless, we were unable to find experimental support for the methodology proposed in this study. Researchers believe that case studies will be used in the future to verify this method.

q: VALUE-BASED INTELLIGENT REQUIREMENT PRIORITIZATION(VIRP) [S63]

In this study, the author introduced VIRP, a multilevel fuzzy logic intelligent requirement prioritization method. VIRP uses fuzzy logic to rank requirements and improve accuracy and consistency, while addressing the shortcomings of earlier techniques. This approach quantifies the need for more precise and complex prioritization. Extensive testing showed that the VIRP delivered better results and consistency. The study's findings highlight VIRP as an adaptable and reliable method for software development teams, ensuring that crucial requirements are met and software products are produced more successfully.

r: VALUE-BASED FUZZY REQUIREMENT PRIORITIZATION [S68]

To address the imprecise nature of software requirements, this study proposes a value-based fuzzy requirement prioritization (VFRP) technique. The goal is to overcome the drawbacks of the current prioritization techniques, such as their lack of automation and dependence on expert judgement. The proposed method enhances accuracy and reduces human bias by automating prioritization with fuzzy logic. The effectiveness of this technique was evaluated and compared with that of other methods using case studies from academic initiatives. Better prioritization accuracy and less dependence on expert opinions were demonstrated by the outcomes. However, it faces challenges such as dependence on fuzzy input definitions, complexity of the fuzzy system's rule set, and need for expert-defined membership functions. Future research entails creating a fuzzy logic-based system that is fully operational and can be applied to industrial requirement prioritization.

After analyzing all these fuzzy-based approaches, we identify the strengths and weaknesses of each fuzzy-based strategy based on the literature pertaining to that technique. For a clear comparison to direct future studies and applications in

the field, we describe the strengths and weaknesses of each technique in Table 11.

Table 11 summarizes the strengths and weaknesses of several fuzzy-based requirement prioritisation techniques. Their strengths include the ability to deal with ambiguous data, adapt to changing requirements, improve decision-making, and enhance prioritization accuracy. However, prominent disadvantages include computational complexity, reliance on expert input, limited scalability, and insufficient empirical testing in real-world circumstances. These strategies show promise in solving certain prioritization difficulties; however, more research is needed to evaluate their practical application, increase scalability, and simplify their widespread adoption.

2) MACHINE LEARNING-BASED APPROACHES

Machine learning has received less attention in the field of requirement prioritization than other AI-based technologies. A systematic literature review found that, out of 102 analyzed papers, only four recent publications applied machine learning techniques for requirement prioritization [5]. However, owing to the potential advantages of utilizing machine-learning techniques, this trend is rapidly expanding [5], [98]. Traditional methods of requirement prioritization may be time consuming because of the potentially overwhelming number of patterns to understand and apply. Machine learning has been used in many domains to analyze and identify patterns in large datasets. After training to recognize patterns in the data, it can build a classification model or estimate them. Similar patterns or probabilities can be recognized, anticipated, or detected by trained models [99]. In Table 12, we present the available machine learning-based requirement prioritization techniques with references and short descriptions.

After analyzing the primary studies mentioned in Table 12, each technique is summarized in the following subsections.

a: SOFTWARE REQUIREMENT PRIORITIZATION USING MACHINE LEARNING [S2]

The purpose of this study is to examine how software requirements for complicated software release planning can be prioritized using machine-learning models. They emphasize software product lines (SPL) and multiple business lines (MBL). It tackles the problem of handling conflicting stakeholder priorities, coupled with more feature requests that the development team can handle. The method involves analyzing data from a semiconductor company's software release process, applying ML models such as decision trees, random forests, and SVM, and evaluating them using metrics such as accuracy, F1 score, and cross-validation. According to their findings, the study concluded that machine learning models can accurately predict software requirements, which could improve the process of software release planning requirement prioritization. The best outcomes were obtained using random forest, K-nearest neighbours, and decision tree

TABLE 11. Strength and weakness of fuzzy based approaches for requirement prioritization.

Study ID	Technique name	Strength	Weakness
S18	URPCalc	The URPCalc systems can automate the prioritization process, adjust to evolving user needs, minimise human error and ensure consistency in decision-making	Key limitations include limited user involvement, complex implementation, scalability, and narrow evaluation.
S19	Volatile Requirement Prioritization: A Fuzzy Based Approach	Can handle volatile requirements	Dependence on expert knowledge, computational complexity, and difficulty in validation.
S20	Prioritization And Partial Selection (PAPS)	PAPS effectively reduces ignored security requirements and increases the accuracy of prioritization .	Their scalability is not verified in practice, less reliable.
S32	A Fuzzy Set-Based Approach for the Prioritization of Stakeholders	The approach prioritizes stakeholders and handles fuzzy data to improve decisions.	A limitation of the study is that it has just utilised the suggested approach on ten stakeholders
S33	Fuzzy_MoSCoW	Making the MoSCoW methods more efficient	Tested on small dataset. Less reliable
S34	Adaptive Fuzzy Hierarchical Cumulative Voting	This approach can deal with ambiguity, complexity and uncertainty	High time consumption is the main weakness of this technique
S35	Fuzzy Approach to Prioritize Usability Requirements Conflicts	More flexible to identify the requirements conflicts. Within short time give an optimal result	The framework must be used with a big list of requirements because it is tested with a limited set of criteria.

TABLE 11. *(Continued.)* **Strength and weakness of fuzzy based approaches for requirement prioritization.**

S42	Approach to Prioritize the Requirements Using Fuzzy Logic	It produce better results in different environments	For validation comparative analysis is not performed
S46	Fuzzy multi-criteria decision making approach	The process is clearly outlined, from requirement generation to final ranking, making it easy to understand	Implementing the proposed approach might require significant computational resources, especially for large datasets.
S47	Phandler	Efficiently handles the problem of requirement scalability and resolves prioritization issue.	Main limitation is associated with the utilisation of highly skilled business analysts
S48	An Approach for Prioritizing NFRs According to Their Relationship with FRs	It is create a linkage between FRs and NFRs to perform prioritization. It is easy to use and trustworthy	Its effectiveness is not evaluated in real word .
S53	Fuzzy preference relation for requirements prioritization	It is useful in situations where information is ambiguous and helps to prioritize requirements during the decision-making process.	An example is provided, but no empirical testing is done.
S54	A Neuro-Fuzzy based Approach	Lower cost and less time consumption comparing with the AHP, ranking, top-ten, PG, W-theory, VIRP techniques	The approach accuracy is not being tested
S58	Multi-person decision-making using Fuzzy AHP	This approach effectively tackles the issue of conflicting requirements	This approach's usefulness is limited to high ambiguity and fuzzy scenarios; it may be less effective in projects where stakeholder preferences are not significantly uncertain or vague.

TABLE 11. *(Continued.)* **Strength and weakness of fuzzy based approaches for requirement prioritization.**

S60	A Simulation-based Fuzzy Multi-attribute Decision Making approach	This methodology can manage imprecise data and can be easily customized for decision making	Its efficacy needs more empirical confirmation and its reliance on intricate fuzzy logic and simulation may limit accessibility.
S61	Software requirement prioritization using Fuzzy Multi-attribute Decision Making	It is easier to use, less expensive, and time-efficient	There was no case study conducted to compare the outcomes in actual scenarios
S63	Value Based Intelligent Requirement Prioritization (VIRP)	Over 90% of the requirements should be prioritized accurately.	It is more expensive than conventional techniques
S68	Value Based Fuzzy Requirement Prioritization	The strength of this approach is its adaptability and precision in handling imprecise requirements through fuzzy variables	Its weakness lies in the computational complexity and potential difficulty in implementation

classifiers. This study suggests that hyperparameter modification should be performed in future studies to investigate other models and assess the results.

b: A MACHINE LEARNING APPROACH TO WORKFLOW PRIORITIZATION [S21]

This study attempted to exploit operational metadata at the lowest level of information extraction given the wide potential of prioritizing a highly granular workflow. They attempted to maintain parsimony in creating a model that could offer useful insights into workflow optimization rather than parsing the documents themselves. The chosen model was trained with a logistic output and gradient decision tree boosting to determine the likelihood of task success. They successfully classified tasks that resulted in changes to any of their clients' enormous datasets by integrating multiple elements that had not been used previously. They found that XGBoost performed better than all other models, indicating that their choice was sound given the true-positive classification performance criterion. They established a priority score at a greater degree of granularity, where the deployment of a rating system was more realistically helpful for the customer

TABLE 12. Machine learning based requirement prioritization approaches.

Study ID	Technique name	Description
S2	Software Requirements Prioritization Using Machine Learning	The research aims to use machine learning models to understand the factors affecting the inclusion of software requirements in a release plan.
S21	A Machine Learning Approach to workflow Prioritization	The authors of this study aimed to maintain simplicity while creating a model that may offer actionable insight towards workflow optimization
S36	Framework to enhance ERP Usability	The article suggests an IS success framework that includes machine learning-based RP methodologies in order to improve customer satisfaction and ensure that an ERP project is successful.
S49	A hybridized approach based on K-means and evolutionary algorithms	This study proposes a hybrid software requirement prioritization approach using the K-means algorithm, emphasizing strong relationships between key requirements and stakeholders' language assessments.
S71	Facing scalability issues with machine learning	They use a case-based framework, known as Case-Based Ranking, to prioritize requirements by leveraging past data and context, addressing scalability challenges with machine learning techniques to handle large datasets and complex scenarios more efficiently
S73	Case-Based ranking for Decision Support System	In this study the author offer a solution, based on machine learning techniques. They provide examples of how a boosting algorithm can efficiently estimate paired preferences and lessen the elicitation process's effort.

in scheduling, given the 98% F1 score acquired for the prediction at this level. The entire model must be retrained if major adjustments are to be made in a single department. The model was trained using consumer financial domain data from 2018 to extend the results to other domains.

c: FRAMEWORK TO ENHANCE ERP USABILITY [S36]

The article proposes an IS success framework that includes machine learning-based requirements prioritization methodologies in order to improve customer satisfaction and ensure that an ERP project is successful. The framework, with some modifications, is based on the D&M IS Success Model. The framework uses user interaction, requirement elicitation, and machine-learning approaches to rank requirements efficiently and reliably. The proposed approach aims to save time and effort, while maintaining user satisfaction. Strengths include automation, scalability, and improved decision-making, whereas disadvantages include a lack of empirical validation and the difficulty of applying machine learning to large-scale ERP systems.

d: A HYBRIDIZED APPROACH BASED ON K-MEANS AND EVOLUTIONARY ALGORITHMS [S49]

In this study, a hybrid algorithm based on stakeholder linguistic assessments is developed to determine the required preference weights. This technique was validated by using a criterion with relative stakeholder weights from the RALIC dataset. The goal of this study was to offer a more efficient method for scalability, computing complexity, and ranking reversals. Clustering- and evolutionary-based techniques were used in this study. By reducing conflicts among prioritized requirements, the proposed method can also effectively classify large numbers of requirements by computing the maximum, minimum, and mean scores. Based on these results, this study can be viewed as a breakthrough in the field of computational intelligence. A hybrid approach based on differential evolution and the k-means algorithm was used to group requirements and evaluate their relative relevance. By combining differential evolution to refine the initial solution and k-means clustering to construct the first solution, we attempt to combine the benefits of the two approaches. This method may require extensive parameter adjustments and fine-tuning to achieve optimum results, which can be challenging and time consuming.

e: FACING SCALABILITY ISSUES WITH MACHINE LEARNING [S71]

A unique framework for requirement prioritization, known as Case-Based Ranking, is thoroughly described in this research. Their framework is different from AHP in that it permits a prioritization process even over a broad collection of requirements, even if it employs an elicitation approach based on the acquisition of paired preferences. This is made possible by the use of machine learning techniques, which derive requirement ranking approximations from a dataset. Simulated data were used to assess the Case-Based Ranking paradigm in relation to the AHP. Their framework demonstrated its effectiveness in handling extensive sets of requirements, as demonstrated by the tests they outlined and their findings. In basic terms, they demonstrated that in terms of the trade-off between the accuracy of requirement prioritization and expert elicitation effort, their method performs better than the AHP methodology on average.

f: CASE-BASED RANKING FOR DECISION SUPPORT SYSTEMS [S73]

The authors proposed a case-based elicitation procedure to solve the difficulty of rank evaluation planning. This approach involved two automated steps. The first step is to create a strategy for case selection and the second step involves estimating preferences for each example using paired comparisons to collect preferences. The goals were to reduce the amount of time spent on elicitation and simplify comparisons. Through a combination of multiple weak rules, this approach achieves incredibly precise prediction rules, while maintaining a balance between accuracy and work. This process is evaluated using both online and offline tests. Online tests include actual users in the civil defence industry, whereas offline tests evaluate procedures and solutions. The case study results show that the rank boost method successfully strikes a compromise between elicitation effort and precision and that the case-based approach is sustainable over a large number of examples.

We have already obtained a summary of each machine-learning approach for prioritizing requirements. Table 13 presents a comprehensive analysis of the strengths and weaknesses of different approaches, enabling us to assess their respective benefits and drawbacks. We will use these insights to determine the most effective strategies for improving requirement prioritization in future projects.

After analyzing Table 13, we can see that machine-learning-based requirement prioritization approaches offer various strengths and face specific challenges. The approach using machine learning for software requirement prioritization achieves a high accuracy of up to 80% but encounters computational difficulties with hyperparameter tuning [S2]. Although a workflow prioritization approach offers comprehensive priority scores, it requires retraining in the event that a department experiences major changes [S21]. An ERP usability framework meets user requirements and reduces the effort required for large-scale initiatives, although its real-world effectiveness remains untested [S36]. Hybrid techniques address scalability and computational complexity while requiring substantial fine-tuning [S49]. Addressing scalability concerns with machine learning improves average accuracy and reduces effort variance, but it fails to handle requirement dependencies [S71]. Finally, case-based ranking for decision support systems strikes a balance between accuracy and elicitation effort, though its effectiveness increases with the number of events and the acquisition policy design is lacking in details [S73].

TABLE 13. Strength and weakness of Machine learning based requirement prioritization approaches.

Study ID	Technique name	Strength	Weakness
S2	Software Requirements Prioritization Using Machine Learning	The high degree of accuracy, which the majority of the models in this case reached at 80% accuracy, is the strength of utilizing machine learning models	A challenge to practical implementation might be the computing work involved in hyperparameter tuning.
S21	A Machine Learning Approach to workflow Prioritization	Creates a priority score with more precision	If significant changes are made in one department, the entire model must be retrained.
S36	Framework to enhance ERP Usability	This framework meets the requirements of the users and reduce the time and effort required to undertake these large-scale initiatives.	The framework has not been empirically evaluated
S49	A hybridized approach based on K-means and evolutionary algorithms	Scalability, rank reversals, and computing complexity were all addressed by the suggested approach	To get the best results, the approach might need a lot of fine-tuning and parameter changes, which can be difficult and time-consuming.
S71	Facing scalability issues with machine learning	Better average accuracy and reduced effort variance	It cannot handle requirement dependencies
S73	Case-Based ranking for Decision Support System	Accuracy and elicitation effort are balanced in this strategy	The rated preference and associated cognitive overload lose their usefulness as the number of cases rises. The acquisition policy's design elements were not illustrated in the paper

3) OPTIMIZATION-BASED APPROACHES

An optimization algorithm is a process that repeatedly compares different solutions to obtain the best or most appropriate solution [35]. To obtain the best possible set of requirements that yields an optimal solution, optimization methods are well suited for system requirements that are regarded as the next-release problem (NRP) [2], [100].

By employing optimization methods, one can ensure that the chosen requirements maximize the overall value while adhering to project constraints. In the next section, we explore the optimization-based approaches presented in Table 14.

The following subsection provides a detailed overview of the optimization-based requirement prioritization approaches mentioned in Table 14. All of these techniques provide distinct solutions to the challenging problem of software requirement prioritization.

a: ARTIFICIAL BEE COLONY ALGORITHM [S16]

This article presents a Parallel Multi-Objective Artificial Bee Colony (PMOABC) algorithm for optimizing software requirements in incremental development, which addresses the Next Release Problem (NRP), an NP-hard difficulty in balancing customer satisfaction and cost. The approach was tested on two datasets utilizing a master-slave parallel processing model, and it outperformed serial methods, such as NSGA-II and GRASP, in terms of solution quality, runtime, and scalability. While advantages include the ability to efficiently handle multi-objective criteria and reduce runtime, disadvantages include high prices, unresolved requirements dependencies, parallel processing overhead, and hardware dependency.

b: PERFORMANCE ENHANCEMENT USING LEAST-SQUARES-BASED RANDOM GENETIC ALGORITHM [S26]

This article proposes a Least-Squares-Based Random Genetic Algorithm (LSRGA) to improve requirement prioritization while minimizing decision-making time and effort. The process uses crossover, mutation, and random initialization based on least-squares to generate and rank a set of requirements. Compared to earlier genetic algorithm approaches, the LSRGA demonstrated lower error probability, less redundancy, and higher accuracy when evaluated on a variety of requirement sets. The method works well for handling complex requirements and is straightforward and efficient; however, it needs to be validated in the real world and may incur computational overhead in large-scale systems.

c: METHOD OF SYSTEMS REQUIREMENT OPTIMIZATION BASED ON GENETIC ALGORITHMS [S37]

This study used meta-heuristic techniques in conjunction with an adapted multi-objective function, which has proven to be effective in solving multiple real-world instances of the problem. The topic of selecting system requirements has been stated as a multi-objective optimization problem with two objectives. This maximizes the total customer satisfaction and minimizes the overall cost of system development. The effectiveness of the multi-objective proposed approach was demonstrated and proven by adapting a GA to solve real-world cases and testing it with case studies on two real datasets. The experimental results demonstrate that the updated GA can effectively generate high-quality solutions

TABLE 14. Optimization based requirement prioritization approaches.

Study ID	Technique name	Description
S16	Artificial bee colony algorithm	The authors of this study present a similar approach that is predicated on the idea of a master and slave. To improve the quality of the output
S26	Performance enhancement using least-squares-based random genetic algorithm	This paper introduces a novel method that uses a random genetic algorithm based on least squares to improve performance
S37	Method of systems requirement optimization based on genetic algorithms	The study proposes using a non-dominated sorting genetic algorithm with Pareto tournament (NSGA-IIPT) for multi-objective optimization in system requirements selection.
S43	Hybrid Enriched Genetic Revamped Integer Linear Programming Model(Hybrid EGRILP)	In order to maximise income and reduce project time, the article uses a hybrid EGRILP model that combines the processes of software release, prioritization, and scheduling.
S59	Interactive Genetic Algorithm	In this study, they propose an Interactive Genetic Algorithm (IGA) that integrates the current limitations, including dependencies and priorities, with incremental knowledge acquisition.
S62	An ant colony optimization approach	This study offer a novel ant colony optimization based approach for the software release planning problem with the presence of dependent requirement
S65	Interactive RP using genetic algorithm	An interactive genetic algorithm has been suggested in this study to gather paired data that can be used to prioritize the requirements for a software system
S66	Search Techniques for Next Release Problem	The authors of this study use and explain three meta-heuristic search strategies: ant colony systems, genetic algorithms, and simulated annealing to solve NRP.
S70	Search-based approaches to the component selection and prioritization problem	The issue of selecting software component sets to combine in component-based software engineering is the focus of this paper.
S72	EVOLVE method	This paper suggests EVOLVE, an optimization technique that uses a genetic algorithm to continuously plan incremental software development.

TABLE 14. *(Continued.)* **Optimization based requirement prioritization approaches.**

S74	NSGA-II	The study proposes non-dominated sorting genetic algorithm II (NSGA-II), that is based on non-dominated sorting-based multi-objective EA (MOEA)
S75	DDP: A requirements interaction model	They have described a novel iterative method of execution, summarization, and decision-making in order to converge towards nearly-optimal requirement attainment in large-scale requirements models
S76	Using heuristics techniques to NRP	The authors of this article use different heuristics algorithms to address the Next Release Problem (NRP).

and outperform other relevant algorithms previously published in the literature using a set of public datasets.

d: HYBRID ENRICHED GENETIC REVAMPED INTEGER LINEAR PROGRAMMING MODEL (HYBRID EGRILP) [S43]

This study describes the requirement prioritization and scheduling process in software release planning and proposes an Enriched Genetic Algorithm (EGA) with revised integer linear programming (RILP). The EGA introduces an aging factor to prevent premature convergence and uses a dynamic population to improve the computational efficiency. By including resource restrictions to reduce project costs and duration, RILP improves upon the conventional Integer Linear Programming. Two models–a Hybrid EGRILP model with temporal dependency restrictions and one without–were compared through simulations. According to the results, the Hybrid EGRILP model schedules and prioritizes requirements in an ideal manner, thereby minimizing project delays and increasing revenue.

e: INTERATIVE GENETIC ALGORITHM [S59]

This paper proposes an Interactive Genetic Algorithm (IGA) that incorporates current limitations, such as dependencies and priorities, together with incremental knowledge gain. We also evaluated the performance of the proposed algorithm. An actual case study was used to validate the proposed approach, which has many constraints that render exhaustive elicitation (AHP)-based prioritization techniques. They contrasted their method with IAHP, an AHP variation that allows for partial requirement elicitation, and hence avoids AHP scaling issues. The results show that IGA performs better than IAHP, particularly when there are fewer evoked pairs; however, in any event, there are between 25 and 125 elicited pairs. The IGA can generate a good approximation of the standard requirement ranking with a reasonable quantity of manual effort and a reasonable rate of human error tolerance.

f: AN ANT COLONY OPTIMIZATION APPROACH [S62]

A unique ACO-based solution to the software release planning problem with dependent requirements is presented in this study. Experiments were performed to assess the efficacy of the proposed method in comparison with a Genetic Algorithm and Simulated Annealing. The proposed ACO technique demonstrated the potential to produce accurate solutions with minimal computational cost, as indicated by all testing findings. Future research could expand the tests to include real-world examples, which should lead to further intriguing discoveries regarding the suitability of the suggested ACO algorithm for software release planning.

g: INTERATIVE RP USING GENETIC ALGORITHM [S65]

This study suggests an interactive evolutionary algorithm that uses pairwise comparisons to prioritize software requirements. The method outperformed AHP, which has trouble handling large datasets when used in genuine case studies with numerous requirements. The method outperformed a conventional genetic algorithm that optimized the initial restrictions without user input and scaled well. This greatly improved the prioritization of requirements by asking users to perform 50–100 pairwise comparisons, especially in cases where the original ranking information was lacking. This technique is suitable for scenarios with partially reliable input because it strikes a compromise between user effort and performance, and maintains its robustness even in the face of user failures. Further investigations in various contexts and case studies are required to corroborate these results.

h: SEARCH TECHNIQUES FOR NEXT RELEASE PEOBLEM [S66]

The Next Release Problem (NRP), which aims to maximize customer satisfaction in software development under resource restrictions, was evaluated in this study using ant colony optimization (ACO). The method optimizes requirement selection and strikes a balance between effort and satisfaction by using pheromone-based heuristic updates. According to a case study, ACO outperformed Simulated Annealing and Genetic Algorithms in terms of achieving customer satisfaction and efficiently allocating resources. Scalability and efficient management of conflicting requirements are among its strengths. On the other hand, iterative pheromone updates and the need for parameter adjustment for best outcomes contribute to longer execution times.

i: SEARCH-BASED APPROACHES TO THE COMPONENT SELECTION AND PRIORITIZATION PROBLEM [S70]

The application of search-based software engineering to the selection and prioritization of software components is explored in this work. The authors present the topic of selecting the best component sets as a feature subset selection issue. They show how search methods can balance trade-offs

like cost, desirability, development time, and revenue using data from a telecom network. Its main strength is its capacity to manage complicated constraints and inter-component interactions. However, thorough information about every component is necessary for practical execution, and this information is not always easily accessible.

j: EVOLVE METHOD [S72]

The EVOLVE strategy promotes continuous software development planning among all the strategies that use evolutionary algorithms to rank software requirements. It uses a genetic algorithm within the Risk Optimizer tool to assign requirements to increments, optimizing stakeholder satisfaction while adhering to the technological limits. The outcome is a set of prospective solutions that can be used to make the final decisions. The iterative nature and flexibility of this approach in dealing with competing objectives are two of its advantages. A potential problem is the reliance on correct effort estimates as well as the complexity of balancing stakeholder preferences.

k: NSGA-II [S74]

Non-dominated Sorting Genetic algorithm II (NSGA-II) is a fast and elitist multi-objective genetic algorithm aimed at improving computational efficiency and maintaining solution diversity in multi-objective optimization. Its goal is to solve problems with previous iterations, namely, the requirement for a sharing parameter, lack of elitism, and high computing costs. An elitist selection procedure, crowding-distance computation, and a quick non-dominated sorting strategy are all used in practical implementations. Compared to the two existing elitist MOEAs, PAES and SPEA, the proposed NSGA-II was able to retain a better dispersion of solutions and converge better in the produced non-dominated front on nine distinct challenging test problems taken from the literature. Strengths include faster sorting, better diversity, and efficient constraint-handling, whereas weaknesses include complexity in high-dimensional problems and dependence on problem-specific parameters.

l: DDP: A REQUIREMENTS INTERACTION MODEL [S75]

This study suggests a convergent method for prioritizing software requirements in large-scale systems by combining the TAR2 summarization tool with a requirements interaction model. The goal is to optimize the cost-benefit ratio while handling complex requirement relationships. Practical implementation uses NASA's Defect Detection and Prevention (DDP) model alongside TAR2 to analyze and identify critical requirements from large datasets. The method successfully narrowed the solution space and revealed important priority decisions when tested on a dataset with 99 mitigation measures. A key strength is its ability to handle large-scale problems, whereas its weaknesses include being time-consuming and heavily dependent on the TAR2 tool.

m: USING HEURISTICS TECHNIQUES TO NRP [S76]

The study focuses on employing a next release problem (NRP) model that incorporates search-based strategies to optimise requirement selection under resource constraints in order to solve software release planning (SRP) problem. The method uses algorithms such as genetic algorithms, simulated annealing, and hill climbing to rank requirements according to a cost-benefit analysis. According to a comparative study, simulated annealing performs consistently better than hill climbing and greedy algorithms, particularly for larger and more complicated issue settings. However, its computational cost is high, and more research is required to determine the practicality of the model.

The explanation above provides a general idea of each approach. Now, we are going to find out the strengths and weaknesses of the mentioned approaches based on the primary studies. Table 15 presents the strengths and weaknesses of these approaches. This analysis will help to identify the most suitable methods for optimizing requirement prioritization in various project environments.

Optimization-based methods for prioritising requirements include a number of strengths and weaknesses. Among its main advantages are enhanced performance, as evidenced by the artificial bee colony algorithm [S16] and the ant colony optimisation technique [S62], which shorten execution times and improve solution quality. The optimisation of systems requirements using evolutionary algorithms is one of the strategies that prioritise customer satisfaction while reducing development expenses [S37]. Furthermore, methods like DDP efficiently manage a lot of requirements [S75], while flexible approaches, such the EVOLVE method [S72], adapt to shifting priorities and restrictions. However, a number of weaknesses were found. Hybrid EGRILP and the Interactive Genetic Algorithm are two examples of approaches that need additional testing on larger datasets or more intricate user models to confirm their usefulness [S43], [S59]. Search-based techniques for component prioritisation and the performance enhancement genetic algorithm are two examples that need to be implemented in real-world settings to demonstrate their efficacy [S26], [S70]. Challenges include significant computing costs and time consumption, as demonstrated by heuristic-based approaches and DDP [S75], [S76]. In addition, there are still ways to improve the calibre of solutions and the limitations in handling needs dependencies or constraints.

In the previous sections, we reviewed and assessed a range of AI-based methods for requirement prioritization (RP), categorized as fuzzy logic, machine learning (ML), and optimization-based approaches. In the previous tables, the strengths and weaknesses of these methods are provided separately. We now compare various AI-based methods in Table 16 according to the following key evaluation criteria: scalability, accuracy, efficiency, consistency, adaptability, and optimization, along with their tool usage. In table 16, adaptability and consistency is only evaluated for fuzzy logic-based approaches, whereas optimization pertains to machine learning and optimization-based techniques. As a result, the

TABLE 15. Strength and weakness of optimization based requirement prioritization approaches.

Study ID	Technique name	Strength	Weakness
S16	Artificial bee colony algorithm	It shortens execution times while increasing solution quality.	Requirements dependencies are not handled and cost is high.
S26	Performance enhancement using least-squares-based random genetic algorithm	Straightforward, simple to use, and quick to handle greater requirements.	It is necessary to validate the methods by assessing them in practical environments.
S37	Method of systems requirement optimization based on genetic algorithms	It maximizes customer satisfaction and minimizes development cost	Tests on a bigger dataset in real-world settings are necessary
S43	Hybrid Enriched Genetic Revamped Integer Linear Programming Model(Hybrid EGRILP)	This model minimizes the project duration as well as project cost	This is implemented only on 8 requirements ,need to apply on large dataset
S59	Interactive Genetic Algorithm	An accurate estimate of the reference requirements ranking is produced by IGA	They test using a very basic user error model, more complex models are required
S62	An ant colony optimization approach	It outperforms simulated annealing and genetic algorithms in accuracy and efficiency.	Require more time than the simulated annealing problem and the genetic algorithm
S65	Interactive RP using genetic algorithm	It improves the non-interactive optimization and ignore the elicited preferences and handle a number of problematic requirements	Further empirical research is needed to ascertain the actual applicability and feasibility.
S66	Search Techniques for Next Release Problem	The obtained solutions are promptly adjusted to eliminate all problem constraints, and GRASP consistently identifies the most satisfying solution.	It is necessary to improve the quality of the solutions
S70	Search-based approaches to the component selection and prioritization problem	This approach simultaneously optimizes multi-objective functions	Need to be implemented in real world environment
S72	EVOLVE method	It provides more flexibility by accommodating modifications to priorities and requirements constraints	It can only identify solutions that are located near the convex hull in the target space
S74	NSGA-II	Compared to two other elitist MOEAs, PAES and SPEA, the suggested NSGA-II was able to preserve a better distribution of solutions and converge better in the resulting non-dominated front	This algorithm encountered challenges when attempting to solve this so-called highly epistatic problem
S75	DDP: A requirements interaction model	DDP addresses up to 150 requirements, risk, and mitigation while producing nearly ideal solutions	It is time consuming and heavily depends on the TAR2 tool.
S76	Using heuristics techniques to NRP	Simple objective function and an easy implementation process	GRASP techniques increase the time of computation

optimization column is purposefully left blank for fuzzy logic methods, whereas the adaptability and consistency column is vacant for machine learning and optimization approaches. This table provides a clear understanding of how these techniques perform against various parameters, highlighting their strengths and limitations in a consolidated format.

TABLE 16. Comparative analysis of AI-based requirement prioritization approaches.

Category	Study ID	Scalability	Accuracy	Efficiency	Consistency	Adaptability	Optimization	Tool Used
Fuzzy Logic	S19	No	No	No	No	Yes	-	Yes
	S20	Yes	Partial	Yes	No	No	-	Yes
	S33	No	Partial	No	No	No	-	No
	S34	Yes	No	Yes	No	Yes	-	No
	S35	Yes	Partial	Yes	No	No	-	No
	S42	No	Partial	Yes	No	No	-	Yes
	S46	No	Partial	No	No	No	-	No
	S47	Yes	Yes	Yes	Yes	No	-	No
	S48	No	No	Yes	No	No	-	Yes
	S53	Yes	Partial	No	No	Yes	-	No
	S54	Yes	Partial	Yes	No	Yes	-	Yes
	S61	Yes	No	Yes	No	No	-	Yes
	S63	Yes	Yes	Yes	No	Yes	-	No
Machine Learning	S2	Yes	Yes	Yes	-	-	Yes	No
	S21	Yes	Yes	Yes	-	-	No	Yes
	S36	Yes	Yes	Yes	-	-	Yes	No
	S49	Yes	Partial	Yes	-	-	Yes	Yes
	S71	Yes	Yes	No	-	-	Yes	No
	S73	Yes	Partial	Yes	-	-	Yes	No
Optimization	S16	No	Yes	Partial	-	-	Yes	Yes
	S26	Partial	Yes	Partial	-	-	Yes	No
	S37	Partial	Yes	Partial	-	-	Yes	No
	S43	No	Yes	Partial	-	-	Yes	No
	S59	Partial	Yes	Partial	-	-	No	No
	S62	Partial	Yes	No	-	-	Yes	No
	S65	Partial	Yes	No	-	-	No	No
	S66	Partial	Yes	No	-	-	No	No
	S70	No	Yes	Partial	-	-	Yes	No
	S72	No	Yes	No	-	-	Yes	Yes
	S74	No	Yes	Partial	-	-	Yes	Yes
	S75	No	Yes	No	-	-	No	Yes
	S76	No	Yes	No	-	-	Yes	No

A comparison of AI-based requirement prioritization techniques is presented in Table 16 and categorized into fuzzy logic, machine learning, and optimization techniques. For fuzzy logic examining 13 of the 18 techniques that provided relevant information on these criteria. The accuracy of fuzzy logic techniques is limited; only [S63] and [S47] achieved over 90% accuracy, while seven approaches reported partial accuracy, and others gave no information on accuracy. Scalability is a strength of studies like [S20], [S34], [S35], [S47], [S53], [S54], [S61], and [S63], indicating their ability to handle complex datasets. Only [S47] calculated the consistency ratio, indicating that consistency has rarely been discussed. While some, such as [S20], [S33], [S35],[S42], [S46], [S47], [S48], and [S61] lacked adaptability features, [S34] had them, including self-optimization and runtime prioritization. Some studies [S19],[S20], [S42], [S48], [S54] used tools, such as Microsoft Visual Studio and MATLAB Fuzzy Logic Toolbox, to assess tool support; however, some studies did not describe tool implementation. Overall, fuzzy approaches show flexibility and efficiency at runtime; however, there are still issues with consistency, accuracy reporting, and tool documentation.

Machine-learning approaches have demonstrated strong performance in terms of scalability, accuracy, and optimization. All these areas were good in studies [S2], [S36], and [S71], but they lacked information on how the tools were implemented. [S21] fared well in terms of accuracy, scalability and efficiency, but lacked optimization, whereas [S49] demonstrated good scalability and optimization despite partial accuracy. Comparably, [S73] focused on optimization while omitting tool specifics and offering partial accuracy.

Optimization-based approaches consistently deliver high accuracy and optimization capabilities across studies such as [S16], [S26], [S37], [S43], [S62], [S70], [S72], [S74], and [S76]. However, scalability remains a challenge with only partial scalability reported in [S26], [S37], [S59], [S62], and [S65]. The efficiency varied with partial results in [S16], [S26],[S37], [S43],[S59], [S70] and [S72], despite the fact that specialized tools were only employed in studies [S16], [S72], [S74], and [S75]. All things considered, optimization methods are more accurate and optimized; however, they require improvement in terms of scalability, efficiency, and tool integration.

These findings highlight the necessity for additional research to overcome the shortcomings of each AI-based approach that have been highlighted. By overcoming these gaps, future research can enhance the practical applicability, reliability, and performance of AI techniques in requirement prioritization, ensuring more effective and efficient solutions for real-world challenges.

D. RQ4) WHICH SEMI-AUTOMATED APPROACHES ARE CURRENTLY AVAILABLE FOR REQUIREMENT PRIORITIZATION?

Current software requirement prioritization procedures are manual, offline, and laborious, particularly for large requirements. The lack of automation reduces the effectiveness and increases the effort. Ineffective prioritization and a lack of validation can cause delays in the delivery of products and, in extreme situations, affect the project's total cost, time, and risk [36], [37]. Relying on manual procedures and expert knowledge can be inefficient and time-consuming and can lead to biased evaluations, affecting project success and solution quality [38], [39], [40]. Semi-automated methods of requirement prioritization are used to overcome these challenges by combining manual and automatic processes. These methods take into account a variety of considerations, including importance, time limits, financial implications, and associated risks, and seek to expedite the prioritization process [36]. Therefore, for this research question, we explored the available semi-automated approaches for requirement prioritization. Table 17 presents the semi-automated approaches identified after applying our search strategy.

Each technique is summarized in the following subsections, which are listed in Table 17.

TABLE 17. Semi-automated approaches for requirement prioritization.

Study ID	Approaches	Description
S4	Semi-Automated Requirements Prioritization framework (SARiP)	In this study, a semi-automated requirements prioritization framework (SARiP) is presented to apply a semi-automatic method in requirements prioritization.
S6	CDBR:a semi-automated dependency based collaborative requirement prioritization approach	This study introduce a dependency-based, semi-automated collaborative requirement prioritization method (CDBR).
S10	SRPTackle: A semi-automated requirements prioritization technique	This research overcome the key challenges by introducing a novel semiautomated scalable prioritization technique known as "SRPTackle.
S11	A semi-automatic multiple-criteria prioritization process for functional and non-functional requirements	The study proposes a semi-automatic method for prioritizing functional and non-functional requirements in software product lines
S22	StakeQP: A semi-automated stakeholder quantification and prioritization technique	The purpose of this research is to introduce StakeQP, a novel semi-automated technique for quantifying and prioritising stakeholders' requirements for software system projects.
S38	DRank: A Semi-automated Requirements Prioritization Method	This study developed DRank to prioritize optional requirements by taking stakeholder preferences and requirement dependencies into account
S39	SAFFRON: A Semi-Automated Framework for Software Requirements Prioritization	This study presents the Semi-Automated Framework for Software Requirements Prioritization (SAFFRON), which addresses the rank reversal issue in requirement prioritization

1) SEMI-AUTOMATED REQUIREMENTS PRIORITIZATION FRAMEWORK(SARiP) [S4]

This study proposes a semi-automated requirement prioritization framework (SARiP). It implements a semi-automatic method to prioritize the requirements. Using a classification tree and ranking algorithm, SARiP prioritizes the requirements and focuses on regions linked to requirement prediction priority grouping. The i-Tegur team of the Department of Information Technology, Malaysian Ministry of Housing and Local Government (KPKT) successfully evaluated SARiP in the government sector domain. A significant majority of the participants (80 percent) expressed a high likelihood of finding SARiP, which is useful for prioritizing the requirements of the project. All participants agreed that SARiP is a reliable and useful tool. Future work will consider documenting the requirements and prioritization findings, and a traceability mechanism will be incorporated to track changes in requirements.

2) CDBR: A SEMI-AUTOMATED DEPENDENCY-BASED COLLABORATIVE REQUIREMENT PRIORITIZATION APPROACH [S6]

This paper proposes a CDBR, which is a semi-automated collaborative requirement prioritization approach. The objective is to improve communication between stakeholders and developers during requirement prioritization. The CDBR combines stakeholder input with dependency relationships between requirements using an Analytic Hierarchy Process and an Interactive Genetic Algorithm. Using Particle Swarm Optimization (PSO), the approach refines initial rankings to produce a final priority list with minimal conflict. Based on multiple requirement sets and a case study (Cargo Booking Management), CDBR demonstrated scalability, accuracy, and efficiency, outperforming AHP in accuracy and IGA in

processing time; however, it requires computational effort for larger datasets.

3) SRPTackle: A SEMI-AUTOMATED REQUIREMENTS PRIORITIZATION TECHNIQUE [S10]

In this study, a new semi-automated RP technique (SRPTackle) and an automation implementation tool (SRPTackle-Tool) were developed to address the issues of scalability, time consumption, overreliance on expert intervention, and lack of automation when assessing stakeholders' impact on requirement prioritization. K-means clustering for classification, Binary Search Tree (BST) for sorting, and Weighted Sum Model (WSM) for requirement priority value (RPV) computation are all combined in this method.When used with the SRPTackle tool, it ensures unambiguous implementation processes and reduces the need for expert input. Using the RALIC benchmark dataset, the technique demonstrated high accuracy (up to 94.65%) and reduced time consumption compared to existing techniques, such as StakeRare, Lim et al.'s GA, and Saffron. Its strengths include efficient handling of large-scale requirements, reduced reliance on experts, and improved accuracy and speed. However, it is partially automated and still requires stakeholder input, and its practical application has been validated only on benchmark datasets.

4) A SEMI-AUTOMATIC MULTIPLE-CRITERIA PRIORITIZATION PROCESS FOR FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS [S11]

This study describes a data-driven requirement prioritization process that SPL projects can apply to. The primary goal of the suggested prioritization approach is to decrease stakeholder participation by identifying additional criteria that will help mitigate risks such as stakeholder disagreement and time constraints. They used artificial intelligence (AI) methods such as natural language processing (NLP) and machine learning algorithms to leverage data from many sources to optimize criteria identification. Additionally, the team examined two datasets that were utilized for requirement prioritization and FR/NFR categorization. This approach addresses frequent issues, such as stakeholder disagreement and time constraints, by significantly reducing the requirement for considerable stakeholder input. This method improves the quality and relevance of prioritized requirements by utilizing data from several sources to guarantee an objective and data-driven prioritization process.

5) StakeQP: A SEMI-AUTOMATED STAKEHOLDER QUANTIFICATION AND PRIORITIZATION TECHNIQUE [S22]

This study introduces StakeQP, a semi-automated stakeholder quantification and prioritization technique that addresses significant weaknesses in existing systems, such as dependency on expert input, lack of automation, and lack of standardized attribute measurement standards. StakeQP automates the SQP process, minimizes biases, and saves time by introducing Attribute Measurement Criteria (AMC) and integrating TOPSIS for multi-attribute decision-making. When tested on the RALIC industrial dataset, StakeQP outperformed conventional approaches in terms of both accuracy (89.69%) and efficiency, generating findings substantially faster. Automation, low-level implementation details, and minimal expert participation are among its strengths; however, its reliance on predetermined AMC may limit flexibility in extremely dynamic contexts.

6) DRank: A SEMI-AUTOMATED REQUIREMENT PRIORITIZATION METHOD [S38]

The DRank method proposes a semi-automated requirements prioritization approach that integrates stakeholder preferences and dependencies between requirements. The objective is to improve the prioritization accuracy and practicality while minimizing human effort. DRank employs a Prioritization Evaluation Attributes Tree (PEAT) for criteria selection, uses RankBoost for subjective prioritization, and leverages a PageRank-based algorithm to analyze requirement dependencies. The controlled experiments revealed that DRank outperformed AHP, EVOLVE, and CBRank in terms of speed and effectiveness. It effectively manages contributions and business dependencies but ignores other types, such as technical or resource-based requirements, limiting its overall applicability.

7) SAFFRON: A SEMI-AUTOMATED FRAMEWORK FOR SOFTWARE REQUIREMENTS PRIORITIZATION [S39]

This study introduces SAFFRON, a semi-automated system for solving the rank reversal problem in software requirement prioritization. SAFFRON minimizes human interaction by predicting stakeholder evaluations through item- and model-based collaborative filtering. When tested on the RALIC dataset, it revealed a significant ranking correlation while reducing human interaction by 13.5-27%. Scalability, minimal dependency on stakeholders, and reliable prioritization results are among its strengths. However, it depends on requirement similarity, struggles with evolving or new requirements that lack historical data, and may encounter difficulties in highly dynamic contexts.

From the above explanation, we obtain an overview of each approach. Based on these primary studies, the strengths and weaknesses of the aforementioned approaches are presented in Table 18.

The strengths and weaknesses of various semi-automated approaches are listed in Table 18. The SARiP framework [S4] is reliable but stores results in an Excel file rather than a database. CDBR [S6] is efficient and faster than AHP and IGA but requires computationally intensive optimization for large requirement sets. The RALIC dataset is a major source of dependence for SRPTackle [S10], which limits its generalizability despite achieving high accuracy (93.0% - 94.65%). The multiple-criteria prioritization procedure [S11] increases accuracy using NLP, sentiment analysis, and machine learning. However, it struggles with skewed datasets and requires additional ones. StakeQP [S22]

TABLE 18. Strength and weakness of semi-automated approaches in requirement prioritization.

Study ID	Technique name	Strength	Weakness
S4	Semi-Automated Requirements Prioritization framework (SARiP)	SARiP framework is reliable and useful according to the opinion of the participants	The suggested SARiP prototype does not store the results in a database, instead, it uses an excel file.
S6	CDBR:A semi-automated dependency based collaborative requirement prioritization approach	In terms of accuracy and processing time, respectively, CDBR beat AHP and IGA.	While Particle Swarm Optimization reduces disagreement, large requirement sets may demand more computationally intensive methods.
S10	SRPTackle: A semi-automated requirements prioritization technique	The minimum and highest accuracy percentages that SRPTackle can achieve are 93.0% and 94.65%, respectively.	Reliance on the RALIC dataset may limit its generalizability to all software scenarios.
S11	A semi-automatic multiple-criteria prioritization process for functional and non-functional requirements	The use of natural language processing (NLP), sentiment analysis, and machine learning algorithms improves the accuracy	This review revealed several of their limitations such as imbalanced datasets and the need for additional datasets.
S22	StakeQP: A semi-automated stakeholder quantification and prioritization technique	This technique can produce more accurate result in less time	Relies on predefined AMC, limiting flexibility in dynamic environments.
S38	DRank: A Semi-automated Requirements Prioritization Method	This method is less time consuming and more effective than the compared methods in this study	DRank is limited because it only handles contribution and business dependencies, missing other types like technical or resource-based dependencies.
S39	SAFFRON: A Semi-Automated Framework for Software Requirements Prioritization	The suggested approach minimises human interaction by 13.5-27%.	Struggles with new or evolving requirements due to insufficient historical data.

quickly generates accurate findings, although it may have erroneous stakeholder details in the RALIC dataset. Despite its efficacy and time efficiency, DRank [S38] only supports contributions and business dependencies. SAFFRON [S39] addresses rank reversal and reduces human involvement by 13.5-27%; however, it may not be able to keep up with rapidly changing or unique requirements without previous data.

E. RQ5) WHY REPRIORITIZATION IS IMPORTANT WITHIN AGILE SOFTWARE DEVELOPMENT?

This research question explored the importance of reprioritization in ASD. From Table 9 and Figure 6, we can clearly see that, among other challenges, continuous requirement changes are the most cited challenges that need to be addressed to enhance Agile RP. Therefore, the requirements must be reprioritised after changes occur. Therefore, after exploring the challenges, we selected this research question to show the importance of reprioritization to deal with the most cited challenges and to motivate researchers to find an efficient technique that is most suitable for agile environments to reprioritize requirements.

Compared with other engineering disciplines, change is a fundamental characteristic of software engineering. Specifying every software requirement in real-world scenarios is challenging owing to the constantly evolving needs and circumstances. Changes can arise from external factors, such as customer pressure and evolving business needs, or internal factors, such as poor understanding of requirements and technical infeasibility. Additionally, fluctuating market conditions, global competition, governmental regulations, and other factors significantly influence how the requirements change over time [101]. Managing changes in software development is critical for project success because inadequate change management can lead to delays, increased expenses, and project failure.

Requirement changes occur throughout the lifecycle, including during maintenance. Agile methodologies help

manage shifting requirements by prioritizing valuable user stories, ensuring high-quality software, and maximizing the business value. Effective change management, especially in geographically dispersed teams with multiple stakeholders, relies on effective communication and collaboration [81]. Various surveys have shown that a major cause of large software project failures is the simultaneous change in requirements [102], [103]. Therefore, reprioritization is required to adjust for any changes in the prioritization list.

In agile-scrum development, requirements are maintained in a dynamic product backlog that is continuously reprioritized to adjust the changed requirements based on client needs [104]. In traditional requirements engineering, the prioritization process typically occurs once. However, in agile environments, prioritization is required in every developmental life cycle [29]. The product owner plays a pivotal role in managing and prioritizing the product backlog for agile software development, ensuring that it aligns with evolving client needs at the start of each sprint [28]. Agile software development requires constant requirement prioritization, with the requirements ranked in order at the start of each sprint. Current methods frequently depend on outdated approaches, which increase the risk of project interruptions and result in significant costs [33]. Prioritization methods based on a ratio scale, such as AHP, cannot be used for reprioritization because of the huge complexity involved in pairwise comparisons involving already prioritized requirements and newly arrived or changed requirements. Limited research has explored efficient agile requirement reprioritization, leaving a critical gap in dynamic development environments [84].

One of the most important components of agile development is requirement reprioritization, which aims to maximize customer value and accommodate changing requirements. However, the agile requirements engineering (RE) literature provides limited insight into how reprioritization is actually carried out in practice [19]. Neither the academic literature nor the industry possesses robust and reliable reprioritization techniques that demonstrate effectiveness. The field of reprioritization remains relatively unexplored, and research on efficient methods for conducting this activity with minimal effort is lacking [84]. After identifying the need for reprioritization, we searched the literature that focused on reprioritization and obtained a very small number of papers, as shown in Table 19, with references and short descriptions.

The exploration of reprioritization through systematic literature surveys and case studies are held in a research paper. This study aims to provide background information about reprioritization and offer a detailed look at current practices in the software industry. It explores different reprioritization techniques by studying existing literature and real-world examples from case studies. It conducts an in-depth investigation through a specific case study focused exclusively on reprioritization within a few multinational software development organizations. Addressing a research question about managing dynamism, the study concludes that none of the existing techniques explicitly addresses the reprioritization of decision aspects. The study suggests areas for future investigation by the scientific community and identifies research gaps in the field of reprioritization. The main emphasis is on the need for effective reprioritization techniques to better manage software development in organizations [S44].

TABLE 19. Short description of the primary studies regarding reprioritization.

Study ID	Description
S44	The purpose of this research is to define reprioritisation in software development, distinguish it from prioritisation, examine present methods, identify research gaps, and investigate future potential. Using a thorough literature survey, previous case studies, and a detailed case study of multinational software organisations, it concludes that efficient reprioritization methods are vital for competitiveness and calls for advanced reprioritization strategies to address industry needs.
S51	This article examines incremental development in mass-market software, focusing on decision-making, requirement prioritization, and reprioritization to enhance quality, reduce costs, and shorten delivery times. Insights from two multinational and one medium-sized company show that minimal-effort prioritization boosts initial increments, while efficient reprioritization enhances delivery efficiency.
S34	This paper proposes Adaptive Fuzzy Hierarchical Cumulative Voting (AFHCV), an improved version of FHCV with an adaptive mechanism to handle runtime events. It adds new requirements, reallocates, alters priorities, and reprioritizes, enhancing overall results.
S55	This paper proposes a reprioritization technique that minimizes effort by reducing pairwise comparisons while covering most requirements. Evaluated on a live system, the technique showed higher requirement coverage and reduced effort for reprioritization.
S69	This paper supports dynamic reprioritization in Agile by exploring issues in prioritization decision-making. It presents a client-based model for inter-iteration prioritisation, and intends to provide a decision-support tool that incorporates value and cost criteria to improve practices.

Another case study examines the role of prioritization and reprioritization in both requirements and decision aspects for creating an implementation order in the incremental delivery of evolving software, emphasizing their importance for the success of software products and the survival of developing organizations in the market. They conducted their interviews in a few software developing firms in the fields of web, mobile, and desktop applications. The results of the interviews are presented in this study. It demonstrates how frequently there are gaps in customer- and developer-centric prioritization and reprioritization activities. Key findings indicate that while priorities frequently change over time, the priorities of decision aspects rarely change, resulting in neglected reprioritization. There is a recognized need for an effective, low-effort technique to adjust the priorities of the already implemented requirements [S51].

Researchers in software engineering have illustrated various requirement prioritization techniques. Selecting a specific prioritization method is entirely dependent on the nature and objectives of the project. However, existing techniques often fail to adapt whenever the value of a requirement's attributes changes and are not suitable when decision-makers are uncertain about assigning points to requirements [S34].

Reprioritizing requirements involves changing the order of importance for requirements that have already been implemented, as well as updating the requirements that are new, delayed, and modified. Requirement prioritization is dynamic owing to this approach. These changes occur because of different factors, such as shifts in stakeholders' viewpoints and decision aspects as well as changes in the environment, such as competitors and business rules. These factors force software analysts to re-evaluate and reassign requirement priorities accordingly. The product owner is responsible for reprioritizing new or changed requirements in the product backlog. Therefore, there is a need for an effective reprioritization method that requires minimal effort [S55].

Thus, it is evident from the above explanation that reprioritization techniques that can adapt to requirement changes in agile contexts are urgently needed. Agile teams should ideally have effective mechanisms for fast adjustment to changing circumstances, enabling flexible and adaptable user requirement prioritization. However, the techniques currently in use are inadequate for dealing with constant growth and changes in requirements. This highlights the need for more research in the field of reprioritization as well as the necessity of a useful reprioritization method.

F. RQ6) WHAT ARE THE AVAILABLE REPRIORITIZATION TECHNIQUES?

In this research question, we explore whether there are any techniques available for reprioritisation. After understanding the need for reprioritization in ASD based on research question 5, we explored reprioritization techniques if available. Therefore, we used our search strategy to find relevant literature. There are no solid, trustworthy, and effective reprioritization methods in the academic or industrial literature. There is a scarcity of research on effective ways to perform reprioritisation with little effort, leaving the topic mostly unexplored. For reprioritisation, a few conceptual models are available [18], [68], [104], [105]. However, we obtained only one reprioritisation technique based on pairwise comparisons [S55]. We found additional techniques that emphasized the importance of reprioritisation without specifically focusing on it. Instead, they emphasize increasing the coverage of events that can occur at runtime. [S34]. However, because they used fuzzy logic to deal with uncertainty and changes that are uncertain, their methodological approach is appropriate for an agile context and may be used for reprioritization. However, more thorough investigation is required to support this viewpoint. Table 20 presents the two available techniques, with their objectives and limitations.

The table presents two available techniques: reprioritization of pairwise compared requirements and Adaptive Fuzzy Hierarchical Cumulative Voting (AFHCV).

The approach proposed for reprioritisation worked well for reprioritizing the requirements that were compared pairwise. It focuses only on new, changed, or high-impact requirements and adjusts the priorities of related requirements based on dependencies. It takes a hierarchical approach, first categorizing requirements as ''high,'' ''medium,'' or ''low'' priority using a numerical assignment mechanism before selectively using pairwise comparisons. The primary rationale for this is that if the requirement's priority is determined to change, there is a good chance that all other requirements, whose execution is dependent upon this requirement or vice versa, will also likely see their priorities adjusted to align with the new and relative priorities. The technique was tested in a live case study with automated tools for requirement prioritization, proving its capacity to considerably reduce the number of pairwise comparisons while maintaining full coverage. However, this approach has limitations, such as trouble handling ties, exponential growth when compared to more requirements, and challenges in maintaining accurate dependency hierarchies at the systems scale. Furthermore, they may fail to adapt to rapidly changing priorities in agile environments [S55].

The current techniques for prioritizing requirements are inadequate in situations where decision-makers are uncertain about how many points to assign to various requirements. An article proposed the integration of fuzzy logic with requirement prioritization methods to address ambiguity and uncertainty. The reaction of the system to change serves as the foundation for justifying the addition of an adaptive mechanism. To effectively manage uncertainties in the face of extremely dynamic consumer or stakeholder requirements, it is recommended that adaptive mechanisms and fuzzy logic be integrated. The suggested Adaptive Fuzzy Hierarchical Cumulative Voting (AFHCV) incorporates an adaptive

TABLE 20. Existing technique regarding reprioritization.

Techniques	Description	Limitations	Study ID
Requirement reprioritization for pairwise compared requirements	The proposed technique had been suitable for reprioritization of requirements that are pairwise compared	1)Handling ties or equal preferences can be challenging within the pairwise comparison framework 2) As the number of requirements increases, the number of pairwise comparisons grows exponentially 3) Limited stakeholder involvement in dependency assessment can result in an inaccurate hierarchical structure, leading to suboptimal prioritization due to an incomplete understanding of the true nature of dependencies 4) As the software system grows, maintaining an accurate hierarchical structure of dependencies becomes harder 5) The method can be time-consuming and resource-intensive especially for large projects	S55
AFHCV (Adaptive Fuzzy Hierarchical Cumulative Voting)	To expand the range of possible events at runtime, the proposed Adaptive Fuzzy Hierarchical Cumulative Voting (AFHCV) technique combines an adaptive mechanism with the already-existing Fuzzy Hierarchical Cumulative Voting (FHCV) technique	1)It doesn't go into specific details about the exact steps involved in the reprioritization process within AFHCV technique. Although it mentions that reprioritization is part of the adaptive mechanism, it provides some general steps commonly associated with the process. 2)The scalability of the technique might be a concern, especially when dealing with large projects or complex hierarchical structures. 3)The technique's effectiveness may be more theoretical than practical if it hasn't been extensively validated in a diverse set of real-world scenarios.	S34

TABLE 20. *(Continued.)* Existing technique regarding reprioritization.

mechanism into the existing Fuzzy Hierarchical Cumulative Voting (FHCV) technique. The goal of this integration is to improve the adaptability in response to runtime events. Requirements are arranged hierarchically, triangular fuzzy numbers are used for fuzzification, and stakeholder feedback and modified decision parameters are used for reallocation and reprioritization. A comparative study showed that the AFHCV performed better than the FHCV in terms of results. However, AFHCV lacks specific details on the reprioritization steps and raises concerns regarding scalability for large projects or complex structures. In addition, the practicality of the technique may be limited if it is not tested in diverse real-world scenarios. The reliance of this technique on the human assignment of priorities may lead to longer processing times, which is another limitation [S34].

It is necessary to close this critical gap because both the literature currently in publication and industrial practices lack sufficiently efficient reprioritisation plans. Uncontrolled changes throughout the development process pose a serious danger of disrupting projects in Agile Software Development (ASD) because of a lack of suitable tools for dynamic decision-making in response to requirement changes. In addition, the lack of strong reprioritization techniques may make it more difficult for the team to adjust to changing market conditions and client demands, which could jeopardize the general responsiveness and agility essential to ASD.

Improving the effectiveness and success of agile projects requires addressing these limitations.

IV. IMPLICATIONS FOR RESEARCH AND PRACTICE

This systematic literature review has significant implications for academic research on agile software development and industrial practice. It emphasizes the importance of agile software development projects to succeed by using efficient methods for requirement prioritization and reprioritization to provide value to stakeholders. This study offers a thorough overview of state-of-the-art requirement prioritization techniques by examining both the conventional and newly developed AI-based and semi-automated approaches.

This review identifies critical gaps in the existing literature, particularly in the area of requirement reprioritization. Agile projects are dynamic and iterative, which means that ongoing reprioritization is required to effectively adapt to changing requirements and stakeholder needs. However, a notable deficiency in robust techniques and structures exists for systematically managing frequent reprioritisations. This emphasizes the necessity of creating more sophisticated techniques that are scalable and adaptable.

Researchers are encouraged to look into advanced methods, including artificial intelligence and machine learning algorithms, to improve the flexibility and responsiveness of requirements management procedures. This review highlights how AI-based approaches may be used in requirement prioritization as a potential area for future work. Some of the most promising techniques include fuzzy logic, machine learning, and optimization-based approaches, which can handle the uncertainties and complexities of agile environments. However, more empirical validation and real-world testing are required to establish their effectiveness and reliability. Thus, researchers should concentrate on developing artificial intelligence models capable of predicting changes in priorities, optimizing decision-making processes, and providing up-to-date recommendations for action.

Similarly, semi-automated approaches that combine both manual and automated procedures have been discussed as a way to streamline requirement prioritization. These methods help balance the flexibility of manual processes with the efficiency of automation. Research needs to develop and refine these hybrid solutions to ensure that they are user friendly and adaptable across various project contexts.

This review provides useful insights for practitioners of various requirement prioritization techniques. By understanding the strengths and weaknesses of diverse techniques, practitioners can select them appropriately based on their unique project requirements. In addition, this study highlights some challenges associated with requirement prioritization, where scalability issues, constant changes, and neglecting non-functional requirements are the most discussed problems. Thus, practitioners can anticipate and mitigate frequent project challenges. These can be managed by recognizing such challenges, thus putting plans in place to improve stakeholder participation, control dependencies, and align evolving requirements with project objectives.

Exploration of AI-based and semi-automated techniques offers practical solutions for improving the prioritization of requirements. Using fuzzy logic-based methods, practitioners can attempt to manage uncertainty and ambiguity, or use machine learning models to predict changes in priority. Many benefits accrue to an organization when it employs semi-automated techniques that enable it to streamline its processes by combining human judgement with automated efficiency. Additionally, Agile projects require dynamic reprioritisation because they require flexibility and responsiveness. Therefore, reprioritization techniques should be incorporated into this process so that frequent changes can be handled without much trouble, because project conditions change very quickly. It is important to note that these approaches ensure that critical requirements are always prioritized, thereby delivering the maximum value to stakeholders.

V. THREATS TO VALIDITY

The threat to validity assessment is a crucial component of any empirical study, including systematic literature reviews (SLRs). This typically involves evaluating several types of validity, such as construct, internal, external, and conclusion. Other forms of validity, such as theoretical and interpretive validity, are rarely mentioned in software engineering [106]. Therefore, in our study, we discuss three types of validity: internal validity, construct validity, and external validity, to ensure the quality of our study.

A. INTERNAL VALIDITY

According to Kitchenham and Charters, a typical threat to SRL is the examination of all research papers and pertinent papers related to specific research questions [42]. Our aim was to optimize the internal validity by selecting relevant online databases containing an adequate number of pertinent studies. Five well-known online databases were selected: IEEE Xplore, Springer Link, Scopus, Science Direct, and ACM Digital Library. Some of these databases, such as ACM and IEEE, were specifically focused on in our field of study, and included all journal articles and conference proceedings. To optimize internal validity, we also searched a variety of current scientific publications for effective and comprehensive search terms, including synonyms, other substitute words, and acronyms, that pinpoint and explore the specific study subject. A backward and forward evaluation of the selected studies was performed using the snowballing search strategy to locate relevant articles that would be missed if the search phrase was limited.

B. CONSTRUCT VALIDITY

To optimize the construct validity, we used the guidelines of Kitchenham and Charters for article selection [42]. We maintained high standards by carefully assessing each article to ensure compliance with predetermined criteria and assessing the quality of the included studies. Data extraction forms were created to gather data, confirm the information gathered, and provide a checklist for identifying the required data. Data

synthesis is a methodology that attempts to provide comprehensive answers to significant research issues regarding the method of requirement prioritization and reprioritization in agile software development. At this stage, the retrieved data were initially categorized into manual, AI-based prioritization, semi-automated, and reprioritization techniques. Through this categorization, we addressed the research questions that we wanted to answer. We aimed to cover the prioritization and reprioritization techniques, and our selected primary studies successfully covered all of these parts. However, there is a lack of available studies on reprioritisation techniques. We obtained a sufficient number of primary studies to cover the prioritization process in all categories. We provide our best efforts to maximize the construct validity of our research using a systematic approach.

1) EXTERNAL VALIDITY

External validity refers to the extent to which a study's findings can be generalized beyond its specific context. In this study, we focus on prioritization and reprioritization techniques in the field of agile software development. This study included 76 articles gathered using the forward and backward snowballing strategies. Using this approach allowed us to include a large amount of relevant literature, which enhanced the external validity of the study by including different research perspectives on requirements prioritising. Therefore, our findings provide valuable guidance for future software engineering research, particularly in the field of requirement prioritization. The challenges associated with agile software development (ASD) have also drawn attention to more significant problems with requirement engineering that must be fixed. Thus, our results cover several aspects of requirement prioritization in software engineering and extend beyond ASD.

VI. FUTURE WORK

In this systematic literature review, the exploration of requirement prioritization and reprioritization in agile software development (ASD) indicated topics that require further research. The dynamic nature of agile projects requires techniques that can closely manage the changing requirements and stakeholder needs. To address the challenges and limitations described in the current literature as well as emerging approaches to improving the agility and effectiveness of software development processes, this section highlights future work.

Some potential future works include:

1. It is important to develop and evaluate dynamic reprioritization techniques that can manage rapid changes that occur in agile projects. The goal of this study is to develop techniques that adapt to the changing requirements of stakeholders and project conditions.
2. Integrate machine learning and artificial intelligence into processes to prioritize requirements, with an emphasis on developing flexible AI models and hybrid strategies that combine AI-driven automation with human expertise.
3. Improve semi-automated approaches for prioritization by creating hybrid approaches that are flexible and easy to use. Researchers must focus on various factors, including risks, costs, time restrictions, and importance.
4. Empirical validation is required for many of the suggested AI-based and semiautomated approaches to determine their efficacy in actual agile projects. Experiments and case studies should be part of future research to support these techniques.
5. The scalability and applicability of prioritization techniques should be the focus of future research, particularly in large-scale projects with multiple stakeholders. However, most of these techniques have scalability issues. Accordingly, different frameworks and tools to support prioritization processes for projects at various levels of complexity are being created.
6. More research is needed to determine how the best stakeholders can be engaged in the process of prioritizing their requirements, particularly when artificial intelligence-based techniques are used. This is accomplished by developing an interface that can be easily navigated without many restrictions and ensuring that all stakeholder values and goals have been appropriately recorded and represented in the outcomes.
7. A longitudinal research study will help track how agile project lifecycles are affected by various prioritization and reprioritization strategies, providing insights into possible downsides as well as long-term benefits in relation to different approaches, such as Scrum or Kanban.

By addressing the areas outlined in this future work section, researchers can significantly contribute to the advancement of requirement prioritization and reprioritization techniques in Agile Software Development, thereby improving the efficiency and effectiveness of agile development practices.

VII. CONCLUSION

A systematic literature review performed with the objective of identifying and categorizing techniques used for requirement prioritization and reprioritization in the context of ASD presents a relevant understanding of the current state of the art and highlights several gaps in the current state of knowledge. This review, which adhered to the guidelines of Kitchenham and Charters, systematically extracted and synthesized data from 76 primary studies, offering valuable insights into the most commonly used techniques, effectiveness of AI-based and semi-automated approaches, and challenges and limitations faced in the field. MoSCoW, Analytic Hierarchy Process (AHP), Cumulative Voting, Kano Model, and Cost-Value Ranking were found to be the most commonly used techniques in ASD according to the review. These techniques have been widely used because of their efficiency in confirming the evolution of stakeholder needs with project outcomes. However, the dynamic nature of agile projects necessitates continuous reprioritisation, and

this review underscores the need for more robust and scalable techniques to manage this effectively.

AI-based methods for requirement prioritization, such as fuzzy logic, machine learning techniques, and optimization algorithms, show potential because they can handle prioritization challenges. Although empirical validation in real-world agile projects is required, these methods are promising in terms of efficiency and flexibility. Research has also been conducted on semi-automated approaches, which combine both manual and automatic elements, and promising results have been noted, especially in areas such as the usability and flexibility of prioritization techniques.

A literature review revealed some significant shortcomings and challenges of the present study. Significant issues include the computation of costs, scalability, and dynamic reprioritization techniques that can adapt to the changing conditions of the project. There is a clear need for further research and development in this area, as there is a notable lack of reliable methods and frameworks to effectively manage frequent reprioritisation.

Most primary research studies included in the quality assessment displayed high levels of validity and reliability. In contrast, four studies were invalidated for failing to meet the minimum standards set for excellence. This detailed assessment guarantees that the conclusions derived from the review are valid and reliable, thereby forming a solid foundation for further research and applications.

The importance of the implications of the research findings for future academic and industrial research courses is also discussed. To provide value to stakeholders, this study also highlights the significance of implementing efficient techniques for requirements prioritization and reprioritization in ASD initiatives. To identify significant gaps in the literature, the need for advanced techniques that can be scaled and adjusted to the dynamic character of agile projects, especially in the area of requirement reprioritization, was proposed.

In conclusion, this systematic review provides a comprehensive overview of requirement prioritization and reprioritization techniques for ASD, outlining the strengths and weaknesses of current techniques. This study contributes to the ongoing development of agile methodologies by identifying gaps in literature and suggesting new research directions. To ensure the success of agile projects and their ability to adapt to the changing requirements of stakeholders and the dynamic nature of software development, there is a clear need to develop more effective and efficient reprioritisation techniques.

APPENDIX A
PRIMARY STUDIES

This section describes the selected primary studies, sorted by publication year, from the newest to the oldest.

[S1] A. H. Muhammad, A. Siddique, M. Mubassher, A. Aldweesh, and Q. N. Naveed, ''Prioritizing non-Functional Requirements in Agile Process using Multi Criteria Decision Making Analysis,''*IEEEAccess*, 2023, doi:10.1109/ACCESS.2023.3253771

[S2] A. Fatima, A. Fernandes, D. Egan, and C. Luca, ''Software Requirements Prioritization Using Machine Learning,'' in *International Conference on Agents and Artificial Intelligence*, Science and Technology Publications, 2023, pp. 893–900. doi: 10.5220/0011796900003393

[S3]H. Zulzalil et al., ''REQUIREMENTS PRIORITIZATION IN AGILE PROJECTS: FROM EXPERTS' PERSPECTIVES,'' *Article in Journal of Theoretical and Applied Information Technology*, vol. 15, p. 19, 2022

[S4]F.-F. Chua, T.-Y. Lim, B. Tajuddin, and A. P. Yanuarifiani, ''Incorporating Semi-Automated Approach for Effective Software Requirements Prioritization: A Framework Design,'' *Journal of Informatics and Web Engineering*, vol. 1, no. 1, pp.1–15,Mar. 2022, doi: 10.33093/jiwe.2022.1.1.1

[S5]P. Marnada, Raharjo T, B. Hardian, Prasetyo A, ''Agile project management challenge in handling scope and change: A systematic literature review,'' *Procedia Computer Science*,vol.197,pp.290300,2022,doi:10.1016/j.procs.2021.12.143

[S6]A. Gupta and C. Gupta, ''CDBR: A semi-automated collaborative execute-before-after dependency-based requirement prioritization approach,'' *Journal of King Saud University -Computer and Information Sciences*, vol.34,no.2, pp.421–432, Feb.2022,doi: 10.1016/j.jksuci.2018.10.004

[S7]N. Govil and A. Sharma, ''Information Extraction on Requirement Prioritization Approaches in Agile Software Development Processes,'' in *Proceedings - 5th International Conference on Computing Methodologies and Communication, ICCMC 2021*, Institute of Electrical and Electronics Engineers Inc., April 2021,pp.10971100. doi:10.1109/ICCMC51019.2021.9418285

[S8]N. H. Borhan, H. Zulzalil, S. Hassan, and A. N. Mohd, ''A Prioritization Approach Upon Non-Functional and Functional User Stories in Agile-Scrum Software Development (i-USPA): A Preliminary Results,'' *International Journal of Multidisciplinary Sciences and Advanced Technology*, vol. 2, no. 6, pp. 62–76, 2021, [Online]. Available at: http://www.ijmsat.com

[S9]A. Jarzebowicz and P. Weichbroth, ''A Qualitative Study on Non-Functional Requirements in Agile Software Development,'' *IEEE Access*, vol. 9, pp. 4045840475,2021, doi:10.1109/ACCESS.2021.3064424

[S10]F. Hujainah, R. Binti Abu Bakar, A. B. Nasser, B. Al-haimi, and K. Z. Zamli, ''SRPTackle: A semi-automated requirements prioritization technique for scalable requirements of software system projects,'' *Inf Softw Technol*, vol. 131, Mar. 2021, doi: 10.1016/j.infsof.2020.106501

[S11]M. I. Limaylla, N. Condori-Fernandez, M. R. Luaces, M. A. González, J. Pereira, and M. G. Penedo, ''Towards a Semi-Automated Data-Driven Requirements Prioritization Approach for Reducing Stakeholder Participation in SPL Development,'' *engineering proceedings*, 2021, doi: 10.3390/engproc

[S12]U. Singh, N. Upadhyay, ''A Review on Requirements Prioritization Techniques,'' *International Journal of Creative Research Thoughts (IJCRT)*, vol. 8, no. 12, pp. 2320–2882, 2020, [Online]. Available: www.ijcrt.org

[S13]J. C. B. Somohano-Murrieta, J. O. Ocharan-Hernandez, A. J. Sanchez-Garcia, and M. De Los Angeles Arenas-Valdes, ''Requirements prioritization techniques in the last decade: A systematic literature review,'' *Proceedings - 2020 8th Edition of the International Conference in Software Engineering Research and Innovation, CONISOFT 2020*,., Nov. 2020, pp. 11–20. doi: 10.1109/CONISOFT50191.2020.00013

[S14]A. Jarzebowicz and N. Sitko, ''Agile requirements prioritization in practice: Results of an industrial survey,'' *Procedia Computer Science*, Elsevier B.V., 2020, pp. 3446–3455. doi: 10.1016/j.procs.2020.09.052

[S15]K. Abdelazim, R. Moawad, and E. Elfakharany, ''A Framework for Requirements Prioritization Process in Agile Software Development,'' in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Mar. 2020. doi: 10.1088/1742-6596/1454/1/012001

[S16]H. Alrezaamiri, A. Ebrahimnejad, and H. Motameni, ''Parallel multi-objective artificial bee colony algorithm for software requirement optimization,'' *Requir Eng*, vol. 25, no. 3, pp. 363–380, Sep. 2020, doi: 10.1007/s00766-020-00328-y

[S17]N. Borhan, H. Zulzalil, N. Ali, and S. Hassan, ''Requirements Prioritization Techniques Focusing on Agile Software Development: A Systematic Literature Review,'' *Article in International Journal of Scientific & Technology Research*, vol. 8, no. 11, 2019, [Online]. Available: www.ijstr.org

[S18]A. Sapunkov and T. Afanasieva, ''Software for automation of user requirements prioritization,'' *ACM International Conference Proceeding Series*, Association for Computing Machinery, 2019, pp. 1–5. doi: 10.1145/3318236.3318251

[S19]H. Sadia, S. Qamar Abbas, and M. Faisal, ''Volatile Requirement Prioritization: A Fuzzy Based Approach,'' *International Journal of Engineering and Advanced Technology (IJEAT)*, no. 5, pp. 2249–8958, 2019 [Online]. Available at: www.ijeat.org

[S20]D. Mougouei, D. M. W. Powers, E. Mougouei, ''A fuzzy framework for prioritization and partial selection of security requirements in software projects,'' *Journal of Intelligent and Fuzzy Systems*, vol. 37, no. 2, pp. 2671–2686, 2019, doi: 10.3233/JIFS-182907

[S21] N. R. Bollumpally, A. C. Evans, S. W. Gleave, A. R. Gromadzki, and G. L. Sr., ''A Machine Learning Approach to Workflow Prioritization,'' in *2019 Systems and Information Engineering Design Symposium (SIEDS)*, IEEE, 2019. [Online]. Available: 10.1109/SIEDS.2019.8735589

[S22]F. Hujainah, R. B. A. Bakar, and M. A. Abdulgabber, ''StakeQP: A semi-automated stakeholder quantification and prioritization technique for requirement selection in software system projects,'' *Decis Support Syst*, vol. 121, pp. 94–108, Jun. 2019, doi: 10.1016/j.dss.2019.04.009

[S23]Z. Shehzadi, F. Azam, M. W. Anwar, and I. Qasim, ''A novel framework for change requirement management (CRM) in agile software development (ASD),'' in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Aug. 2019, pp. 22–26. doi: 10.1145/3357419.3357438

[S24]N. Saher, F. Baharom, and R. Romli, ''A Review of Requirement Prioritization Techniques in Agile Software Development,'' in *Knowledge Management International Conference (KMICe), Miri Sarawak, Malaysia*, 2018, pp. 25–27. [Online]. Available at: http://www.kmice.cms.net.my/

[S25]F. Hujainah, R. B. A. Bakar, M. A. Abdulgabber, and K. Z. Zamli, ''Software Requirements Prioritization: A Systematic Literature Review on Significance, Stakeholders, Techniques and Challenges,'' *IEEE Access*, vol. 6, pp. 71497–71523, 2018, doi: 10.1109/ACCESS.2018.2881755

[S26]H. Ahuja, Sujata, and U. Batra, ''Performance Enhancement in Requirement Prioritization by Using Least-Squares-Based Random Genetic Algorithm,'' in *Studies in Computational Intelligence*, Springer Verlag, 2018, pp. 251–263. doi: 10.1007/978-981-10-4555-4_17

[S27]M. S. Rahim, A. E. Chowdhury, and S. Das, ''RIZE: A Proposed Requirements Prioritization Technique for Agile Development,'' in *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC) 21 - 23 Dec 2017, Dhaka, Bangladesh*, 2017

[S28] H. Sheemar and G. Kour, ''Enhancing User-Stories Prioritization Process in Agile Environment,'' in *2017 International Conference on Innovations in Control, Communication and Information Systems (ICICCI)*, IEEE, 2017.

[S29] R. Qaddoura, A. Abu-Srhan, M. H. Qasem, and A. Hudaib, ''Requirements prioritization techniques review and analysis,'' in *Proceedings - 2017 International Conference on New Trends in Computing Sciences, ICTCS 2017*, Institute of Electrical and Electronics Engineers Inc., Jul. 2017, pp. 258–263. doi: 10.1109/ICTCS.2017.55

[S30] M. Alkandari and A. Al-Shammeri, ''Enhancing the Process of Requirements Prioritization in Agile Software Development - A Proposed Model,'' *Journal of Software*, vol. 12, no. 6, pp. 439–453, Jun. 2017, doi: 10.17706/jsw.12.6.439-453

[S31] R. V. Anand and M. Dinakaran, ''Handling stakeholder conflict by agile requirement prioritization using Apriori technique,'' *Computers and Electrical Engineering*, vol. 61, pp. 126–136, Jul. 2017, doi: 10.1016/j.compeleceng.2017.06.022

[S32] M. Sadiq, ''A Fuzzy Set-Based Approach for the Prioritization of Stakeholders on the Basis of the Importance of Software Requirements,'' *IETE J Res*, vol. 63, no. 5, pp. 616–629, Sep. 2017, doi: 10.1080/03772063.2017.1313140

[S33] K. S. Ahmad, N. Ahmad, H. Tahir, and S. Khan, ''Fuzzy_MoSCoW: A Fuzzy based MoSCoW Method for the Prioritization of Software Requirements,'' in *2017 International Conference on Intelligent Computing,Instrumentation and Control Technologies (ICICICT)*, IEEE, 2017

[S34] B. B. Jawale, G. K. Patnaik, and A. T. Bhole, ''Requirement Prioritization using Adaptive Fuzzy Hierarchical Cumulative Voting,'' in *2017 IEEE 7th International Advance Computing Conference (IACC)*, 2017. [Online]. Available: DOI 10.1109/IACC.2017.26

[S35] K. Gulzar, J. Sang, M. Ramzan, and M. Kashif, "Fuzzy approach to prioritize usability requirements conflicts: An experimental evaluation," *IEEE Access*, vol. 5, pp. 13570–13577, Jul. 2017, doi: 10.1109/ACCESS.2017.2725321

[S36] S. Qayyum and A. Abbasi, "Framework to Enhance ERP Usability by Machine Learning Based Requirements Prioritization," *Journal of Software*, vol. 12, no. 8, pp. 664–670, Aug. 2017, doi: 10.17706/jsw.12.8.664-670

[S37] M. H. Marghny, H. M. El-Hawary, W. H. Dukhan, H. Marghny, M. El-Hawary, H. Dukhan, "An Effective Method of Systems Requirement Optimization Based on Genetic Algorithms," *Information Sciences Letters*, vol. 6, 2017, doi: 10.12785/isl/060102

[S38] F. Shao, R. Peng, H. Lai, and B. Wang, "DRank: A semi-automated requirements prioritization method based on preferences and dependencies," *Journal of Systems and Software*, vol. 126, pp. 141–156, Apr. 2017, doi: 10.1016/j.jss.2016.09.043

[S39] S. Ali Asif, Z. Masud, R. Easmin, and A. Ul Gias, "SAFFRON: A Semi-Automated Framework for Software Requirements Prioritization," *IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 8, no. 12, 2017, [Online]. Available at: www.ijacsa.thesai.org

[S40] S. Hudda, R. Mahajan, and S. Chopra, "Prioritization of User-Stories in Agile Environment," *Indian Journal of Science and Technology*, vol. 9, no. 45, Dec. 2016, doi: 10.17485/ijst/2016/v9i45/105069

[S41]W. R. Fitriani, P. Rahayu, and D. I. Sensuse, "Challenges in Agile Software Development: A Systematic Literature Review," in *2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, IEEE, 2016[Online].Available at: http://agilemanifesto.org/iso/en/manifesto.html

[S42]N. Mishra, M. A. Khanum, and K. Agrawal, "Approach to Prioritize the Requirements Using Fuzzy Logic," in *ACEIT Conference Proceeding 2016*, 2016.

[S43]S. Valsala and A. R. Nair, "Requirement Prioritization and Scheduling in Software Release Planning Using Hybrid Enriched Genetic Revamped Integer Linear Programming Model," *Research Journal of Applied Sciences, Engineering and Technology* ', vol. 12, no. 3, pp. 347–354, Feb. 2016, doi: 10.19026/rjaset.12.2342

[S44]V. Gupta, D. S. Chauhan, and K. Dutta, "Exploring reprioritization through systematic literature surveys and case studies," *Springerplus*, vol. 4, no. 1, Dec. 2015, doi: 10.1186/s40064-015-1320-0

[S45]I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," *Computers in Human Behavior*, vol. 51. Elsevier Ltd, pp. 915–929, Oct. 01, 2015. doi: 10.1016/j.chb.2014.10.046

[S46]P. Achimugu, A. Selamat, and R. Ibrahim, "Using the fuzzy multi-criteria decision-making approach for software requirements prioritization," *J Teknol*, vol. 77, no. 13, pp. 21–28, 2015, doi: 10.11113/jt.v77.6321

[S47]M. I. Babar, M. Ghazali, D. N. A. Jawawi, S. M. Shamsuddin, and N. Ibrahim, "PHandler: An expert system for a scalable software requirements prioritization process," *Knowl Based Syst*, vol. 84, pp. 179–202, Aug. 2015, doi: 10.1016/j.knosys.2015.04.010

[S48]M. Dabbagh, S. P. Lee, "An Approach for Prioritizing NFRs According to Their Relationship with FRs," *Lecture Notes on Software Engineering*, vol. 3, no. 1, pp. 1–5, 2015, doi: 10.7763/lnse.2015.v3.154

[S49]P. Achimugu, A. Selamat, A. T. Azar, and S. Vaidyanathan, "A hybridized approach for prioritizing software requirements based on k-means and evolutionary algorithms," *Studies in Computational Intelligence*, vol. 575, pp. 73–93, 2015, doi: 10.1007/978-3-319-11017-2_4

[S50]P. Achimugu, A. Selamat, R. Ibrahim, and M. N. R. Mahrin, "A systematic literature review of software requirements prioritization research," *Information and Software Technology*, vol. 56, no. 6. Elsevier B.V., pp. 568–585, 2014. doi: 10.1016/j.infsof.2014.02.001

[S51]V. Gupta, D. S. Chauhan, C. Gupta, and K. Dutta, "Current prioritization and reprioritization practices: a case study approach," *Int. J. Computer Aided Engineering and Technology*, vol. 6, no. 2, pp. 159–170, 2014

[S52]M. A. A. Elsood and H. A. Hefny, "A Goal-Based Technique for Requirements Prioritization," in *The 9th International Conference on INFOrmatics and Systems (INFOS2014)*, IEEE, 2014

[S53]M. Sadiq and S. K. Jain, "Applying fuzzy preference relation for requirements prioritization in goal-oriented requirements elicitation process," *International Journal of System Assurance Engineering and Management*, vol. 5, no. 4, pp. 711–723, Dec. 2014, doi: 10.1007/s13198-014-0236-3

[S54]H. Momeni, H. Motameni, and M. Larimi, "A Neuro-Fuzzy based Approach to Software Quality Requirements Prioritization," *International Journal of Applied Information Systems (IJAIS)*, vol. 7, no. 7, 2014, [Online]. Available: www.ijais.org

[S55]D. S. Chauhan and K. Dutta, "Requirement reprioritization for pairwise compared requirements," *Int. Journal. Computer Aided Engineering and Technology*, vol. 6, no. 1, pp. 29–47, 2014

[S56]A. Perini, A. Susi, and P. Avesani, "A machine learning approach to software requirements prioritization," *IEEE Transactions on Software Engineering*, vol. 39, no. 4, pp. 445–461, 2013, doi: 10.1109/TSE.2012.52

[S57]A. Martakis and M. Daneva, "Handling Requirements Dependencies in Agile Projects: A Focus Group with Agile Software Development Practitioners," in *IEEE 7th International Conference on Research Challenges in Information Science (RCIS)*, IEEE, Aug. 2013. [Online]. Available at: www.scopus.com

[S58]P. Bajaj and V. Arora, "Multi-person decision-making for requirements prioritization using fuzzy AHP," *ACM SIGSOFT Software Engineering Notes*, vol. 38, no. 5, pp. 1–6, Aug. 2013, doi: 10.1145/2507288.2507302

[S59]P. Tonella, A. Susi, and F. Palma, "Interactive requirements prioritization using a genetic algorithm," in

Information and Software Technology, Jan. 2013, pp. 173–187. doi: 10.1016/j.infsof.2012.07.003

[S60]A. Ejnioui, C. E. Otero, and L.D. Otero, "A Simulation-based Fuzzy Multi-attribute Decision Making for Prioritizing Software Requirements," in *RIIT '12: Proceedings of the 1st Annual conference on Research in information technology*, 2012

[S61]A. Ejnioui, C. E. Otero, and A. A. Qureshi, "Software Requirement Prioritization Using Fuzzy Multi-attribute Decision Making," in *2012 IEEE Conference on Open Systems*, Oct. 2012, pp. 1-6. [Online]. Available: 10.1109/ICOS.2012.6417646

[S62]J. Teixeira De Souza et al., "An Ant Colony Optimization Approach to the Software Release Planning with Dependent Requirements," *Search Based Software Engineering. SSBSE 2011*, vol. 6956, pp. 142–157, 2011, [Online]. Available: https://doi.org/10.1007/978-3-642-23716-4_15

[S63]M. Ramzan, M. Arfan Jaffar, and A. A. Shahid, "VALUE BASED INTELLIGENT REQUIREMENT PRIORITIZATION (VIRP): EXPERT DRIVEN FUZZY LOGIC BASED PRIORITIZATION TECHNIQUE," *International Journal of Innovative Computing*, vol. 7, no. 3, pp. 1017–1038, 2011

[S64]M. Aasem, M. Ramzan, and A. Jaffar, "Analysis and optimization of software requirements prioritization techniques," in *2010 International Conference on Information and Emerging Technologies*, 2010. [Online]. Available: 10.1109/ICIET.2010.5625687

[S65]P. Tonella, A. Susi, and F. Palma, "Using interactive GA for requirements prioritization," in *Proceedings - 2nd International Symposium on Search Based Software Engineering, SSBSE 2010*, 2010, pp. 57–66. doi: 10.1109/SSBSE.2010.17

[S66]J. Del Sagrado, I.M. Del Águila, and F. J. Orellana, "Ant colony optimization for the next release problem a comparative study," in *Proceedings - 2nd International Symposium on Search Based Software Engineering, SSBSE 2010*, 2010, pp. 67–76. doi: 10.1109/SSBSE.2010.18

[S67]Z. Racheva and M. Daneva, "Reprioritizing the Requirements in Agile Software Development: Towards a Conceptual Model from Clients' Perspective," in *Proceedings of the Twenty-First International Conference on Software Engineering & Knowledge Engineering*, 2009, pp. 73–80. [Online]. Available at: https://www.researchgate.net/publication/221391125

[S68]M. Ramzan, M. A. Jaffar, M. A. Iqbal, S. Anwar, and A. A. Shahid, "Value Based Fuzzy Requirement Prioritization and its Evaluation Framework," in *2009 Fourth International Conference on Innovative Computing, Information and Control*, IEEE, 2009.

[S69]Z. Racheva, M. Daneva, and L. Buglione, "Supporting the dynamic reprioritization of requirements in agile development of software products," in *2008 2nd International Workshop on Software Product Management, ISWPM'08*, IEEE Computer Society, 2008, pp. 49–57. doi: 10.1109/IWSPM.2008.7

TABLE 21. Quality assessment criterion.

Study ID	QA1	QA2	QA3	QA4	QA5	Overall Score
S1	1	1	1	1	1	5
S2	1	1	1	1	0.5	4.5
S3	1	1	1	1	0.5	4.5
S4	1	1	1	1	1	5
S5	1	1	1	1	1	5
S6	1	1	1	1	1	5
S7	1	1	1	1	1	5
S8	1	1	1	1	1	5
S9	1	1	0.5	0.5	1	4
S10	1	1	1	1	1	5
S11	1	1	0.5	1	1	4.5
S12	1	1	1	1	0.5	4.5
S13	1	1	1	0.5	1	4.5
S14	1	1	1	1	1	5
S15	1	1	0.5	1	0.5	4
S16	1	1	1	1	1	5
S17	1	1	1	1	1	5
S18	1	0.5	1	0.5	1	4
S19	1	1	0.5	0.5	0.5	3.5
S20	1	1	0.5	1	0.5	4
S21	1	1	0.5	1	1	4.5
S22	1	1	1	1	1	5
S23	1	1	1	0.5	0.5	4
S24	1	1	1	1	0.5	4.5
S25	1	1	1	1	1	5
S26	1	1	1	1	1	5
S27	1	1	0.5	0.5	1	5
S28	1	1	1	1	1	5
S29	1	1	1	1	1	5
S30	1	0.5	0.5	0.5	0.5	3
S31	1	1	0.5	1	1	4.5
S32	1	0.5	0.5	0.5	0.5	3
S33	1	0.5	1	1	1	4.5
S34	1	0.5	1	0.5	1	4
S35	1	1	0.5	0.5	1	4
S36	1	0.5	0.5	0.5	0.5	3
S37	1	1	0.5	1	0.5	4
S38	1	1	1	1	1	5
S39	1	1	1	1	0.5	4.5
S40	1	1	1	1	0.5	4.5
S41	1	1	0.5	1	1	4.5
S42	1	1	0.5	0.5	0.5	3.5
S43	1	1	1	0.5	0.5	4
S44	1	1	1	1	1	5
S45	1	1	1	1	1	5
S46	1	1	1	1	0.5	4.5
S47	1	1	1	1	1	5
S48	1	0.5	0.5	0.5	0.5	3
S49	1	1	1	1	1	5
S50	1	1	1	1	1	5
S51	1	1	1	1	0.5	4.5
S52	1	1	1	1	1	5
S53	1	1	1	1	1	5
S54	1	1	1	1	0.5	4.5
S55	1	1	1	0.5	0.5	4
S56	1	1	1	1	1	5
S57	1	1	1	1	1	5
S58	1	1	1	1	1	5
S59	1	1	1	1	1	5
S60	1	1	1	1	1	5
S61	1	1	1	1	1	5
S62	1	1	1	1	1	5
S63	1	1	1	1	0.5	4.5
S64	1	1	1	1	1	5
S65	1	1	1	1	1	5

TABLE 21. *(Continued.)* **Quality assessment criterion.**

S66	**1**	**1**	**1**	**1**	**1**	**5**
S67	**1**	**1**	**1**	**1**	**0.5**	**4.5**
S68	**1**	**1**	**1**	**1**	**1**	**5**
S69	**1**	**1**	**1**	**1**	**1**	**5**
S70	**1**	**1**	**0.5**	**1**	**1**	**4.5**
S71	**1**	**1**	**1**	**1**	**1**	**5**
S72	**1**	**1**	**0.5**	**1**	**1**	**4.5**
S73	**1**	**0.5**	**0.5**	**1**	**1**	**4**
S74	**1**	**1**	**0.5**	**1**	**1**	**4.5**
S75	**1**	**1**	**0.5**	**1**	**1**	**4.5**
S76	**1**	**0.5**	**0.5**	**0.5**	**0.5**	**3**

[S70]M. Harman, A. Skaliotis, K. Steinhöfel, and S. Steinhöfel, "Search-Based Approaches to the Component Selection and Prioritization Problem," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 2006. [Online]. Available: https://doi.org/10.1145/1143997.1144321

[S71]P. Avesani, C. Bazzanella, A. Perini, and A. Susi, "Facing Scalability Issues in Requirements Prioritization with Machine Learning Techniques," in *13th IEEE International Conference on Requirements Engineering (RE'05)*, IEEE, 2005. [Online]. Available: 10.1109/RE.2005.30

[S72]D. Greer and G. Ruhe, "Software release planning: An evolutionary and iterative approach," *Inf Softw Technol*, vol. 46, no. 4, pp. 243–253, Mar. 2004, doi: 10.1016/j.infsof.2003.07.002

[S73]P. Avesani, S. Ferrari, and A. Susi, "Case-Based Ranking for Decision Support Systems," *5th International Conference on Case-Based Reasoning, ICCBR 2003*, Springer, 2003. [Online]. Available: https://doi.org/10.1007/3-540-45006-8_6

[S74]K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, vol. 6, no. 2, 2002

[S75]M. S. Feather and T. Menzies, "Converging on the optimal attainment of requirements," in *Proceedings of the IEEE International Conference on Requirements Engineering*, IEEE Computer Society, 2002, pp. 263–270. doi: 10.1109/ICRE.2002.1048537

[S76]A. J. Bagnall, V.J. Rayward-Smith, and I.M. Whittley, "The next release problem," *Inf Softw Technol*, vol. 43, no. 14, pp. 883–890, 2001, [Online]. Available at: www.elsevier.com/locate/infsof

APPENDIX B
QUALITY ASSESSMENT CRITERION

This section presents the quality assessment criteria listed in Table 21. After applying the quality assessment criteria, scores for the five criteria were collected for evaluation. The quality evaluation defined three levels: a high level with scores ranging from 4.5 to 5.0, a medium level with scores from 3.5 to 4.0, and a low level with scores from 2.5 to 3.0. The final quality score for each primary study was calculated by adding the scores of the five quality checklist questions listed in Table 4.

REFERENCES

[1] J. Dick, E. Hull, and K. Jackson, *Requirements Engineering*, 4th ed., Cham, Switzerland: Springer, 2017, doi: 10.1007/978-3-319-61073-3.

[2] R. Anwar and M. B. Bashir, "A systematic literature review of AI-based software requirements prioritization techniques," *IEEE Access*, vol. 11, pp. 143815–143860, 2023, doi: 10.1109/ACCESS.2023.3343252.

[3] P. Talele and R. Phalnikar, "Automated requirement prioritisation technique using an updated Adam optimisation algorithm," *Int. J. Intell. Syst. Appl. Eng.*, vol. 11, no. 3, pp. 1211–1221, Jul. 2023.

[4] B. Jawale and A. T. Bhole, "Adaptive fuzzy hierarchical cumulative voting: A novel approach toward requirement prioritization," *Int. J. Res. Eng. Technol.*, vol. 4, no. 5, pp. 365–370, May 2015. [Online]. Available: http://www.ijret.org

[5] F. A. Bukhsh, Z. A. Bukhsh, and M. Daneva, "A systematic literature review on requirement prioritization techniques and their empirical evaluation," *Comput. Standards Interfaces*, vol. 69, Mar. 2020, Art. no. 103389, doi: 10.1016/j.csi.2019.103389.

[6] S. Robertson and J. Robertson, *Mastering the Requirements Process: Getting Requirements Right*, 3rd ed., Reading, MA, USA: Addison-Wesley, 2012.

[7] A. Aurum and C. Wohlin, *Engineering and Managing Software Requirements*, 1st ed., Berlin, Germany: Springer, 2005, doi: 10.1007/3-540-28244-0.

[8] M. S. Jahan, F. Azam, M. W. Anwar, A. Amjad, and K. Ayub, "A novel approach for software requirement prioritization," in *Proc. 7th Int. Conf. Softw. Eng. Res. Innov. (CONISOFT)*, Oct. 2019, pp. 1–7, doi: 10.1109/CONISOFT.2019.00012.

[9] M. Fowler and J. Highsmith. (2001). *The Agile Manifesto*. [Online]. Available: www.Martinfowler.com/articles/newMethodology.html

[10] J. Miler and P. Gaida, "Identification of the Agile mindset and its comparison to the competencies of selected Agile roles," in *Advances in Agile and User-Centred Software Engineering* (Lecture Notes in Bus. Information Processing), vol. 376, A. Przybyłek and M. E. M. Trujillo, Eds., Cham, Switzerland: Springer, 2020, pp. 41–62, doi: 10.1007/978-3-030-37534-8_3.

[11] K. Schwaber and J. Sutherland, *The Scrum Guide the Definitive Guide to Scrum The Rules of the Game*. USA: Scrum.org & Scrum Alliance, Oct. 2011.

[12] K. Beck, *Praise for Extreme Programming Explained*, 2nd ed., Reading, MA, USA: Addison-Wesley, 2012.

[13] M. Omar and R. B. Romli, "The key factors of evaluating Agile approaches: A systematic literature review," *Int. J. Supply Chain Manag.*, vol. 8, no. 2, pp. 1–11, 2019. [Online]. Available: https://www.researchgate.net/publication/332493463

[14] M. Ochodek and S. Kopczyńska, "Perceived importance of agile requirements engineering practices—A survey," *J. Syst. Softw.*, vol. 143, pp. 29–43, Sep. 2018, doi: 10.1016/j.jss.2018.05.012.

[15] K. Curcio, T. Navarro, A. Malucelli, and S. Reinehr, "Requirements engineering: A systematic mapping study in agile software development," *J. Syst. Softw.*, vol. 139, pp. 32–50, May 2018, doi: 10.1016/j.jss.2018.01.036.

[16] K. Wnuk and P. Mudduluru, "Value-based requirements engineering: Challenges and opportunities," in *Engineering Software Systems: Research and Praxis* (Advances in Intelligent Systems and Computing), vol. 830. Berlin, Germany: Springer, 2019, pp. 20–33, doi: 10.1007/978-3-319-99617-2_2.

[17] Z. Racheva, M. Daneva, K. Sikkel, A. Herrmann, and R. Wieringa, "Do we know enough about requirements prioritization in Agile projects: Insights from a case study," in *Proc. 18th IEEE Int. Requirements Eng. Conf.*, Sep. 2010, pp. 147–156, doi: 10.1109/RE.2010.27.

[18] Z. Racheva, M. Daneva, and A. Herrmann, "A conceptual model of client-driven agile requirements prioritization: Results of a case study," in *Proc. ACM-IEEE Int. Symp. Empirical Softw. Eng. Meas.*, Sep. 2010, pp. 1–4, doi: 10.1145/1852786.1852837.

[19] Z. Bakalova, M. Daneva, A. Herrmann, and R. Wieringa, "Agile requirements prioritization: What happens in practice and what is described in literature," in *Proc. 17th Int. Working Conf. Requirements Eng., Found. Softw. Qual. (REFSQ)*. Essen, Germany: Springer, 2011, pp. 181–195.

[20] A. Martakis and M. Daneva, "Handling requirements dependencies in agile projects: A focus group with agile software development practitioners," in *Proc. IEEE 7th Int. Conf. Res. Challenges Inf. Sci. (RCIS)*, Aug. 2013, pp. 1–11. [Online]. Available: www.scopus.com

[21] A. Jarzębowicz and N. Sitko, "Agile requirements prioritization in practice: Results of an industrial survey," *Proc. Comput. Sci.*, vol. 176, pp. 3446–3455, Jan. 2020, doi: 10.1016/j.procs.2020.09.052.

[22] R. Berntsson Svensson and R. Torkar, "Not all requirements prioritization criteria are equal at all times: A quantitative analysis," *J. Syst. Softw.*, vol. 209, Mar. 2024, Art. no. 111909, doi: 10.1016/j.jss.2023.111909.

[23] N. H. Borhan, H. Zulzalil, S. Hassan, and N. M. Ali, "Requirements prioritization techniques focusing on agile software development: A systematic literature review," *Article Int. J. Sci. Technol. Res.*, vol. 8, no. 11, pp. 2118–2125, Nov. 2019. [Online]. Available: www.ijstr.org

[24] N. Govil and A. Sharma, "Information extraction on requirement prioritization approaches in agile software development processes," in *Proc. 5th Int. Conf. Comput. Methodologies Commun. (ICCMC)*, Apr. 2021, pp. 1097–1100, doi: 10.1109/ICCMC51019.2021.9418285.

[25] U. Singh and N. Upadhyay, "A review on requirements prioritization techniques," *Int. J. Creative Res. Thoughts*, vol. 8, no. 12, pp. 1–7, 2020. [Online]. Available: www.ijcrt.org

[26] J. C. B. Somohano-Murrieta, J. O. Ocharán-Hernández, A. J. Sánchez-García, and M. de los Ángeles Arenas-Valdés, "Requirements prioritization techniques in the last decade: A systematic literature review," in *Proc. 8th Edition Int. Conf. Softw. Eng. Res. Innov. (CONISOFT)*, Nov. 2020, pp. 11–20, doi: 10.1109/CONISOFT50191.2020.00013.

[27] H. Tufail, I. Qasim, M. F. Masood, S. Tanvir, and W. H. Butt, "Towards the selection of optimum requirements prioritization technique: A comparative analysis," in *Proc. 5th Int. Conf. Inf. Manage. (ICIM)*, Mar. 2019, pp. 227–231.

[28] N. Hazlini Borhan, H. Zulzalil, A. Hassan, N. Hayati, and M. Ali, "Requirements prioritization in agile projects: From experts' perspectives," *J. Theor. Appl. Inf. Technol.*, vol. 15, p. 19, Oct. 2022. [Online]. Available: https://docs.google.com/forms/d/e/1FAIpQLSeDT

[29] N. Saher, F. Baharom, and R. Romli, "A review of requirement prioritization techniques in agile software development," in *Proc. Knowl. Manage. Int. Conf. (KMICe)*, Miri Sarawak, Malaysia, Jul. 2018, pp. 25–27. [Online]. Available: http://www.kmice.cms.net.my/

[30] R. Qaddoura, A. Abu-Srhan, M. H. Qasem, and A. Hudaib, "Requirements prioritization techniques review and analysis," in *Proc. Int. Conf. New Trends Comput. Sci. (ICTCS)*, Jul. 2017, pp. 258–263, doi: 10.1109/ICTCS.2017.55.

[31] M. I. L. Lunarejo, "Requirements prioritization based on multiple criteria using artificial intelligence techniques," in *Proc. IEEE Int. Conf. Requirements Eng., IEEE Comput. Soc.*, Sep. 2021, pp. 480–485, doi: 10.1109/RE51729.2021.00072.

[32] V. Gupta, D. S. Chauhan, C. Gupta, and K. Dutta, "Current prioritisation and reprioritisation practices: A case study approach," *Int. J. Comput. Aided Eng. Technol.*, vol. 6, no. 2, pp. 159–170, 2014, doi: 10.1504/IJCAET.2014.060297.

[33] P. Marnada, T. Raharjo, B. Hardian, and A. Prasetyo, "Agile project management challenge in handling scope and change: A systematic literature review," *Proc. Comput. Sci.*, vol. 197, pp. 290–300, Jan. 2022, doi: 10.1016/j.procs.2021.12.143.

[34] M. S. Rahim, A. Z. M. E. Chowdhury, and S. Das, "Rize: A proposed requirements prioritization technique for agile development," in *Proc. IEEE Region 10 Humanitarian Technol. Conf. (R10-HTC)*, Dec. 2017, pp. 634–637, doi: 10.1109/R10-HTC.2017.8289039. Accessed: Sep. 27, 2024.

[35] N. Mishra, M. A. Khanum, and K. Agrawal, "Approach to prioritize the requirements using fuzzy logic," in *Proc. ACEIT Conf.*, 2016, pp. 1–6.

[36] F.-F. Chua, T.-Y. Lim, B. Tajuddin, and A. P. Yanuarifiani, "Incorporating semi-automated approach for effective software requirements prioritization: A framework design," *J. Informat. Web Eng.*, vol. 1, no. 1, pp. 1–15, Mar. 2022, doi: 10.33093/JIWE.2022.1.1.1.

[37] F. Hujainah, R. B. A. Bakar, A. B. Nasser, B. Al-haimi, and K. Z. Zamli, "SRPTackle: A semi-automated requirements prioritisation technique for scalable requirements of software system projects," *Inf. Softw. Technol.*, vol. 131, Mar. 2021, Art. no. 106501, doi: 10.1016/j.infsof.2020.106501.

[38] F. Hujainah, R. B. A. Bakar, and M. A. Abdulgabber, "StakeQP: A semi-automated stakeholder quantification and prioritisation technique for requirement selection in software system projects," *Decis. Support Syst.*, vol. 121, pp. 94–108, Jun. 2019, doi: 10.1016/j.dss.2019.04.009.

[39] F. Hujainah, R. B. Abu Bakar, B. Al-haimi, and M. A. Abdulgabber, "Stakeholder quantification and prioritisation research: A systematic literature review," *Inf. Softw. Technol.*, vol. 102, pp. 85–99, Oct. 2018, doi: 10.1016/j.infsof.2018.05.008.

[40] M. I. Babar, M. Ghazali, D. N. A. Jawawi, S. M. Shamsuddin, and N. Ibrahim, "PHandler: An expert system for a scalable software requirements prioritization process," *Knowl.-Based Syst.*, vol. 84, pp. 179–202, Aug. 2015, doi: 10.1016/j.knosys.2015.04.010.

[41] N. H. Borhan, H. Zulzalil, S. Hassan, and N. M. Ali, "A hybrid prioritization approach by integrating non-functional and functional user stories in agile-scrum software development (i-USPA): A preliminary study," in *Proc. IEEE Int. Conf. Comput. (ICOCO)*, Nov. 2022, pp. 276–282, doi: 10.1109/ICOCO56118.2022.10031863.

[42] B. Kitchenham and S. M. Charters, "Guidelines for performing systematic literature reviews in software engineering," Softw. Eng. Group, Keele Univ., Keele, U.K., Tech. Rep. EBSE-2007-01, Jan. 2007. [Online]. Available: https://www.researchgate.net/publication/302924724

[43] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering—A systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, Jan. 2009, doi: 10.1016/j.infsof.2008.09.009.

[44] B. Kitchenham, "Procedures for performing systematic reviews," Softw. Eng. Group, Keele Univ., U.K., Aug. 2004. [Online]. Available: https://www.researchgate.net/publication/228756057

[45] H. Zhang, M. A. Babar, and P. Tell, "Identifying relevant studies in software engineering," *Inf. Softw. Technol.*, vol. 53, no. 6, pp. 625–637, Jun. 2011, doi: 10.1016/j.infsof.2010.12.010.

[46] J. Webster and R. T. Watson, "Analyzing the past to prepare for the future: Writing a literature review," *MIS Quart.*, vol. 26, no. 2, pp. 13–23, Jun. 2002. [Online]. Available: http://www.misq.org/misreview/announce.html

[47] G. Y. Koi-Akrofi, J. K. Akrofi, and H. Akwetey Matey, "Understanding the characteristics, benefits and challenges of agile it project management: A literature based perspective," *Int. J. Softw. Eng. Appl.*, vol. 10, no. 5, pp. 25–44, Sep. 2019, doi: 10.5121/ijsea.2019.10502.

[48] R. S. Kostev, "Challenges and problems of the Moscow method application in ERP system implementation," in *Proc.11th Int. Sci. Conf. Comput. Sci. (COMSCI) Proc. Inst. Elect. Electron. Eng.*, Jan. 2023, pp. 1–4, doi: 10.1109/COMSCI59259.2023.10315816.

[49] K. AbdElazim, R. Moawad, and E. Elfakharany, "A framework for requirements prioritization process in agile software development," *J. Phys., Conf. Ser.*, vol. 1454, no. 1, Feb. 2020, Art. no. 012001, doi: 10.1088/1742-6596/1454/1/012001.

[50] K. S. Ahmad, N. Ahmad, H. Tahir, and S. Khan, "Fuzzy_MoSCoW: A fuzzy based Moscow method for the prioritization of software requirements," in *Proc. Int. Conf. Intell. Comput., Instrum. Control Technol. (ICICICT)*, Jul. 2017, pp. 433–437.

[51] A. W. Khan, I. Hussain, and M. Zamir, "Analytic hierarchy process-based prioritization framework for vendor's reliability challenges in global software development," *J. Softw., Evol. Process*, vol. 33, no. 3, Mar. 2021, Art. no. e2310, doi: 10.1002/smr.2310.

[52] S. Abusaeed, S. U. R. Khan, and A. Mashkoor, "A fuzzy AHP-based approach for prioritization of cost overhead factors in agile software development," *Appl. Soft Comput.*, vol. 133, Jan. 2023, Art. no. 109977, doi: 10.1016/j.asoc.2022.109977.

[53] M. Shameem, R. R. Kumar, C. Kumar, B. Chandra, and A. A. Khan, "Prioritizing challenges of agile process in distributed software development environment using analytic hierarchy process," *J. Softw., Evol. Process*, vol. 30, no. 11, Nov. 2018, Art. no. e1979, doi: 10.1002/smr.1979.

[54] A. Rida, S. Nazir, A. Tabassum, and S. Asim, "The impact of analytical assessment of requirements prioritization models: An empirical study," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 2, pp. 1–11, 2017. [Online]. Available: www.ijacsa.thesai.org

[55] N. A. Teridi, Z. Adzhar, N. M. D. Rahim, J. Kamis, T. Ridzuan, Z. Adnan, and M. F. A. Rauf, "The approach using cumulative voting and spanning tree technique in implementing functional requirement prioritization: A case study of student's financial system development," *Article J. Theor. Appl. Inf. Technol.*, vol. 15, no. 3, pp. 1106–1117, 2023. [Online]. Available: https://www.researchgate.net/publication/373772244

[56] N. Saher*, F. Baharom, and R. Romli, "Guideline for the selection of requirement prioritization techniques in agile software development: An empirical research," *Int. J. Recent Technol. Eng. (IJRTE)*, vol. 8, no. 5, pp. 3381–3388, Jan. 2020, doi: 10.35940/ijrte.e6634.018520.

[57] F. Diniz, T. Silva, Ì. Moura, D. Grosmann, F. M. Neto, and P. R. Neto, "Application of a data visualization technique based on trees to aid prioritization of requirements in agile projects," in *Proc. Int. Conf. Eng. Comput. Educ.*, Mar. 2013, pp. 81–85.

[58] Y. Li, K. Sha, H. Li, Y. Wang, Y. Dong, J. Feng, S. Zhang, and Y. Chen, "Improving the elicitation of critical customer requirements through an understanding of their sensitivity," *Res. Eng. Design*, vol. 34, no. 3, pp. 327–346, Jul. 2023, doi: 10.1007/s00163-023-00410-w.

[59] S. Mote, V. Kulkarni, and D. B. E. Narkhede, "Kano model application in new service development and customer satisfaction," *IOSR J. Bus. Manage.*, vol. 18, no. 8, pp. 10–14, Aug. 2016, doi: 10.9790/487x-1808011014.

[60] P. Achimugu, A. Selamat, R. Ibrahim, and M. N. Mahrin, "A systematic literature review of software requirements prioritization research," *Inf. Softw. Technol.*, vol. 56, no. 6, pp. 568–585, Jun. 2014, doi: 10.1016/j.infsof.2014.02.001.

[61] Y. He and J. Zhong, "Improve requirement prioritization by end-user demands model building and evaluation," Blekinge Inst. Technol., Karlskrona, Sweden, 2021. [Online]. Available: www.bth.se

[62] L. Rojas, C. Olivares-Rodríguez, C. Alvarez, and P. G. Campos, "OurRank: A software requirements prioritization method based on qualitative assessment and cost-benefit prediction," *IEEE Access*, vol. 10, pp. 131772–131787, 2022, doi: 10.1109/ACCESS.2022.3230152.

[63] T. Amelia and R. B. Mohamed, "Review on cost-value approach for requirements prioritization techniques," in *Proc. 5th Int. Conf. Inf. Technol., Comput., Electr. Eng. (ICITACEE)*, Sep. 2018, pp. 310–314, doi: 10.1109/ICITACEE.2018.8576908.

[64] N. H. Borhan, H. Zulzalil, S. Hassan, N. M. Ali, A. B. M. Sultan, and R. Bahsoon, "Stakeholder analysis using fuzzy logic operations for integrated user story prioritisation approach in agile-scrum method," *J. Adv. Res. Appl. Sci. Eng. Technol.*, vol. 47, no. 2, pp. 76–93, May 2024, doi: 10.37934/araset.47.2.7693.

[65] L. Karlsson, T. Thelin, B. Regnell, P. Berander, and C. Wohlin, "Pair-wise comparisons versus planning game partitioning–experiments on requirements prioritisation techniques," *Empirical Softw. Eng.*, vol. 12, no. 1, pp. 3–33, Feb. 2007, doi: 10.1007/s10664-006-7240-4.

[66] J. Azar, R. Smith, and D. Cordes, "Value-oriented requirements prioritization in a small development organization," *IEEE Softw.*, vol. 24, no. 1, pp. 32–37, Jan. 2007.

[67] M. Khari and N. S. Kumar, "Comparison of six prioritization techniques for software requirements," *J. Global Res. Comput. Sci.*, vol. 4, no. 1, pp. 38–43, Feb. 2013. [Online]. Available: www.jgrcs.info

[68] Z. Racheva, M. Daneva, and L. Buglione, "Supporting the dynamic reprioritization of requirements in agile development of software products," in *Proc. 2nd Int. Workshop Softw. Product Manage.*, Sep. 2008, pp. 49–57, doi: 10.1109/IWSPM.2008.7.

[69] S. Devulapalli, O. R. S. Rao, and A. Khare, "Comparison of ABC framework with AHP, wiegers method, cost-value, priority groups for requirements prioritization," in *Proceedings of International Conference on Communication and Networks* (Advances in Intelligent Systems and Computing). Berlin, Germany: Springer, 2017, pp. 591–600, doi: 10.1007/978-981-10-2750-5_61.

[70] L. Lehtola and M. Kauppinen, "Empirical evaluation of two requirements prioritization methods in product development projects," in *Proc. Eur. Conf. Softw. Process Improvement*, Jan. 2004, pp. 161–170.

[71] A. Hassan and N. Ramadan, "A fuzzy approach for Wieger's method to rank priorities in requirement engineering," *CIIT Int. J. Fuzzy Syst.*, vol. 9, pp. 189–196, Nov. 2017. [Online]. Available: https://www.researchgate.net/publication/322695979

[72] P. Voola and A. V. Babu, "Requirements uncertainty prioritization approach: A novel approach for requirements prioritization," *Softw. Eng. Int. J.*, vol. 2, no. 2, pp. 37–49, Sep. 2012. [Online]. Available: http://www.inf.kiev.ua/GMDH-home/GMDH_abo.htm

[73] R. Beg, Q. Abbas, and R. P. Verma, "An approach for requirement prioritization using B-tree," in *Proc. 1st Int. Conf. Emerg. Trends Eng. Technol.*, 2008, pp. 1216–1221, doi: 10.1109/ICETET.2008.158.

[74] R. V. Anand and M. Dinakaran, "Multi-voting and binary search tree-based requirements prioritisation for e-service software project development," *Electron. Government*, vol. 13, no. 2, pp. 111–128, 2017.

[75] A. Ahmad, A. Shahzad, V. K. Padmanabhuni, A. Mansoor, S. Joseph, and Z. Arshad, "Requirements prioritization with respect to geographically distributed stakeholders," in *Proc. IEEE Int. Conf. Comput. Sci. Autom. Eng.*, vol. 4, Jun. 2011, pp. 290–294.

[76] A. Ionica, M. Leba, and R. Dovleac, "A QFD based model integration in agile software development," in *Proc. 12th Iberian Conf. Inf. Syst. Technol. (CISTI)*, Jun. 2017, pp. 1–6.

[77] A. Ahmadzadeh, A. S. Aboumasoudi, A. Shahin, and H. Teimouri, "Developing a QFD model for prioritizing the CSFs of ERP based on the enablers of organizational agility," *Benchmarking Int. J.*, vol. 28, no. 4, pp. 1164–1185, Apr. 2021, doi: 10.1108/bij-08-2020-0411.

[78] A. Rasheed, B. Zafar, T. Shehryar, N. A. Aslam, M. Sajid, N. Ali, S. H. Dar, and S. Khalid, "Requirement engineering challenges in agile software development," vol. 2021, no. 1, 2021, Art. no. 6696695, doi: 10.1155/2021/6696695.

[79] F. Hujainah, R. B. A. Bakar, M. A. Abdulgabber, and K. Z. Zamli, "Software requirements prioritisation: A systematic literature review on significance, stakeholders, techniques and challenges," *IEEE Access*, vol. 6, pp. 71497–71523, 2018, doi: 10.1109/ACCESS.2018.2881755.

[80] J. Karlsson, C. Wohlin, and B. Regnell, "An evaluation of methods for prioritizing software requirements," *Inf. Softw. Technol.*, vol. 39, nos. 14–15, pp. 939–947, Jan. 1998.

[81] Z. Shehzadi, F. Azam, M. W. Anwar, and I. Qasim, "A novel framework for change requirement management (CRM) in agile software development (ASD)," in *Proc. 9th Int. Conf. Inf. Commun. Manage.*, Aug. 2019, pp. 22–26, doi: 10.1145/3357419.3357438.

[82] Z. Hoy and M. Xu, "Agile software requirements engineering challenges-solutions–A conceptual framework from systematic literature review," *Information*, vol. 14, no. 6, p. 322, Jun. 2023, doi: 10.3390/info14060322.

[83] D. Lloyd, R. Moawad, and M. Kadry, "A supporting tool for requirements change management in distributed agile development," *Future Comput. Informat. J.*, vol. 2, no. 1, pp. 1–9, Jun. 2017, doi: 10.1016/j.fcij.2017.04.001.

[84] V. Gupta, D. S. Chauhan, and K. Dutta, "Exploring reprioritization through systematic literature surveys and case studies," *SpringerPlus*, vol. 4, no. 1, p. 539, Sep. 2015, doi: 10.1186/s40064-015-1320-0.

[85] A. Jarzebowicz and P. Weichbroth, "A qualitative study on non-functional requirements in agile software development," *IEEE Access*, vol. 9, pp. 40458–40475, 2021, doi: 10.1109/ACCESS.2021.3064424.

[86] A. Muhammad, A. Siddique, M. Mubasher, A. Aldweesh, and Q. N. Naveed, "Prioritizing non-functional requirements in agile process using multi criteria decision making analysis," *IEEE Access*, vol. 11, pp. 24631–24654, 2023, doi: 10.1109/ACCESS.2023.3253771.

[87] S. Kopczyńska, M. Ochodek, and J. Nawrocki, "On importance of non-functional requirements in agile software projects—A survey," in *Integrating Research and Practice in Software Engineering* (Studies in Computational Intelligence), vol. 851. Berlin, Germany: Springer, 2020, pp. 145–158, doi: 10.1007/978-3-030-26574-8_11.

[88] R. V. Anand and M. Dinakaran, "Handling stakeholder conflict by agile requirement prioritization using apriori technique," *Comput. Electr. Eng.*, vol. 61, pp. 126–136, Jul. 2017, doi: 10.1016/j.compeleceng.2017.06.022.

[89] A. Perini, A. Susi, and P. Avesani, "A machine learning approach to software requirements prioritization," *IEEE Trans. Softw. Eng.*, vol. 39, no. 4, pp. 445–461, Apr. 2013, doi: 10.1109/TSE.2012.52.

[90] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. N. O. Dag, "An industrial survey of requirements interdependencies in software product release planning," in *Proc. 5th IEEE Int. Symp. Requirements Eng.*, Aug. 2001, pp. 84–91, doi: 10.1109/ISRE.2001.948547.

[91] F. Noviyanto, R. Razali, and M. Z. A. Nazree, "Understanding requirements dependency in requirements prioritization: A systematic literature review," *Int. J. Adv. Intell. Informat.*, vol. 9, no. 2, pp. 249–272, Jul. 2023, doi: 10.26555/ijain.v9i2.1082.

[92] Å. G. Dahlstedt, "What about these requirements interdependencies?-A proposal for future research," Univ. Skövde, Skövde, Sweden, 2003.

[93] I. Nurdiani, R. Jabangwe, and K. Petersen, "Practices and challenges of managing requirements interdependencies in agile software development: A survey," in *Proc. Int. Conf. Eng., Technol. Innov./IEEE Int. Technol. Manage. Conf. (ICE/ITMC)*, Jun. 2016, pp. 1–8, doi: 10.1109/ICE/ITMC39735.2016.9025919.

[94] K. Liu, S. Reddivari, and K. Reddivari, "Artificial intelligence in software requirements engineering: State-of-the-art," in *Proc. IEEE 23rd Int. Conf. Inf. Reuse Integr. Data Sci. (IRI)*, Piscataway, NJ, USA. Institute of Electrical and Electronics Engineers, Aug. 2022, pp. 106–111, doi: 10.1109/IRI54793.2022.00034.
[95] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965.
[96] M. Sadiq and S. K. Jain, "Applying fuzzy preference relation for requirements prioritization in goal oriented requirements elicitation process," *Int. J. Syst. Assurance Eng. Manage.*, vol. 5, no. 4, pp. 711–723, Dec. 2014, doi: 10.1007/s13198-014-0236-3.
[97] M. Ramzan, M. A. Jaffar, M. A. Iqbal, S. Anwar, and A. A. Shahid, "Value based fuzzy requirement prioritization and its evaluation framework," in *Proc. 4th Int. Conf. Innov. Comput., Inf. Control (ICICIC)*, Dec. 2009, pp. 1464–1468.
[98] J. Dąbrowski, E. Letier, A. Perini, and A. Susi, "Analysing app reviews for software engineering: a systematic literature review," *Empirical Softw. Eng.*, vol. 27, p. 43, Mar. 2022, doi: 10.1007/s10664-022-10135-4.
[99] A. Fatima, A. Fernandes, D. Egan, and C. Luca, "Software requirements prioritisation using machine learning," in *Proc. 15th Int. Conf. Agents Artif. Intell.*, 2023, pp. 893–900, doi: 10.5220/0011796900003393.
[100] A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley, "The next release problem," *Inf. Softw. Technol.*, vol. 43, no. 14, pp. 883–890, Dec. 2001. [Online]. Available: www.elsevier.com/locate/infsof
[101] S. Jayatilleke and R. Lai, "A systematic review of requirements change management," *Inf. Softw. Technol.*, vol. 93, pp. 163–185, Jan. 2018, doi: 10.1016/j.infsof.2017.09.004.
[102] M. Shafiq, Q. Zhang, M. A. Akbar, A. A. Khan, S. Hussain, F. E. Amin, A. Khan, and A. A. Soofi, "Effect of project management in requirements engineering and requirements change management processes for global software development," *IEEE Access*, vol. 6, pp. 25747–25763, 2018, doi: 10.1109/ACCESS.2018.2834473.
[103] D. Pandey, U. Suman, and A. K. Ramani, "An effective requirement engineering process model for software development and requirements management," in *Proc. Int. Conf. Adv. Recent Technol. Commun. Comput.*, Oct. 2010, pp. 287–291, doi: 10.1109/ARTCom.2010.24.
[104] Z. Racheva, M. Daneva, A. Herrmann, and R. J. Wieringa, "A conceptual model and process for client-driven agile requirements prioritization," in *Proc. 4th Int. Conf. Res. Challenges Inf. Sci. (RCIS)*, May 2010, pp. 287–298, doi: 10.1109/RCIS.2010.5507388.
[105] Z. Racheva and M. Daneva, "Reprioritizing the requirements in agile software development: Towards a conceptual model from clients' perspective," in *Proc. 21st Int. Conf. Softw. Eng. Knowl. Eng.*, Jul. 2009, pp. 73–80. [Online]. Available: https://www.researchgate.net/publication/221391125
[106] X. Zhou, Y. Jin, H. Zhang, S. Li, and X. Huang, "A map of threats to validity of systematic literature reviews in software engineering," in *Proc. 23rd Asia–Pacific Softw. Eng. Conf. (APSEC)*, Dec. 2016, pp. 153–160, doi: 10.1109/APSEC.2016.031.

NOSHIN TASNEEM was born in Dhaka, Bangladesh, in 1998. She received the B.Sc. degree from the Computer Science and Engineering Department, Jahangirnagar University, Savar, Dhaka, Bangladesh. She is currently pursuing the M.Sc. degree with the Department of Software Engineering, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). Her research interests include software engineering, requirement engineering, and agile software development.

HAZURA BINTI ZULZALIL received the B.Sc. degree in computer science and the M.Sc. and Ph.D. degrees in software engineering from Universiti Putra Malaysia (UPM). She is currently an Associate Professor with the Faculty of Computer Science and Information Technology, UPM. Her research interests include software metrics, software quality, and software evaluation.

SA'ADAH HASSAN received the B.Comp.Sc. degree from Universiti Teknologi Malaysia, the master's degree in software engineering from the University of Malaya, and the Ph.D. degree from Ulster University, U.K. She is currently an Associate Professor with Universiti Putra Malaysia. Her research interests include software engineering, intelligent systems, and management information systems.