IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 21, 2021, accepted October 1, 2021, date of publication October 4, 2021, date of current version October 13, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3117989

Challenges of Software Requirements Quality Assurance and Validation: A Systematic Literature Review

ISSA ATOUM[1], MAHMOUD KHALID BAKLIZI[1], IZZAT ALSMADI[2], AHMED ALI OTOOM[3], TAHA ALHERSH[4], JAFAR ABABNEH[1], JAMEEL ALMALKI[5], AND SAEED MASOUD ALSHAHRANI[6]

[1]Faculty of Information Technology, The World Islamic Sciences and Education, Amman 11947, Jordan
[2]Department of Computing and Cyber Security, Texas A&M University-San Antonio, San Antonio, TX 78224, USA
[3]Faculty of Science and Information Technology, Irbid National University, Irbid 21110, Jordan
[4]BioQuant, Heidelberg University, 9117 Heidelberg, Germany
[5]Department of Computer Science, College of Computer in Al-Leith, Umm Al-Qura University, Mecca 24244, Saudi Arabia
[6]Department of Computer Science, College of Computing and Information Technology, Shaqra University, Shaqra 11961, Saudi Arabia

Corresponding author: Issa Atoum (issa.atoum@wise.edu.jo)

ABSTRACT Validation of software requirements is a primary phase in requirements engineering that ensures requirements match the target system with the intended needs of the acquirer. It aims to detect and correct errors that prevail in the specified requirements. Although there are tremendous requirements validation approaches, some software may fail because of limited or ineffective requirements validation techniques and unreliable requirements' quality characteristics. In this study, a systematic literature review of requirements validation is performed. The study analyzes the most adopted validation techniques, reports requirements quality characteristics, and discovers significant challenges of validation techniques. The review identified 66 relevant primary studies analyzed to derive deep insights into the following aspects of requirements validation: trends of requirements validation methods, including their subtechnique strengths and weaknesses, requirements quality characteristics categories, and adopted tools and datasets in these techniques. We grouped validation techniques into categories: prototyping, inspection, knowledge-oriented, test-oriented, modeling and assessment, and formal models. The analysis reported 19 validation techniques, 27 tools, new requirements validation characteristics, and several challenges that prevailed through validation techniques. The trend of validation techniques is to those methods that apply machine learning techniques with knowledge from dictionaries and ontologies. Most challenges are about how to express the requirements and how to revert clients' feedback. There is a strong relationship between validation techniques, software application domain, and requirements validation quality attributes. Thus, there is an immense need to unify the quality characteristics and domain-specific validation methods.

INDEX TERMS Requirements validation, requirements analysis, requirements engineering, validation techniques, requirements quality, systematic literature review.

I. INTRODUCTION

Requirements engineering (RE) involves discovering (gathering), eliciting, analyzing, evaluating and negotiating, documenting, verifying and validating, and managing requirements [1]. Research has shown that most software failures are related to RE; most issues are triggered because of ''inconsistent'' or hidden requirements [2]. As requirements are gathered from multiple stakeholders, they are commonly documented in a natural language to ease communication between stakeholders in a common textual language form. However, natural languages are prone to semantic and synaptic errors, as they do not adhere to a complete formal semantic grammar [S62], [S63]. Consequently, requirements written in a textual form, such as initial requirements or user stories [3], are prone to errors that may lead to unacceptable systems. Additionally, the requirements specified in other diagrammatic notations, such as use cases [S15], sequence

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

diagrams [S21], and behavior interaction diagrams [S08], are difficult to comprehend by nontechnical users. Moreover, the requirements specification notations depend on the system type, application domain, requirements hierarchy, and the specification's role as a communication medium between stakeholders [4]. Thus, requirements validation is required with various specification notations.

To this end and for clarity, we refer to RE as a combination of several interleaved and recursive processes [5] that are not separated [6]. We follow the classification as identified by [7], which includes elicitation, modeling, analysis, and verification and validation, which is also relatively similar to the grouping in [8]. Nuseibeh and Easterbrook [9] grouped the RE process, that is, eliciting requirements, modeling and analyzing requirements, communicating requirements, agreeing requirements, and evolving requirements, whereas Bennaceur *et al.* [10] used the groups' requirements elicitation, modeling, assurance, and management. This issue of inconsistency in research methods and terminology has recently been highlighted [11].

This paper focuses on requirements analysis at the early stages of software development before an actual application is coded or even before the requirements are approved. The requirements analysis—(also referred to as validation in many references [12]–[14])—is the process of identifying errors to increase the quality of the requirements. Errors could be at the document level, context level, and agreement between stakeholders [14]. Thus, we use the two terms (analysis or validation) interchangeably to mean ways to enhance software requirement assurance. Consequently, we focus on requirements analysis and validation as described in [7] and follow the ''agreeing requirements'' described by Nuseibeh and Easterbrook [9]. However, early requirements analysis during the elicitation analysis [15], [16] or discussing requirements analysis for purposes other than software development [17] is outside the scope of this work.

Studies have reported that requirement defects constitute 32.65% of the significant sources of failure [S02]. Despite the ever-increasing research on RE, little research has been conducted on requirements validation. The literature reports many RE books, journal articles, and relevant standards; however, when it comes to requirements validation, no agreed set of quality characteristics is universally adopted to support requirements validations [S11], [S15], [S26]. It is also crucial to automate validation techniques using tools because of requirements subjectivity and limited tools [S29].

Thus, motivated by the software requirements research community, our primary objective is to analyze requirements validation techniques and the quality of their accompanying requirements characteristics to increase the overall software quality. Thus, the essential objectives of this study are as follows:

(1) Identify state-of-the-art requirements validation techniques and their weaknesses and strengths.
(2) Analyze requirements checking characteristics.
(3) Report tools and datasets that should support requirements analysis.

The main research question is as follows: What are the main challenges in requirements validation techniques in the context of different application domains?

The remainder of this paper is organized as follows. Section II discusses the background of this study. Section III presents the study's research methodology, which is applied in Section IV. Section V discusses the challenges of requirements validation. Section VI explains the most critical research challenges to requirements validation. Finally, Section VII provides the conclusions and directions for future research.

II. BACKGROUND

The requirements validation conceptual process is contextualized by requirements, organizational knowledge, and the system context [S33]. Organizational knowledge includes policies, lessons learned, requirements repository, requirements document templates, regularities, and legal aspects. Ignoring the context of the intended system may complicate the validation process and lead to unintended consequences [18]. The validation process outputs a list of discovered errors, agreed actions regarding each error, and recommendations. For example, stakeholders may report hidden requirements (incompleteness) or contradicting requirements (inconsistencies). The list of agreed actions states corrective actions that should be taken to fix the detected problems. Consequently, the requirements analysis enables requirements conformance to the stated or implied threshold level of predefined quality indicators or characteristics [13].

In the requirements validation process, software engineers and stakeholders perform various checks (also called quality detection) to check requirements conformance to specified quality criteria. A study of 26 large financial systems showed that 44% of the analyzed requirements had errors due to specified requirements for subjective language (34.6%) [19]. The most commonly used requirements characteristics are the 3C's (completeness, correctness, and consistency). According to research [20], incomplete and incorrect requirements are marginally quality characteristics (23.5%; 35.3%), whereas ''inconsistent'' requirements cover 5.9% of the cases. Additionally, ambiguity is one of the most reported defects in requirements validation and is considered an inherent problem of natural languages [S63]. For example, the Oxford English dictionary reported that 23 meanings were used in the top 500 words. Although the research community expands the list of requirements characteristics starting from eight characteristics [1], [21] to 15 characteristics [22], it is common to have a different definition of the same quality characteristic, possibly with a different name. However, the number and definitions of the characteristics vary [1], [21], [22]. Table 1 presents a sample of these checks.

Consequently, the problems related to requirements validation could be solved with research that aims to understand

and automate the validation process with tools reducing the prevailed errors in requirements. The purpose of this study is to enhance the research community's understanding of requirements validation at early software development. One of the research community goals is to measure the significant validation quality characteristics that result in a suitable and high-quality product. The research is considered significant as it is one of the latest systematic literature reviews (SLRs) in software requirements validation.

III. RESEARCH METHOD

As shown in Section II, requirements validation touches many aspects of RE, including software quality, software development, and software modeling. Consequently, exploring the validation dimensions is discussed via an SLR. SLR techniques allow the exploration, evaluation, and interpretation of research relevant to specific research questions. Therefore, an SLR is an excellent choice for summarizing research, identifying research gaps, and laying the ground for new research [23].

The research method adopted in this research is based on Kitchenham and Charters [23] , which has three major stages: planning, conducting, and reporting.

The first stage highlights the need for review and development of the review protocol. We discuss in this stage the research questions that drive the search protocol and the search strategy. The conducting stage explains the study selection and the data extraction. Finally, the reporting stage provides the results and analyses of our review.

TABLE 1. Requirements checking characteristics [24], [10].

Requirement characteristic check	Description
Completeness	Completeness checks ensure that a requirement document should include all the requirements and their accompanying constraints.
Consistency	Consistency checks ensure that the requirements items should not conflict or have a different description of the same software feature.
Correctness	Correctness (also called adequacy or validity) indicates an acceptable degree of mutual understanding of requirements, which often implies compliance to policies, standards, and laws[10].
Validity	Validity checks ensure that the functions proposed by stakeholders should be aligned with what the system needs to perform without any additional functions.
Realism	Realism includes requirements that are achievable on the basis of project constraints.
Ambiguity	Ambiguity describes the degree of uncertainty of information relative to events or confidence in information quality and reliability[25]. Unambiguous requirements ensure that there is one interpretation of the requirements by stakeholders.
Verifiability	Verifiability checking ensures that requirements are specified to be tested, demonstrating that the system meets the specified requirements.

IV. APPLICATION OF THE RESEARCH METHOD

This section details the adopted research method.

TABLE 2. Research questions rationale.

Research question	Rationale
RQ1: What are the state-of-art requirements validation methods, and what are their strengths and weaknesses?	Identify state-of-the-art validation approaches that are commonly used for requirements validation. Moreover, the research question examines the drawbacks and limitations of each approach.
RQ2: What are the associated requirements quality factors with the validation techniques?	Identify which quality factors are used for requirements validation and their role in increasing the overall software quality.
RQ3: What are the most common software applications and the distribution of software process models covered by validation techniques?	Identify lacking research areas on application areas or process models.
RQ4: What are the software supporting tools and datasets that are applied in requirements validation?	Identify adopted requirements validation tools and datasets used to validate the requirements.

A. PLANNING STAGE

The first phase in the adopted systematic literature review includes the research questions, related work, search strategy, and search protocol.

1) RESEARCH QUESTIONS

Our research questions investigate the discrepancy between software requirements quality characteristics and the most prominent requirements validation technique in reviewed application domains. We specify the research questions (IV) that guide the research string and the inclusion/exclusion criteria. The overall rationale is to discuss the main challenges concerning requirements validation in various software domains.

2) SLRs IN REQUIREMENTS VALIDATION

This research identified several software requirements validation techniques, quality requirements characteristics, and tools (Table 2). Regardless of how the requirements were specified, we aim to discuss the techniques and tools used for validation [26].

The work of Moketar *et al.* [27] is the one most related to this study. It investigates the requirements approaches and tools that are commonly used in the validation process. They found that prototyping was the most used requirements validation technique, and the 3C's are still the most prominent requirements quality characteristics. Our approach expands and reinvestigates the techniques and quality factors complimentary with previous research.

3) SEARCH STRATEGY

First, we decided to find papers focused on software requirements; thus, this study adopted the inclusion and exclusion criteria in Table 4. Second, the authors applied the title, abstract, and keyword analysis criteria in continuous research meetings, laying the foundation for consistent information

TABLE 3. Systematic literature reviews of software requirements validation.

Source	Goal	Related research question	Study period
[27]	The study reports requirements validation techniques, tools, and quality requirements factors.	(1) "What was the current trend in requirements validation approaches/technique/tool that was applied or proposed in the studies? " (2) "What are the quality criteria of requirements that were validated in the studies?"	2007–2016
[28]	Discover how the software requirements engineering process is used in software startups.	"How does the software startup validate requirements?"	2000–2016
[29]		"What areas in requirement engineering are addressed, and how many articles cover the different areas?"	2015–2020
[30]	Examine used approaches for eliciting, modeling, specifying, validating safety-critical systems' requirements, and investigating the reported approaches' usefulness.	(1) "What approaches have been proposed to elicit, model, specify or validate safety requirements in the context of safety-critical systems?"	1983–2014

TABLE 4. Inclusion and exclusion criteria.

Criteria	Type
(1) Studies from January 2016 until the end of December 2020	Inclusion
(2) Peer-reviewed studies related to the defined search string	Inclusion
(3) Studies that discuss requirements engineering validation concerning the RQs	Inclusion
(4) Studies must be written in the English language	Inclusion
(5) Studies published in posters, letters, abstracts, reference works, and standards	Exclusion
(6) Literature review studies	Exclusion
(7) Studies that are not written in the English language	Exclusion
(8) Studies that show validation techniques by surveys from practitioners	Exclusion
(9) Papers that do not pass the quality assessment	Exclusion

extraction. Consequently, following the criteria in Table 4, studies describing software development frameworks are not considered unless they have sufficient details about requirements validation (e.g., [31]).

4) SEARCH PROTOCOL

The current work builds upon previous work of Besrour *et al.* [32], which identifies critical challenges in Software Engineering (1995–2016); Moketar *et al.* [27], which shows trends in tools of requirements validation (2007–2016); and Zogaan *et al.* [33], which discusses datasets of software requirements. Hence, the search strings for screening studies were limited for studies from January 2016 to December 2020. The search string was developed according to the following criteria: (1) careful analysis of the ISO 29148:2018 standard for potential keywords; (2) a sample of papers from the Requirements Engineering Journal as it is specialized in RE; and (3) informal search keywords on Google Scholar. The search string was used with little modification to include synonyms and to handle the requirements of the adopted databases. Moreover, snowballing was added to reduce the potential for missing reference papers. Snowballing technique enables to explore more studies by expanding the primary studies. Forward snowballing enables the addition of more studies based on the references section of a primary study, whereas the backward snowballing enables the acquisition of studies that cite a specific primary study.

Hence, we decided to use the following primary search string for several reasons.

(1) After several trials, including keywords that use terms from the research questions, they were not fruitful. For example, our findings showed that adding the expression "AND (challenges OR datasets OR Tools)" to the end of the primary search string would reduce the number of research results by approximately 20% and result in the string being unacceptable by some literature databases. Conversely, splitting the search query into more than one would retrieve a list of tools or datasets that do not comply with the fourth research question; tools and datasets must be used alongside a validation technique.
(2) We aim to find techniques and accompanied tools and datasets (if available); thus, finding tools and datasets in isolation of the validation technique does not comply with the research questions.
(3) There is an issue concerning the requirements quality characteristics keywords. It was not practical to include quality checking keywords (such as incompleteness, correctness, and ambiguity) because there is no agreement in the literature on the complete list of quality keywords. Moreover, the list is more extensive than the capabilities of the adopted database engines.

This study uses the following commonly used databases, which are also used in related works: ACM Digital Library, Springer Link, IEEE Xplore, and Science Direct. Consequently, the primary search string is as follows:

(requirements engineering) AND (validation OR assurance OR analysis OR quality) AND (application OR software)

5) THREATS TO VALIDITY

This study discusses reliability criteria, including external validity, internal validity, construct validity, and conclusion validity. External validity includes how the results can be generalized to other contexts. The collected papers come from well-known databases often used in software engineering research [30], [34]. One threat to validity is the restricted

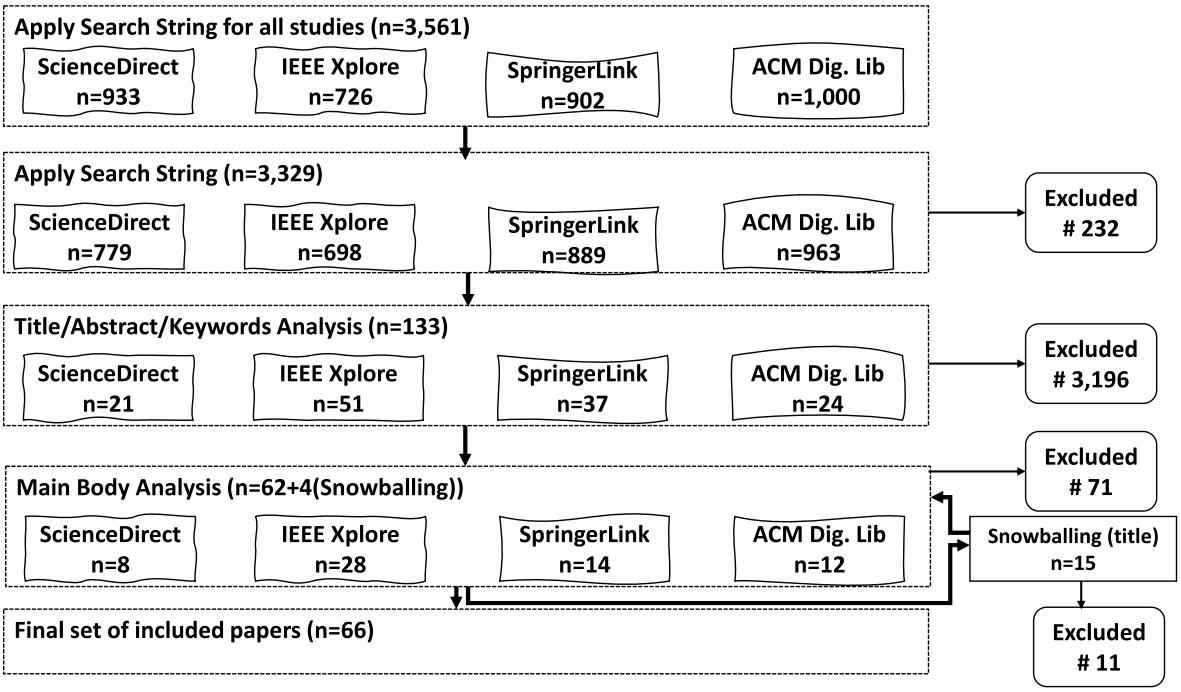

FIGURE 1. Database search diagram following stages of Kitchenham and Charters [23].

time search string interval, which was overcome by reporting related literature reviews matching research questions from the previous periods. The other threat is inadequate primary studies, which were reduced using the snowballing method.

Internal validity includes whether the evidence supports the truthfulness of the list of identified primary studies and the reported validation challenges. For example, one threat could be improper research methods, which were overcome using well-known databases for software engineering papers. Therefore, each paper was analyzed following the method in [24], including the complete paper body analysis during the conducting stage. Moreover, thematic analysis [35] was applied to synthesize the higher-order validation categories. Additionally, we applied a backward snowballing search to retrieve as many relevant primary studies as possible to mitigate any potential threat to any of the primary studies.

Construct validity ensures that the study concepts are measured to align with the research questions. One threat could be imprecise SLR, which was overcome with a review protocol [23] and existing LR tools. Additionally, the threat of incorrect keywords was overcome using keywords from ISO 29148:2018 and previous works that aligned with the RQs.

The validity of the conclusion confirms whether proper actions were taken to extract the literature. The literature review of primary studies was selected and extracted using a systematic literature review of software engineering [23]; the threat could be biased in the selection. It was reduced using literature review tools and reviewing studies several times on the basis of the researchers' experience. Another threat could be how primary studies were grouped, which was overcome following [35] guidelines. Another threat was the threat of replication, which was overcome using Mendeley and Microsoft Excel.

B. CONDUCTING SEARCH STAGE

We report the study selection, snowballing, article quality assessment, and data extraction and analysis to conduct the research.

1) STUDY SELECTION

Figure 1 illustrates the conducting phases of Kitchenham and Charters [23]. First, all studies' metadata, such as authors, abstracts, keywords, and publications information, were collected with Mendeley software, which was also used to remove duplicates. We then conducted the first two steps of our approach (Figure 1) using several research meetings. However, the third step was conducted in joint meetings conducted for a random set of 100 studies from different databases. The objective was to provide a consistent and reliable process in the future. Next, studies that were not selected by at least three authors were excluded from the subsequent phases. Finally, when the abstract analysis was insufficient, studies were delayed for a detailed analysis of the main body.

2) SNOWBALLING

Following the snowballing guidelines for systematic reviews of [36], the primary studies were expanded. We performed backward snowballing on the initial set of 62 studies using

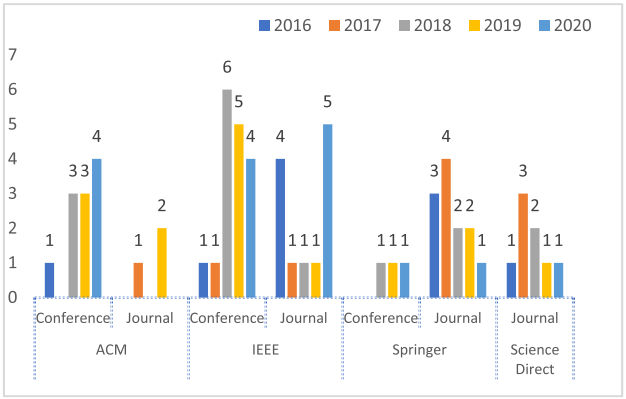

FIGURE 2. Studies distribution.

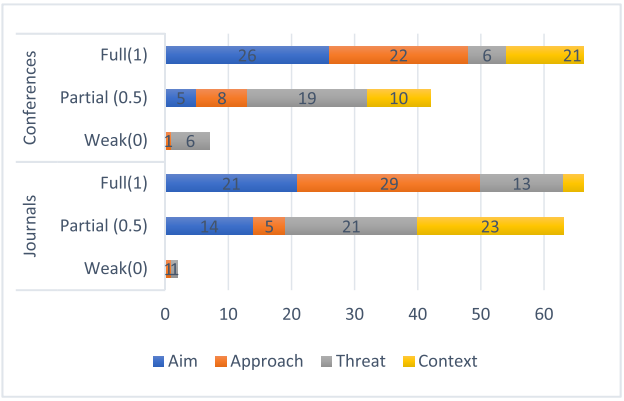

FIGURE 3. Quality assessment. conference and journal papers generally have a similar quality.

their reference lists. We adhered to the search protocol and the search string during the snowballing process. The snowballing results identified an initial of 15 new studies, four were duplicated [S09], [S17], [S24], [S39], and one was excluded because it was a literature review paper [28], whereas the rest did not pass the inclusion and exclusion criteria. Hence, the snowballing process resulted in four additional studies.

3) QUALITY ASSESSMENT

This study depends on rigor, relevant, and high-quality work [30], [34]; thus, we selected studies with clear and comprehensive validation techniques as they often provide sufficient details related to the research questions. The quality assessment criteria were as follows:

(1) The aim of the study should address the research questions (Rigor).
(2) The paper should have a consideration for threats to validity (Rigor).
(3) The presented approach should be transparent in the paper (Relevance).
(4) The context of the study should be with software RE (Relevance).

The quality assessment scores for each paper were yes = 1, partial = 0.5, and weak = 0. Thus, the minimum and maximum scores for rigor and relevance were in the range of 0–2. Studies that scored below 1 for both relevance and rigor were eliminated.

4) DATA EXTRACTION AND ANALYSIS PROCESS

Following our approach in Figure 1, the following information was extracted: all study metadata, including title, abstract, keywords, and publisher; relevant data related to the research questions; and challenges concerning each technique. The authors extracted this information from several joint sessions. We used Excel to collect relevant information and Mendeley to track citations using tags and notes.

Thematic analysis [35], an approach based on relationships and recurring patterns, was used to synthesize the identified themes related to requirements validation. First, the papers were carefully read to obtain a deeper understanding of the topics discussed. Then, important information related to the research questions was coded using content analysis. The codes were then converted into themes and higher-order themes. Next, each researcher analyzed one group of studies on the basis of their experience and early feedback on the previous step. As explained before, conducting 100 articles analysis on the basis of the article title, abstract, and keywords was completed jointly, which is expected to provide consistent information extraction. During this phase, the researchers discuss their findings and regularly provide consistent extractions. The researchers were not obliged to use any software tools; however, they provided patterns of papers they analyzed regularly. Finally, several meetings' sessions were held to synthesize and approve the final list of higher-order themes.

C. REPORTING STAGE

This research analyzes 66 primary studies in software requirements validation that directly address the research questions. Additionally, studies are discussed on the basis of these research questions. A list of all the primary studies is provided in Appendix A.

1) SUMMARY OF PRIMARY STUDIES

Figure 2 depicts the number of primary studies according to journal/conference versus year/database. The results show no strong evidence of a relative increase in articles by year. Expectedly, the most significant number of articles were from the Requirements Engineering Journal and the International Conference on Requirements Engineering.

Figure 3 shows the results of the quality assessment. The figure shows that the papers are not entirely focused on requirements validation. For example, the studies [S01], [S16] focus on requirements specification and validation, whereas most of the other works have a complete model, starting from requirements specification until validation (for example, [S01], [S13], [S14]). Nonetheless, as the figure shows, the conference and journal papers generally have

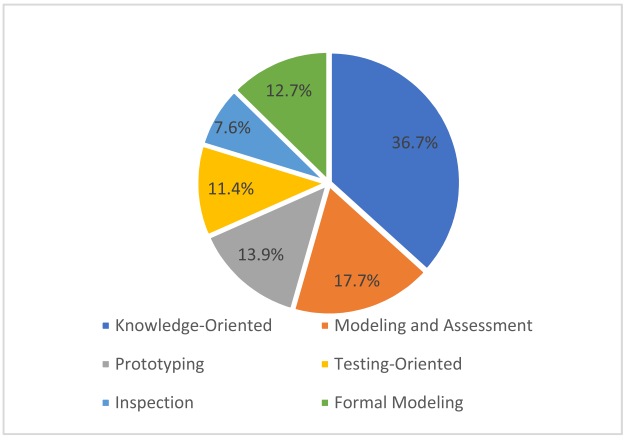

FIGURE 4. Validation techniques category distribution. Most of the validation techniques fall in knowledge-oriented methods.

a similar quality, except in some cases where the threat is unclear, often in conference papers. When the quality assessment was applied, the highest scores for relevance and rigor (2 points) were 41.8% and 22.4%, respectively. Only 32.1% of the studies had the highest scores (relevance = 2 and rigor = 2). When applied to each selected study, 83.6% achieved high relevance (≥1.5 points), and 67.2% achieved high rigor (≥1.5 points). Studies that scored below 1 point conjointly for rigor and relevance were eliminated.

This section assesses the validation techniques used to answer the first research question. For simplicity, in this study, a technique could be an approach, method, or framework. First, we summarize the validation techniques into logical groups: prototyping validation, knowledge-oriented validation, validation by inspection, testing-oriented validation, modeling and assessment, and formal modeling techniques. Table 5 presents the descriptions of each group. Please note that in Table 5, the category modeling and assessment are relatively different from ''model checking,'' which describes a formal method for verifying an abstract model relative to formal specification. However, in our context, we refer to modeling where requirements are specified and then assessed using structure diagrams (conceptual models), simulation models, goal-based models, and modified V&V frameworks.

Figure 4 shows that most of the validation techniques fall in knowledge-oriented methods (36.7%).

All techniques are discussed in the following subsections (2–7). Quality characteristics are discussed in subsection 8, whereas tools and datasets are discussed in subsection 9. Finally, the applications and software process models are discussed in subsection 10.

2) PROTOTYPING TECHNIQUES

Practical prototypes provide legitimate visualization, reducing customers' and developers' efforts [S05]. Visualization was used to help medical experts validate the requirements specifications [S09], [S62]. Studies have shown an increase in

TABLE 5. Validation techniques category descriptions.

Validation category	Description	Primary studies
Validation by prototyping	Prototyping is used when part of the software is built (mockup, UI) to provide early customer feedback about their understanding of the target system, especially when requirements are not precise.	[S05, S11, S15, S18, S20, S24, S29, S30, S47, S56, S62]
Knowledge -oriented validation	The knowledge-oriented approach is based on existing knowledge, such as ontologies and requirements datasets, to analyze the characteristics of the requirements for errors. Often, machine learning is used where requirements defect errors are reported.	[S02, S06, S07, S13, S14, S15, S17, S19, S21, S25, S27, S31, S32, S33, S34, S36, S37, S39, S40, S45, S51, S52, S53, S54, S61, S62, S63, S64, S66]
Inspection	Inspection is a type of validation where the requirements are checked against a predetermined review process. Inspectors find defects in requirements on the basis of their experience and are guided by inspection processes and tools.	[S01, S03, S04, S06, S26, S43]
Testing-oriented validation	Testing-oriented approaches are based on generating testing artifacts of the software system to validate requirements indirectly. If the generation of such artifacts was easy(for example, test cases), the requirements are of high-quality.	[S03, S20, S28, S46, S50, S59, S60, S62, S65]
Modeling and assessment	Modeling techniques describe cases where the requirements are specified to enforce required system properties in the first place. Then, a model checking approach, such as simulation, validates time-consuming or error-prone tasks, such as safety-critical systems or control software.	[S08, S12, S18, S21, S22, S23, S35, S44, S48, S49, S50, S55, S57, S58, S10]
Formal models	In formal models, the requirements are converted to a mathematical model to reason against a system property, usually, consistency and completeness. Finally, these models are mathematically proved.	[S09, S11, S16, S20, S29, S38, S41, S42, S65]

autogenerated prototypes [S05], [S11], [S18], [S47], reducing development rework and providing early detection of requirements errors. By contrast, semimanual mockup prototypes were initially developed and implemented in gaming applications [S24]. However, the prototyping generation is not helpful by itself unless it is combined with other techniques. Our review showed that prototyping was adopted along with knowledge-oriented approaches [S62], modeling and assessment [S50], [S58], formal models [S29], and testing-oriented approaches [S20]. For example, the study [S29] applied a hybrid approach; the iUML-B models were used as a graphical model to generate Event-B models, whereas BMotion Studio was used as a concrete visualization of the HD machine. Thus, visualization techniques are

suitable for complex sequential processes with subprocess branches (e.g., preparation for hemodialysis).

The studied prototyping techniques have several challenges. First, there will be some prototyping issues, where developers might need to add their code [S05]. Moreover, prototyping a part of the requirements does not guarantee that the remaining requirements are intended [S20]. Moreover, hybrid prototyping methods with formal models only apply to specific domains (e.g., the signal temporal logic domain [S11]). Second, gaming applications and playability using a prototype are not entirely specified requirements validation. For large communities, requirements vary; hence, its validation is not trivial [S24]. Third, animators (as a prototype) may still fail to execute a specification in certain situations, where simulation might be needed to resolve the issue [S30]. Finally, simple prototyping with word coloring (or HTML form) does not provide enough feedback to stakeholders; instead, it might confuse them [S15].

3) KNOWLEDGE-ORIENTED TECHNIQUES

With existing language and domain knowledge, requirements errors can be discovered by finding related requirements concepts or indications of errors in a requirements document (or item). These methods eliminate the comprehension of large requirements documents, thus reducing the risk of failure [S02] and providing instant feedback to users [S14], [S21]. One benefit of the knowledge-oriented approach is that it provides immediate requirements feedback and provides stakeholders with shared guidelines for requirements artifacts [S06], [S17]. For instance, the study [S15] showed that they could provide 83% accuracy compared to users who achieved only 47% accuracy in error discovery.

There are two general approaches in knowledge-oriented methods: rule-based and machine learning methods. The first depends on expert rules (i.e., knowledge engineering) to identify the requirements' errors without model training. Moreover, the rules ensure requirements quality on the basis of expertly crafted patterns [S14], [S15] or writing rules [S17], [S21]. By contrast, machine learning extracts knowledge from trained models on the requirements. For example, ontologies can capture generic static domain knowledge and can be reused to share knowledge across stakeholders [S07]. Additionally, with the help of ontologies, controlled vocabularies facilitate the automation of RE defect identification that can be used to reduce requirements conflicts [S40] and ambiguities [S51], [S63]. Additionally, it was reported that feature diagrams enable stakeholders to control their perspectives and preferences concerning the software [S31].

The analysis shows that knowledge-oriented techniques are tightly linked to automated tools. With automated tools in knowledge-oriented techniques, conflicts in requirements could be discovered based on resources, concepts contradictions, directions to society laws, and operation matters [S27]. Other automated tools focus on predicting variabilities [S32], vulnerabilities [S19], inconsistencies [S33], and conflicting viewpoints [S45]. Although many automated tools are in this category, most are not yet mature to reach 100% recall performance [S14]. Moreover, they do not fully automate the error resolution, rather than detecting errors [S15]. It is not easy to generalize the results to different software contexts [S17], [S21] because tools usually target one to three quality characteristics. Additionally, there are issues related to the current techniques; for example, it is unclear how the tension matrix is automated in different situations (validations) and different domains [S27]. Additionally, it is unclear how stakeholder feedback affects current system requirements [S31]. Integrating ontologies for requirement classification and validation can help in generating future porotypes [S07]. Thus, there is much work in this category; however, to provide better results, further enhancement is needed.

4) INSPECTION TECHNIQUES

Requirements review is the most straightforward inspection technique [S17]. The review technique ensures the discovery and solution of errors during the RE process [S26]. Checklists are a communal inspection method that provides consistency between compared requirements defects [S01] and saves time by following a strict process. It also allows end users to provide feedback between domain experts and system modelers [S26]. Questionnaires (a subtechnique of inspection) allow feedback from many stakeholders; however, they must be carefully built.

However, checklists or questionnaires are domain dependent and must be developed based on project types [S04]. Furthermore, once a requirement error is discovered, the stakeholders' role in discovering the solution might not be apparent in each technique [S01]. Additionally, it was found that these methods are not among the top state-of-the-art methods because of the difficulty in automating the inspection process, which is often executed semi manually.

5) TESTING-ORIENTED TECHNIQUES

The testing-oriented technique checks if the use cases built from the requirement are helpful and easy to generate [S65]. If generating test cases is difficult or impossible, the requirements eventually lack information, are incomplete, hard to be verifiable or modifiable, or are not measurable. Test case generation provides early user feedback and guides software developers to understand the system behavior [S28]. Generally, automated methods allow the generation of prototypes combined with the testing-oriented method, thereby providing stakeholders with the rationale of their intended needs [S20], [S46], [S59], [S62].

However, the methods in this group usually generate a test case from the developer's viewpoint; thus, requirement validation might be unsuccessful without collaboration between customers and developers. In other words, customers must have combined knowledge of business and some level of technical experience. Moreover, validating complex requirements (e.g., usability or quality requirements) is difficult because of limited customer competence in RE and their time and involvement constraints [S60]. Thus, human judgment

is needed to analyze requirements, for example, by ''use and misuse'' diagrams that depend heavily on the software engineer's experience [S28].

6) MODELING AND ASSESSMENT TECHNIQUES

This group specifies and validates the requirements as a complete solution for validation. For example, the Bayesian network model [S23] describes the quality characteristics of reused, unexpected dependencies, and variability, indicating the probabilistic validity of the input requirements. Built models provide early assurance of consistency between the requirements and design correctness; correct-by-construction techniques limit the need for a posteriori model checking [S08], [S22]. Combined with knowledge-oriented models, conceptual models (i.e., a subtechnique) can provide more meaningful domain knowledge to experts [S21]. In cases where building a prototype is complicated or in a safety model, simulation provides a solution to observe the system's behavior under changeable environmental conditions [S12]. Goal models are considered to have a higher level of requirement specification models representing the underlying customer goals (i.e., goal-oriented engineering). It also ensures that agents can maintain/achieve the required goals [S44], [S48], [S50], [S57], [S58]. For example, using psychologically driven goal models can interpret the meanings of human behavior to facilitate the validation of requirements [S48].

Some drawbacks of modeling and assessment are domain knowledge representation, design reusability, and scalability [S08]. Conversely, it should be clear which quality factors should be tested and expand use cases to generalize results [S12], [S21], [S22]. Finally, the most troublesome part is solving the errors or exacting the error rather than providing an error occurrence probability [S23]. However, goal models are not complete unless they are merged with testing-oriented models or other knowledge-oriented methods.

7) FORMAL MODELS

The requirements are converted to a formal mathematical model to test a system property; consistency and completeness are the main quality requirements tested in this approach. Although these quality attributes are merely used in requirements verification (as part of traceability), they can provide system validity insight. For example, the system [S16] allows the search for complete and consistent services of the debugging controller device. Some models provide a flexible language (signal temporal logic) to express and validate requirements [S20]. The formal model approach becomes valuable when used in models that can easily comprehend user diagrams [S38].

One major drawback of formal models is the expressiveness of the language requirements that users cannot easily understand. Furthermore, although formal models are robust, they are effortless for high-level stakeholders (managerial level) if no animation (or prototyping) is used. Additionally, not all scenarios (or requirements) are easily implemented. Hence, it might be difficult for users to validate them [S16], [S20].

8) REQUIREMENTS QUALITY

Checking the characteristics of the requirements would reduce potential requirement errors, indicating the rightfulness requirements. It has been shown that with language templates [S06], [S15], [S17], several characteristics such as ambiguity, consistency, and correctness can be predicted. Quality checks of requirements using formal models often target completeness [S16] and precision (correctness) [S20]. The interpretation and evaluation of these quality attributes vary, requiring flexible assessment and improvement techniques [S64]. To this end, a mapping in Table 6 is conducted on the basis of the collected characteristics (Column 1) and baseline requirements characteristics (Column 2). We chose the ground baseline, ISO 29148; however, Hull *et al.* [37] and Young [21] were used as alternatives when the characteristics were unavailable in ISO 29148. Hull *et al.* [37] and Young [21] lists are commonly cited as large lists of requirements characteristics. Note that four characteristics, namely, traceable, affordable, bounded, and implementation-free, were removed from ISO 29148:2018 when compared with ISO 29148:2011.

Figure 5 shows the percentage of the most used quality characteristics for requirements validation extracted from the primary studies. The top-quality characteristics discussed were correctness, completeness, consistency, and ambiguity. This finding is similar to previous findings that reported that completeness and inconsistency are the most critical qualities [38]. However, ambiguity is an additional quality characteristic, as discussed in the literature. Ambiguity is seen as a complex and multilevel quality attribute [39] that should be predicted in cross domains, especially for large scale applications [S66]. Conversely, additional characteristics that do not match the list of characteristics shown in Table 4 are understandability, reusability, unexpected dependencies, variability, and testability, indicating that the origin of the validation problem is not a compiled list of characteristics. Moreover, fine-grained (application-specific) quality characteristics include specificity [S23] and vacuity [S11]. Thus, the new characteristics were as low as 5.43% (5/92) (including repetitions). Hence, further research on these newly identified characteristics is required.

9) TOOLS AND DATASETS

Extracted information of tools and datasets (Appendix B) eliminates datasets not available to the public [S14], [S15], [S17], [S63]. Several tools depend on NLP and machine learning [S04], [S15], [S17], [S23], [S25]. Other tools are considered converter tools [S05], [S14], [S31], [S29], assistant tools [S06], [S21], or simulation tools [S20], [S22]. Notably, datasets were expressed in textual requirements [S06], [S17], [S25], use cases [S04], [S05], class diagrams [S04], and user stories [S14]. As stated earlier, most tools depend on machine learning; thus, expectedly, they

TABLE 6. Technique versus quality characteristics.

Requirements Quality Characteristic from Primary Studies	Equivalent Characteristic (ISO, Hull, Young)	Validation Technique	Total
Estimable	Feasible	S14	1
Conceptually Sound	Comprehensive	S14, S44	2
Unnecessary	Necessary/ Appropriate	S04, S14, S13	3
Problem-Oriented			
Overspecification			
Adequacy			
Valid	Able to be validated	S11	1
Completeness	Complete	S04,S05, S06,S12, S14,S15, S16,S17, S21,S23, S27,S03, S29,S32, S44,S61, S65	17
Specification completeness			
Incomplete References			
Incompleteness			
Omission			
Missing			
Unstated Feature			
The right level of detail			
Conflict-Free	Consistent	S04, S07, S08, S12, S14, S15, S18, S21, S22, S23, S27, S29, S30, S31, S33, S40, S44, S65	18
Consistency			
Conflicting strategies			
Description Homogeneity			
Mismatch			
Defect			
Correctness	Correct	S03,S05, S07,S08, S12,S13, S15,S21, S22,S28, S29,S31, S44,S64	14
Fault			
Independent	Singular	S14	1
Minimal			
Well-Formed	Conforming	S02,S14, S17,S21, S26,S37, S38	7
Uniform			
Full Sentence			
Compliance			
Risk-Mitigation			
Semantic Ambiguity	Unambiguous	S04, S08, S14, S17, S18, S20, S21, S26, S32, S33,	15
Open-Ended, Non-Verifiable Terms			
Plural Ambiguity			

TABLE 6. *(Continued.)* Technique versus quality characteristics[1].

Loopholes Semantic Relatedness		S51, S52, S63, S65, S66	
Modifier Ambiguity			
Negative Statements			
Referential Ambiguity			
Subjective Language			
Superlatives			
Comparatives			
Unambiguous			
Vague Pronouns			
Elliptic Ambiguity			
Conditional Clause, Reference Ambiguity			
Ambiguous Adverbs and Adjectives			
Lexical ambiguity			
Domain ambiguity			
Anaphoricambiguity			
Missing reference/Implicit requirements			
Specificity (S23), Vacuity(S11)	*	S23, S11	2
Traceable	Available in (Young list)	S29, S44, S15	3
Precise (S16), Redundant (S11), Unique (S14)	Available in (Hull list)	S11, S16, S14	3
Understandable	+	S32	1
Reused, Unexpected dependencies, Variability	+	S23	1
Testability	+	S44	1
Ground Total			90

[1]Note: * domain-specific characteristics, + additional characteristics, has a mapping in (Young list) [21], (Hull list) [37].

often adopt a dataset in the knowledge-oriented validation category. Next, all other techniques have relatively similar tools, except for inspection (one tool only), indicating a lack of automated tools for the inspection process.

The tools and datasets are detailed in Appendix B; however, we mentioned a few here. The Requirements to Prototype (RM2PT) tool [S05], [S47] helps to validate requirements by producing a prototype. Second, we have the requirements quality tools, for example, The Automated Inconsistency Checker (MaramaAIC) [S15] and Test-MEReq [S62] tools. MaramaAIC is considered an excellent

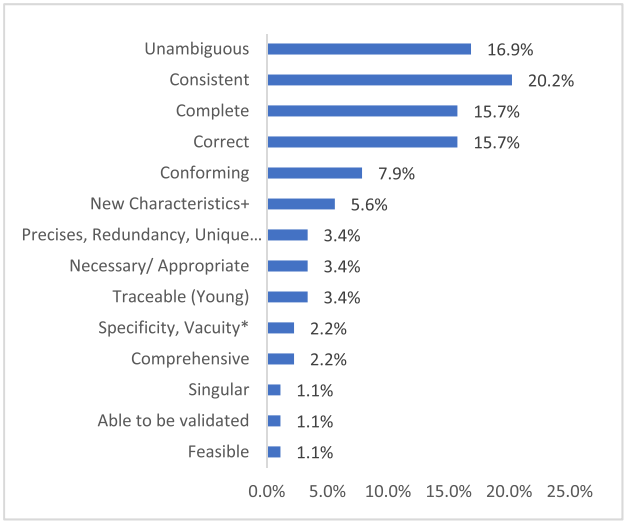

FIGURE 5. Distribution of studies across requirements quality characteristics. * domain-specific characteristics, + additional characteristics, has a mapping in (Young list) [21], (Hull list) [37].

tool to discover inconsistency between requirements, whereas TestMEReq is a comprehensive tool that includes correctness, completeness, and consistency. An advanced ambiguity detector tool in system design is The Adversarial Testing of Autonomous Vehicles (Sim-ATAV) [S20]. Sim-ATAV is a simulation-based adversarial testing framework for Cyber-Physical systems such as search-based test case generation and automated falsification for ambiguity detection.

However, there is a lack of datasets, where a few depend on structured requirements [S39] or user comments [S45], [S57].

Figure 6 shows the distribution of the primary studies on tools and datasets concerning techniques. It shows that (4.55% or three datasets) provided with this study has no specific tool [S39], [S45], [S57], whereas 15.15% of the studies were tools accompanied by datasets. The figure shows that more than half (54.55%) are techniques with no tools, indicating framework or approaches, tools not reported for the public, or using a programming language. Thus, requirements validation tools and datasets are scarce because developing such tools is time-consuming, especially fully automated ones. This finding is consistent with that only 39% of companies used recognized requirements management tools [40]. Consequently, there is a lack of datasets; our findings are consistent with previous work of Zogaan *et al.* [33]. However, many state-of-the-art studies on modeling and assessment still lack more tools than knowledge-oriented models. Thus, practitioners should select tools on the basis of their application capabilities [41].

10) SOFTWARE APPLICATIONS AND SOFTWARE PROCESS MODELS

Our selected studies revealed a list of 11 different application domains grouped on the basis of the application's generic functional behavior, as shown in Figure 7. Knowledge-oriented modeling and checking techniques dominate several

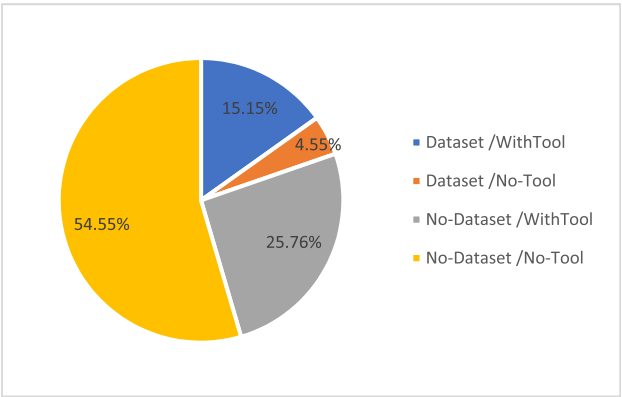

FIGURE 6. Datasets and tools distribution more than half of discussed techniques are not provided with public tools.

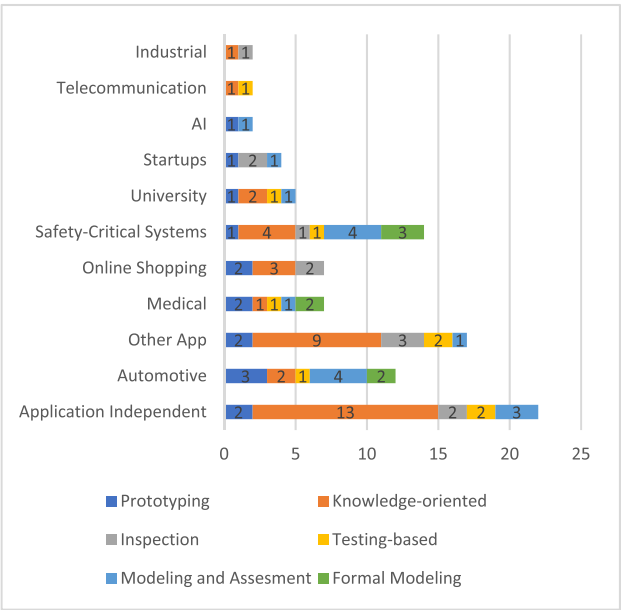

FIGURE 7. Distribution of validation techniques across application domains. Most tools are adopted in knowledge-oriented validation techniques.

applications (37.89% and 17.89%, respectively). Knowledge-oriented validation techniques have been adopted in nine out of 11 reported application domains, showing that machine learning models could validate many application domains [S15] instantly applying domain knowledge and ontologies. Similarly, modeling and assessment methods provide early information to users using models and checking techniques. Notably, the prototyping and inspection approaches have approximately 15.79% and 11.58%, respectively. Thus, this tendency might be related to knowledge-oriented methods. Nevertheless, users still need to foresee the target application through prototypes for better visualization [S29], [S30]. Hence, requirements validation models are recommended to take care of domain-specific properties [S06], [S11], [S23] and increase their applicability in other applications, especially in large scale integrated systems. However, the interpretation of the findings should

be interpreted cautiously, as the application grouping did not conform to any well-known standard.

Based on our primary studies, only 7.58% of validation techniques target the agile framework [S12], [S14], [S16], [S44], [S46], and approximately 15.15% target the nonagile process model [S03], [S05], [S18], [S20], [S23], [S26], [S28], [S29], [S31], [S51], [S52]. However, more than two-thirds of the studies (74.24%) did not clarify which approach was used, whereas one article mentions the use of both development approaches [S17]. Consequently, we infer that validation techniques are less applied in agile development, and several researchers do not consider the development process model while validating the requirements.

The primary studies that do not specify the development techniques indicate that the category does not have any relationship with software development. However, in agile development, validation complexity is related to stakeholders' expertise, requirements elicitation cycles, and change management [34]. The most important quality factors for agile requirements are simplicity and objectivity [42], which can be achieved by validating scenarios in patterns [S14]. The stakeholders' involvement in agile development is crucial as it encompasses stakeholders' vision and expectations, whereas validation is inherited in the agile development process [43]. With more stakeholder involvement, the quality of user stories has rarely been assessed. Thus, many of the studies target nonagile software process models because of the availability of requirements at early stages where completeness, correctness, and consistency could be detected along with the entire SRS.

V. CHALLENGES OF REQUIREMENTS VALIDATION IN THE CONTEXT OF REQUIREMENTS ENGINEERING

Compared with previous research of Moketar *et al.* [27], this study reported 51 techniques grouped into 19 methods, 27 tools, 15 datasets, and 19 quality characteristics (five new quality characteristics that were not discussed in previous studies). By contrast, Moketar *et al.* [27] reported 16 validation techniques, frameworks, and eight tools. Contrary to Moketar *et al.* [27], prototyping is not the leading requirements evaluation technique; however, autogenerative ones are promising. The present study and previous studies [27]–[29] insist on the need for further research on requirement validation.

A. REQUIREMENTS VALIDATION TECHNIQUES

Our investigation showed that 36.7% of the primary studies adopted knowledge-oriented techniques, whereas the lowest was inspection methods (7.6%). Despite the reported study that inspection could detect 50%–90% of requirements problems [44], it is infrequently discussed in the literature for applications where prototyping is favorable. One reason is that an automated inspection tool requires automated reasoning to inspect the SRS quality [S64]. It might also require a knowledge repository of SRS, quality metrics, and associated quality inspection checklists. Table 7 provides a further expanded list of weaknesses and strengths.

B. REQUIREMENTS QUALITY

We found no compromise agreement on the list of characteristics used to detect errors in requirements. However, reported errors could not explain defect resources, such as process execution errors, methods usage errors, or management errors. Management errors, for example, include errors that result from inadequate or insufficient management processes [S07]. To this end, finding requirements quality errors is primarily dependent on contextual factors [45].

Our study revealed that ambiguity is gaining attention, especially in natural language process models (e.g., [S04], [S08], [S14], [S17], [S18], [S20], [S21], [S26], [S32], [S33]). With language patterns, requirements can be transformed into semiformal specification domain ontologies, enabling a semiformal validation of requirements qualities [46]. The surveyed studies showed some characteristics that were not discussed in previous state-of-the-art validation studies (Table 8): understandability, reusability, unexpected dependencies, variability, and testability.

The characteristic of understandably indicates the end users' degree of cognitive load to figure out the implied or intended requirements [47]. It was reported that user stories specification has a limited impact on requirements quality and proposes a conceptual model that increases model understanding and reducing complexity [4]. Moreover, it was shown that extended UML case diagrams allow better game-related requirements representation, which will promote a common understanding among the development team [48].

The characteristics of variability and reusability are related to dependability. A variable system results in new ways to combine components, whereas the reusability of existing requirements results in new systems composed of new dependable components. Requirements variability is frequently coupled with Cyber-Physical systems that frequently evolve, resulting in new multicomponents of software configured differently [49]. Assuring safety requirements (or goals) results in an acceptable level of safety validation. A requirement defect is discovered if the system is doing the wrong thing. Variability is recently studied as a possible consequence of ambiguity based on syntactic rules of nouns and subordinate propositions [50].

Requirements reusability is critical to be validated as it is directly related to handling issues of requirements dependency [51]. Moreover, reusable requirements should pass the correctness characteristics, where only valid requirements items are included in the requirements set of the target system.

The increased software requirements testability implies a more leisurely development of software tests. It is expected that testable requirements are tightly coupled with other requirements characteristics, such as measurable and

TABLE 7. Strengths and weaknesses of requirements techniques.

Validation category	Validation group	Weakness	Strength
Prototyping	Prototype generation S05, S11, S18, S47	• Not all tools allow producing reusable code, whereas others provide interfaces that are hard to change for usability later. • The early involvement of users (clients) may result in producing new needs.	• It provides rapid feedback from clients on the provided subsolution toward usability, functionality, and user experience.
	Visualization and animation S15, S20, S24,S29, S30, S56, S62	• As visualization or animations are abstract, they might be ambiguous when compared with entirely textual requirements. • They are application-specific, lacking entire code generation, and may lack parallel visualizations (simulations).	• It is adequate to provide an overall picture of critical systems. • It shows the system behavior interactively without the need to design costly models.
Knowledge-oriented	Rule-based models S14, S15, S17, S21,S64	• Not all types of errors are detected. • Literature has no agreement on language patterns.	• Requirements errors are reduced by ensuring well-formed language that is shared between stakeholders.
	Dictionary and ontology-based models S02, S07, S13, S19, S25, S31, S32, S33, S36, S37, S39, S40, S45, S51, S52, S63, S66	• Each software domain needs an updated ontology. • Cross domains need proper processing.	• Infer knowledge that helps to identify errors in cross disciplines.
	Relationship models (relationship between characteristics) S34, S54, S61, S27, S06, S53	• It needs frequent updates and is sometimes subjective. • Methods that depend on efficiency could not provide meaningful output.	• Frameworks could guide the validation process.
Inspection	Review S01, S03, S26, S43	• The process must be followed strictly. • Hard to comprehend a large set of requirements manually.	• Simple and easy to apply. • A less experienced stakeholder can work here.
	Survey, questionnaires S04, S06	• Need to be customized to particular application domains. • Time-consuming to collect and synthesize results.	• Practical when it is hard to reach clients, especially for software developed for the public.
Testing-oriented	Test case generation S03, S09, S20, S28, S46, S59, S60,S62, S65	• Needs experts in this domain. • It needs good collaboration between clients and the developers. • Exhaustive as complete testing is related to time and budget constraints.	• Early feedback may save time and provide errors ahead of time.
Modeling and assessment	Structure diagrams (conceptual models) S08, S21, S23	• Improper modeling may lead to a waste of time and cost. • Providing a supportive model does not necessitate detecting requirements problems. • Rigid design ensures consistent and correct properties of system requirements. • Full automation is lacking.	• Reduce the complexity of the original set of requirements. • Could provide early correctness-by-construction when it gets integrated with predefined boilerplates.
	Simulation models S12, S22, S35	• Extensive simulations might be time-consuming. • Reactions to the provided models and their reliability are subjective.	• They suit large scale systems and safety-critical systems.
	Goal models S44, S48, S50, S57, S58	• Models are built manually or semi-automatically.	• Captures systems relationships reducing development time.
	Modified V&V framework S10, S18, S55	• It lacks full automated models. • The requirements validation process is less detailed at the early stages of requirements analysis.	• It eliminates detailed V&V steps using tools such as SCADE. • It helps to get interdisciplinary requirements when integrated with SysML. • It ensures a proper integration with the verification process.
Formal Models	Formal Model S09, S11, S16, S20, S29, S38, S41, S42, S49, S65	• It might be hard to interpret models by end users.	• Robust and mathematical support proving. • They form a basis for cohesive and unambiguous requirements.

complete requirements [52]. Testability comes into primary practice for safety-critical systems, where systems must be early validated before implementation [53].

Not only the number of characteristics but also the software requirements industry context or the application domain may indicate the quality of requirements or how to validate

TABLE 8. New research directions to requirements quality.

Characteristic	Approach description	Study
Understandability	Understandability is seen as variability in ambiguities to indicate the degree of understandability of the requirements by stakeholders. The requirements become hard to understand because of variability due to vagueness, multiplicity, and optionality. The QuARS tool analyzes the requirements to capture variabilities, which are then drawn using the feature diagram for analysis.	S32
Variability, Reusability, Dependencies	Variability represents the number of requirements items that were changed. A changed requirement is likely to change others; thus, a review is essential for validation. Reusable requirements reduce the need for requirements validation. Conversely, unexpected dependencies between requirements or groups of them may need further revisions. These attributes and others are embodied with other variables in a Bayesian network where probabilities are obtained based on input data to validate the system requirements.	S23
Testability	A goal, question, metric approach is used to find the root cause of lousy testability. A testable requirement is achieved by measuring the complexity of SRS features based on SRS quality factors.	S44

such characteristics. For example, the RE of health applications [S10] may focus more on correctness characteristics, whereas the RE of an online shopping system attempts to reduce ambiguity between application users [S01].

C. TOOLS AND DATASETS

Our study reported 30 studies (44.45%) that proposed tools, datasets, or both. Most tools depend on machine learning models; however, they inherit problems using machine learning techniques. They suffer from the ''changing thing that changes everything'' [54] problem and interpretability of results unless stakeholders are provided with visualization. It is evidenced (e.g., [S05], [S06], [S47]) that automated validation tools drive datasets for stakeholders' perspective critical decisions. Tools should consider the willingness of humans to use them on the basis of their performance, usability, or user-friendly aspects. Generally, tools are aged or obsolete because of their complexity and low accuracy [55].

D. SOFTWARE APPLICATIONS AND SOFTWARE PROCESS MODELS

Our study revealed that most validation techniques target demo applications as early proof concepts; however, few studies have targeted large enterprise applications, such as critical-safety applications. Additionally, several applications have not yet been discussed in the literature, including mobile applications, manufacturing applications, and research applications.

When it comes to software process models, the software process models are often not specified in literature or are assumed to be traditional nonagile (waterfall) processes. Our analysis found that the provided solution deals with requirements regardless of how they are elicited; it assumes that all requirements are available to the proposed system. Consequently, it is not easy to achieve consistency and completeness requirements quality given the iterative cycles of the agile process models. Thus, agile process models should have quality validation models that automate the agile process of requirements. However, each agile method embraces different practices; hence, RE should be suitable variability [56].

VI. RESEARCH CHALLENGES

The study of software requirements validation is considered challenging because of the following reasons:

A. VALIDATION METHODS' PERSPECTIVES

Although several validation techniques provide many alternatives to practitioners and the research communities, current techniques are often studied from one perspective. For example, prototype-oriented techniques [S05], [S11], [S15], [S18], [S20], [S24], [S29], [S30], [S47], [S56], [S62] focus on the functional requirements presented as a UX or UI to validate the expectation of customers, leaving out nonfunctional requirements for other techniques such as inspection techniques. Consequently, customers' expectations might be distracted or underestimated. Conversely, most knowledge-oriented techniques are directly linked with specified requirements (often in natural language (e.g., [S33], [S36], [S37], [S39], [S25], [S40], [S51]) or use cases [S15]). Consequently, knowledge-oriented models attempt to detect errors on the basis of specification, which might restrict the nature of software under dynamic changing environments. Furthermore, a validation technique changes based on several factors, including the stakeholders' capabilities, team capacity, conflicting viewpoints, and organization assets. Therefore, validation techniques' selection should be further studied considering such factors.

B. SUBJECTIVENESS OF REQUIREMENTS QUALITY CHARACTERISTICS

Current validation techniques are tightly coupled with quality characteristics to validate requirements (shown in Table 6). However, in many cases [e.g., S23], the validation is subjective, which means that thresholds of errors of quality characteristics could be too restrictive or emancipative. Moreover, studied quality characteristics are mainly discussed from nonagile software process models, whereas agile-based quality characteristics are rarely discussed [S12], [S14], [S16], [S44], [S46]. By contrast, intuitively agile-based software systems have a series of incremental requirements validation that should have unique characteristics that imply traceability of requirements toward trends to better or worse systems

in the future. Consequently, the challenge is to balance sensitivity and specificity; not all requirements are troublesome, some of them are intended to be ''vague'' for further expansions.

C. SOFTWARE VARIANTS

Software developers' common practice is to support a wide range of users who could run software over different variants (e.g., web and mobile). However, few studies have focused on this issue [S05], [S47] to the best of our knowledge. Furthermore, studied requirements validation techniques do not impose constraints regarding software variants. Thus, proper validation methods should harmonize the target application requirements with fewer customer efforts and maximum integration to overall project success. An increasing number of variants increases errors in requirements synchronization [57] and becomes unprofitable and less reused [58]. Thus, validation techniques should automate the validation of similar requirements across variants.

D. VALIDATION PROCESS INTEGRATION

As a result of elicitation and negotiation, the requirements get further elaboration, making the requirements validation process a continuous and recursive process [5]. Consequently, automated validation techniques could handle this issue to reduce customer efforts and increase productivity (e.g., [S05], [S06], [S15], [S47]). Moreover, the validation works in silos with verification [S10], [S18], [S55], coding, and testing (e.g., [S04], [S20], [S46]). It is empirical that identifying conceptual links between requirements, involved stakeholders, software development process, and associated risks may result in software redesign because of traceability issues [59]. Thus, validation should be studied from the customer toward the implementation and backward traced to requirements providing complete customer inceptions.

E. LACK OF TOOLS AND DATASETS

One major issue is the lack of open-source tools and benchmark datasets, as shown in Appendix B and consistent with Zogaan *et al.* [33]. The research community should increase efforts to develop more tools and share more datasets. Most tools are designed to be used by software engineers, whereas customers should have their tools to resolve conflicts and specify requirements in more abstract forms. Mitigating the risk of datasets is crucial for dataset development transparency and accountability, supporting data-driven decisions [60]. Thus, datasets should be credible, rigorous, and large enough for deep learning models. Tools should be robust and flexible to support various application domains. Moreover, tools should target detected requirements errors, software process models, and functional project size. Researchers should also consider handling stakeholders' privacy [61], service level agreements that support measurable customer-oriented services, and knowledge representation and sharing (e.g., [S02], [S07], [S13], [S19], [S25], [S31]–[S33], [S36]).

TABLE 9. List of primary studies.

Study	Reference
S01	N. Ali and R. Lai, "A method of software requirements specification and validation for global software development," *Requirements Engineering*, vol. 22, no. 2, pp. 191–214, 2017, doi: 10.1007/s00766-015-0240-4.
S02	C. M. Abraham, M. S. Elayidom, and T. Santhanakrishnan, "Towards risk based effort estimation: A framework to identify, analyze, and classify risk for early identification at requirement engineering phase," *International Journal of Information System Modeling and Design*, vol. 9, no. 4, pp. 67–84, Oct. 2018, doi: 10.4018/IJISMD.2018100105.
S03	W. Miao et al., "Automated requirements validation for ATP software via specification review and testing," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 10009 LNCS, pp. 26–40, 2016, doi: 10.1007/978-3-319-47846-3_3.
S04	M. Autili *et al.*, "A Tool-Supported Methodology for Validation and Refinement of Early-Stage Domain Models," *IEEE Transactions on Software Engineering*, vol. 42, no. 1, pp. 2–25, 2016, doi: 10.1109/TSE.2015.2449319.
S05	Y. Yang, X. Li, W. Ke, and Z. Liu, "Automated Prototype Generation From Formal Requirements Model," *IEEE Transactions on Reliability*, vol. 69, no. 2, pp. 632–656, Jun. 2020, doi: 10.1109/TR.2019.2934348.
S06	A. Rago, J. A. Diaz-Pace, and C. Marcos, "Do concern mining tools really help requirements analysts? An empirical study of the vetting process," *Journal of Systems and Software*, vol. 156, pp. 181–203, 2019, doi: https://doi.org/10.1016/j.jss.2019.06.073.
S07	H. Alrumaih, A. Mirza, and H. Alsalamah, "Domain Ontology for Requirements Classification in Requirements Engineering Context," *IEEE Access*, vol. 8, pp. 89899–89908, 2020, doi: 10.1109/ACCESS.2020.2993838.
S08	E. Stachtiari, A. Mavridou, P. Katsaros, S. Bliudze, and J. Sifakis, "Early validation of system requirements and design through correctness-by-construction," *Journal of Systems and Software*, vol. 145, pp. 52–78, Nov. 2018, doi: 10.1016/j.jss.2018.07.053.
S09	M. Morandini, L. Penserini, A. Perini, and A. Marchetto, "Engineering requirements for adaptive systems," *Requirements Engineering*, vol. 22, no. 1, pp. 77–103, 2017, doi: 10.1007/s00766-015-0236-0.
S10	Barbosa, R., Basagiannis, S., Giantamidis, G., Becker, H., Ferrari, E., Jahic, J., Kanak, A., Esnaola, M. L., Orani, V., Pereira, D., Pomante, L., Schlick, R., Smrcka, A., Yazici, A., Folkesson, P., & Sangchoolie, B. (2020). The VALU3S ECSEL Project: Verification and Validation of Automated Systems Safety and Security. *Proceedings - Euromicro Conference on Digital System Design, DSD 2020*, 352–359. https://doi.org/10.1109/DSD51259.2020.00064
S11	A. Dokhanchi, B. Hoxha, and G. Fainekos, "Formal requirement debugging for testing and verification of cyber-physical systems," *ACM Transactions on Embedded Computing Systems*, vol. 17, no. 2, pp. 1–26, Apr. 2017, doi: 10.1145/3147451.
S12	Z. Ding, M. Jiang, and M. Zhou, "Generating Petri Net-Based Behavioral Models From Textual Use Cases and Application in Railway Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3330–3343, 2016, doi: 10.1109/TITS.2016.2518745.
S13	B. Tenbergen, T. Weyer, and K. Pohl, "Hazard Relation Diagrams: a diagrammatic representation to increase validation objectivity of requirements-based hazard mitigations," *Requirements Engineering*, vol. 23, no. 2, pp. 291–329, 2018, doi: 10.1007/s00766-017-0267-9.
S14	G. Lucassen, F. Dalpiaz, J. M. E. M. M. M. van der Werf, and S. Brinkkemper, "Improving agile requirements: the Quality User Story framework and tool," *Requirements Engineering*,

TABLE 9. *(Continued.)* **List of primary studies.**

Study	Reference
	vol. 21, no. 3, pp. 383–403, 2016, doi: 10.1007/s00766-016-0250-x.
S15	M. Kamalrudin, J. Hosking, and J. Grundy, "MaramaAIC: tool support for consistency management and validation of requirements," *Automated Software Engineering*, vol. 24, no. 1, 2017, doi: 10.1007/s10515-016-0192-z.
S16	M. Wever, L. Van Rooijen, and H. Hamann, "Multioracle coevolutionary learning of requirements specifications from examples in on-the-fly markets," *Evolutionary Computation*, vol. 28, no. 2, pp. 165–193, Jun. 2019, doi: 10.1162/evco_a_00266.
S17	H. Femmer, D. Méndez Fernández, S. Wagner, and S. Eder, "Rapid quality assurance with Requirements Smells," *Journal of Systems and Software*, vol. 123, pp. 190–213, 2017, doi: https://doi.org/10.1016/j.jss.2016.02.047.
S18	D. Iqbal, A. Abbas, M. Ali, M. U. S. Khan, and R. Nawaz, "Requirement Validation for Embedded Systems in Automotive Industry Through Modeling," *IEEE Access*, vol. 8, pp. 8697–8719, 2020, doi: 10.1109/ACCESS.2019.2963774.
S19	S. M. Imtiaz and T. Bhowmik, "Towards Data-Driven Vulnerability Prediction for Requirements," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 744–748, doi: 10.1145/3236024.3264836.
S20	C. E. Tuncali, G. Fainekos, D. Prokhorov, H. Ito, and J. Kapinski, "Requirements-Driven Test Generation for Autonomous Vehicles With Machine Learning Components," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 2, pp. 265–280, Jun. 2020, doi: 10.1109/TIV.2019.2955903.
S21	J. Vilela, J. Castro, L. E. G. Martins, and T. Gorschek, "Safety Practices in Requirements Engineering: The Uni-REPM Safety Module," *IEEE Transactions on Software Engineering*, vol. 46, no. 3, pp. 222–250, Mar. 2020, doi: 10.1109/TSE.2018.2846576.
S22	D. Wu and E. Schnieder, "Scenario-Based Modeling of the On-Board of a Satellite-Based Train Control System With Colored Petri Nets," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3045–3061, 2016, doi: 10.1109/TITS.2016.2535418.
S23	J. del Sagrado and I. M. del Águila, "Stability prediction of the software requirements specification," *Software Quality Journal*, vol. 26, no. 2, pp. 585–605, 2018, doi: 10.1007/s11219-017-9362-x.
S24	M. Daneva, "Striving for balance: A look at gameplay requirements of massively multiplayer online role-playing games," *Journal of Systems and Software*, vol. 134, pp. 54–75, 2017, doi: https://doi.org/10.1016/j.jss.2017.08.009.
S25	J. Misra, "Terminological inconsistency analysis of natural language requirements," *Information and Software Technology*, vol. 74, pp. 183–193, 2016, doi: https://doi.org/10.1016/j.infsof.2015.11.006.
S26	C. Ribeiro and D. Berry, "The prevalence and severity of persistent ambiguity in software requirements specifications: Is a special effort needed to find them?," *Science of Computer Programming*, vol. 195, p. 102472, 2020, doi: https://doi.org/10.1016/j.scico.2020.102472.
S27	A. Salado and R. Nilchiani, "The Tension Matrix and the Concept of Elemental Decomposition: Improving Identification of Conflicting Requirements," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2128–2139, 2017, doi: 10.1109/JSYST.2015.2423658.
S28	M. El-Attar and H. A. Abdul-Ghani, "Using security robustness analysis for early-stage validation of functional security requirements," *Requirements Engineering*, vol. 21, no. 1, pp. 1–27, 2016, doi: 10.1007/s00766-014-0208-9.
S29	T. S. Hoang, C. Snook, A. Salehi, M. Butler, and L. Ladenberger, "Validating and verifying the requirements and design of a haemodialysis machine using the Rodin toolset,"

TABLE 9. *(Continued.)* **List of primary studies.**

Study	Reference
	Science of Computer Programming, vol. 158, pp. 122–147, 2018, doi: 10.1016/j.scico.2017.11.002.
S30	A. Mashkoor and J. P. Jacquot, "Validation of formal specifications through transformation and animation," *Requirements Engineering*, vol. 22, no. 4, pp. 433–451, 2017, doi: 10.1007/s00766-016-0246-6.
S31	N. Itzik, I. Reinhartz-Berger, and Y. Wand, "Variability Analysis of Requirements: Considering Behavioral Differences and Reflecting Stakeholders' Perspectives," *IEEE Transactions on Software Engineering*, vol. 42, no. 7, pp. 687–706, Jul. 2016, doi: 10.1109/TSE.2015.2512599.
S32	A. Fantechi, S. Gnesi, and L. Semini, "Ambiguity Defects as Variation Points in Requirements," in *Proceedings of the Eleventh International Workshop on Variability Modelling of Software-Intensive Systems (VAMOS)*, 2017, pp. 13–19, doi: 10.1145/3023956.3023964.
S33	I. Atoum, "A Scalable Operational Framework for Requirements Validation Using Semantic and Functional Models," *ACM International Conference Proceeding Series*, pp. 1–6, 2019, doi: 10.1145/3305160.3305166.
S34	A. Schlutter and A. Vogelsang, "Knowledge Extraction from Natural Language Requirements into a Semantic Relation Graph," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 373–379, doi: 10.1145/3387940.3392162.
S35	C. Tsigkanos, N. Li, Z. Jin, Z. Hu, and C. Ghezzi, "On Early Statistical Requirements Validation of Cyber-Physical Space Systems," in *Proceedings of the 4th International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS)*, 2018, pp. 13–18, doi: 10.1145/3196478.3196485.
S36	W. Alhoshan, R. Batista-Navarro, and L. Zhao, "Using frame embeddings to identify semantically related software requirements," in *The 2nd Workshop on Natural Language Processing for Requirements Engineering*, 2019, vol. 2376.
S37	W. Alhoshan, L. Zhao, and R. Batista-Navarro, "Using Semantic Frames to Identify Related Textual Requirements: An Initial Validation," 2018, doi: 10.1145/3239235.3267441.
S38	J. Feng *et al.*, "FREPA: An automated and formal approach to requirement modeling and analysis in aircraft control domain," in *ESEC/FSE 2020 - Proceedings of the 28th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 1376–1386, doi: 10.1145/3368089.3417047.
S39	W. Alhoshan, R. Batista-Navarro, and L. Zhao, "Towards a corpus of requirements documents enriched with semantic frame annotations," in *2018 IEEE 26th International Requirements Engineering Conference (RE)*, 2018, pp. 428–431, doi: 10.1109/RE.2018.00055.
S40	X. Zhang and X. Wang, "Tradeoff Analysis for Conflicting Software Non-Functional Requirements," *IEEE Access*, vol. 7, pp. 156463–156475, 2019, doi: 10.1109/ACCESS.2019.2949218.
S41	J. Greenyer *et al.*, "SCENARIOTOOLS – A tool suite for the scenario-based modeling and analysis of reactive systems," *Science of Computer Programming*, vol. 149, no. C, pp. 15–27, Dec. 2017, doi: 10.1016/j.scico.2017.07.004.
S42	C. Wiecher, J. Greenyer, and J. Korte, "Test-Driven Scenario Specification of Automotive Software Components," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 2019, pp. 12–17, doi: 10.1109/MODELS-C.2019.00009.
S43	J. Baker *et al.*, "Requirements Engineering for Retrofittable Subsea Equipment," in *Proceedings - 2016 IEEE 24th International Requirements Engineering Conference, RE 2016*, 2016, pp. 226–235, doi: 10.1109/RE.2016.44.
S44	A. Beer and M. Felderer, "Measuring and improving testability of system requirements in an industrial context by applying the goal question metric approach," in *Proceedings - International Conference on Software Engineering*, 2018, pp. 25–32, doi:

TABLE 9. *(Continued.)* **List of primary studies.**

Study	Reference
	10.1145/3195538.3195542.
S45	J. A. Khan, Y. Xie, L. Liu, and L. Wen, "Analysis of requirements-related arguments in user forums," in *Proceedings of the IEEE International Conference on Requirements Engineering*, 2019, vol. 2019-Septe, pp. 63–74, doi: 10.1109/RE.2019.00018.
S46	D. N. Jorge, P. D. L. L. Machado, E. L. G. G. Alves, and W. L. Andrade, "Integrating requirements specification and model-based testing in agile development," in *Proceedings - 2018 IEEE 26th International Requirements Engineering Conference, RE 2018*, 2018, pp. 336–346, doi: 10.1109/RE.2018.00041.
S47	Y. Yang, X. Li, Z. Liu, and W. Ke, "RM2PT: A Tool for Automated Prototype Generation from Requirements Model," in *Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings*, 2019, pp. 59–62, doi: 10.1109/ICSE-Companion.2019.00038.
S48	E. Alatawi, A. Mendoza, and T. Miller, "Psychologically-Driven Requirements Engineering: A Case Study in Depression Care," in *2018 25th Australasian Software Engineering Conference (ASWEC)*, 2018, pp. 41–50, doi: 10.1109/ASWEC.2018.00014.
S49	C. Wiecher, S. Japs, L. Kaiser, J. Greenyer, R. Dumitrescu, and C. Wolff, "Scenarios in the Loop: Integrated Requirements Analysis and Automotive System Validation," 2020, doi: 10.1145/3417990.3421264.
S50	R. Delima, A. Wibowo, A. Rachmat Chrismanto, and H. Budi Santoso, "A Model of Requirements Engineering on Agriculture Mobile Learning System Using Goal-Oriented Approach," in *2020 Fifth International Conference on Informatics and Computing (ICIC)*, 2020, pp. 1–8, doi: 10.1109/ICIC50835.2020.9288536.
S51	M. H. Osman and M. F. Zaharin, "Ambiguous Software Requirement Specification Detection: An Automated Approach," in *Proceedings of the 5th International Workshop on Requirements Engineering and Testing*, 2018, pp. 33–40, doi: 10.1145/3195538.3195545.
S52	E. Onyeka, A. S. Varde, V. Anu, N. Tandon, and O. Daramola, "Using Commonsense Knowledge and Text Mining for Implicit Requirements Localization," in *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, 2020, vol. 2020-Novem, pp. 935–940, doi: 10.1109/ICTAI50040.2020.00146.
S53	C. Stanik and W. Maalej, "Requirements Intelligence with OpenReq Analytics," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019, vol. 2019-Septe, pp. 482–483, doi: 10.1109/RE.2019.00066.
S54	G. Deshpande, "SReYantra: Automated Software Requirement Inter-Dependencies Elicitation, Analysis and Learning," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2019, pp. 186–187, doi: 10.1109/ICSE-Companion.2019.00076.
S55	C. Lankeit, J. Michael, C. Henke, and A. Trächtler, "Holistic Requirements for Interdisciplinary Development Processes," in *2018 1st International Workshop on Learning from other Disciplines for Requirements Engineering (D4RE)*, 2018, pp. 4–7, doi: 10.1109/D4RE.2018.00007.
S56	M. Urbieta, N. Torres, J. M. Rivero, G. Rossi, and F. J. Dominguez-Mayo, "Improving Mockup-Based Requirement Specification with End-User Annotations," in *Agile Processes in Software Engineering and Extreme Programming*, 2018, vol. 314, pp. 19–34, doi: 10.1007/978-3-319-91602-6_2.
S57	J. Dąbrowski, E. Letier, A. Perini, and A. Susi, "Mining User Opinions to Support Requirement Engineering: An Empirical Study," in *Advanced Information Systems Engineering (CAiSE)*, 2020, vol. 12127 LNCS, pp. 401–416, doi: 10.1007/978-3-030-49435-3_25.

TABLE 9. *(Continued.)* **List of primary studies.**

Study	Reference
S58	T. Hassan, M. Z. Hussain, M. Z. Hasan, Z. Ullah, and N. ul Qamar, "Quantitative Based Mechanism for Resolving Goals Conflicts in Goal Oriented Requirement Engineering," in *Intelligent Technologies and Applications*, 2019, vol. 932, pp. 822–831, doi: 10.1007/978-981-13-6052-7_71.
S59	A. Gupta and P. Bera, "A Software Tool to Convert Requirements to Test Cases," in *Proceedings of the 6th International Workshop on Requirements Engineering and Testing (RET)*, 2019, pp. 9–12, doi: 10.1109/RET.2019.00009.
S60	S. Vuotto, M. Narizzano, L. Pulina, and A. Tacchella, "Automata Based Test Generation with SpecPro," in *Proceedings of the 6th International Workshop on Requirements Engineering and Testing(RET)*, 2019, pp. 13–16, doi: 10.1109/RET.2019.00010.
S61	K. Mori, N. Okubo, Y. Ueda, M. Katahira, and T. Amagasa, "Supporting Viewpoints to Review the Lack of Requirements in Space Systems with Machine Learning," in *Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2020, pp. 38–44, doi: 10.1145/3387939.3391610.
S62	N. A. Moketar, M. Kamalrudin, S. Sidek, M. Robinson, and J. Grundy, "TestMEReq: Generating Abstract Tests for Requirements Validation," in *Proceedings of the 3rd International Workshop on Software Engineering Research and Industrial Practice (TEFSE)*, 2016, pp. 39–45, doi: 10.1145/2897022.2897031.
S63	Osama, M., Zaki-Ismail, A., Abdelrazek, M., Grundy, J., & Ibrahim, A. (2020). Score-Based Automatic Detection and Resolution of Syntactic Ambiguity in Natural Language Requirements. *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 651–661. https://doi.org/10.1109/ICSME46990.2020.00067
S64	Moreno, V., Génova, G., Parra, E., & Fraga, A. (2020). Application of machine learning techniques to the flexible assessment and improvement of requirements quality. *Software Quality Journal*, *28*(4), 1645–1674. https://doi.org/10.1007/s11219-020-09511-4
S65	Moitra, A., Siu, K., Crapo, A. W., Durling, M., Li, M., Manolios, P., Meiners, M., & McMillan, C. (2019). Automating requirements analysis and test-case generation. *Requirements Engineering*, *24*(3), 341–364. https://doi.org/10.1007/s00766-019-00316-x
S66	Ferrari, A., & Esuli, A. (2019). An NLP approach for cross-domain ambiguity detection in requirements engineering. *Automated Software Engineering*, *26*(3), 559–598. https://doi.org/10.1007/s10515-019-00261-7

VII. CONCLUSION AND DIRECTION TO FUTURE RESEARCH

This section concludes the work and lists a research direction about requirements validation.

A. CONCLUSION

The leading role of requirements validation is to certify that requirements are acceptable descriptions of the target system. Checking requirements (a single or set) for errors entails an acceptable threshold for software validation requirements. A systematic literature review following Kitchenham's method [23] identified 66 primary studies; 36.7% of the validation methods adopted knowledge-oriented techniques (e.g., machine learning methods), and 17.7% adopted

TABLE 10. Datasets and tools.

Study	Reference		
S04	Tool Name	Model Testing by Human Interrogations & Answers (MOTHIA)	
	Description	MOTHIA questionnaire helps in detecting modeling faults based on mutations of the original domain model.	
	Tool URL	http://labsedc.isti.cnr.it/tools/mothia	
	Quality	Necessity, Completeness, Consistency, Ambiguity	
	Dataset	Name	Large Scale Choreographies for the Future Internet (CHOReOS)
		URL	CHOREOS: Large scale choreographies for the future internet [64]
		Performance	Efficacy and adequacy on class diagrams and activity and use case diagrams. Reported (95.90%,91.92%) and (81.78%,71.35%) respectively.
S05 S47	Tool Name	RM2PT (Requirements to Prototype)	
	Description	Automated Prototype Generation from a Requirements model	
	Tool URL	https://github.com/RM2PT	
	Quality	Completeness, Correctness	
	Dataset	Name	A sample dataset (51 use cases) of actors, operations, classes, and invariants
		URL	Not informed
		Performance	93.65% of requirement specifications are executable. In addition, the auto-generation saves approximately nine hours for a skilled programmer.
S06	Tool Name	REAssistant, EAMiner	
	Description	A tool designed for requirements reviews	
	Tool URL	http://rm2pt.com/import_project.html	
	Quality	Completeness	
	Dataset	Name	20 Requirements documents of average 19 pages
		URL	http://www.alejandrorago.com.ar/files/assets/dataset-ConcernVetting.zip
		Performance	Accuracy variation between the initial and final solution using precision = 0.3924, recall 0.1135.
S09	Tool Name	TAOM,Tropos4AS to Jadex	
	Description	Tool for Agent-Oriented visual Modeling for Eclipse and its plugin Tropos4AS to Jadex (t2x)	
	Tool URL	http://selab.fbk.eu/taom	
	Performance	Tropos4AS models covered the evaluation scenarios by more than 87%. Several version of The iCleaner(robot) use case was tested and compared with efficiency, failure, and robustness	
S14	Tool Name	Automatic Quality User Story Artisan (AQUSA)	
	Description	Story Generation	
	Tool URL	https://github.com/gglucass/AQUSA	
	Quality	Feasibility, Comprehensiveness, Necessity, Completeness, Consistency, Conformance, ambiguity,Uniqueness	
S15	Tool Name	Automated Inconsistency Checker (MaramaAIC)	
	Description	Rapid prototyping approach together with a pattern's library	
	Tool URL	https://dl.acm.org/doi/pdf/10.1109/ASE.2009.38	
	Quality	Completeness, Consistency, Correctness, Traceability	
S17	Tool Name	Smell Analysis (Smella)	
	Description	Smella is a web-based tool that enables viewing, reviewing, and deny listing findings and a hotspot analysis at an artifact level.	
	Quality	Completeness, Conformance, Ambiguity	
S18	Tool Name	Safety-Critical Application Development Environment (SCADE)	
	Description	SCADE Suite is used to design critical software, such as flight control and engine control systems, landing gear systems, automatic pilots	
	Tool URL	https://www.ansys.com/products/embedded-software/ansys-scade-suite	
	Quality	Consistency, Ambiguity	
	Performance	In the formal scenario, the model reports efforts of 17.7 PM while it was around 9.6 for the modeling scenario.	
S20	Tool Name	Simulation-based Adversarial Testing of Autonomous Vehicles (Sim-ATAV)	
	Description	Sim-ATAV is a simulation-based adversarial testing framework for Cyber-Physical systems such as search-based test-case generation and automated falsification	
	Tool URL	https://sites.google.com/a/asu.edu/s-taliro/sim-atav	
	Quality	Ambiguity	
	Performance	Deep Neural Networks architecture with 99.46% accuracy.	
S21	Tool Name	Unified Requirements Engineering Process Maturity Model (Uni-REPM)	
	Description	Uni-REPM is a light-weight model presenting the maturity of the RE process through sets of necessary activities	
	Tool URL	http://www.unirepm.com/	
	Quality	Conformance, Completeness, Consistency, Ambiguity, Correctness	
	Performance	Validated by experts with 81.82% acceptance	
S22	Tool Name	Colored Petri Nets Tools	
	Description	Tools for editing, simulating and analyzing Colored Petri nets	
	Tool URL	http://cpntools.org	
	Quality	Consistency, Correctness	
	Performance	All the test cases have passed	
S23	Tool Name	Requisites, InSCo-Requisite	
	Description	A Web-CASE tool prototype for hybrid software development	
	Tool URL	https://link.springer.com/chapter/10.1007/11556985_28	
	Quality	Completeness, Consistency, Specificity, Variability, Reusability	
	Dataset	Name	RALIC (replacement access, library and ID cards)
		URL	https://www.ucl.ac.uk/library/about-us/annual-report/200607/ucl-library-services-annual-report-200607
		Performance	Completeness and degree of revision scores (subjective)
S25	Tool Name	Apache OpenNLP	
	Description	The Apache OpenNLP library is a machine learning-based toolkit for the processing of natural language text.	
	Tool URL	https://opennlp.apache.org/	
	Dataset	Name	41 requirements of65 requirements' sentences
		URL	http://www2.epa.gov/sites/production/files/2015-04/documents/bf_property_appendixe.pdf
		Performance	The minimum context for semantic alias identification was 60% F-measure
S29	Tool Name	Event-B,iUML-B,ProR,BMotion Studio,Dymola	
	Description	Event-B is a formal method for modeling and analysis at the system level. The iUML-B is a collection of diagrammatic editors for Event-B. The ProR adapter identifies traceable artifacts based on the requirement. BMotion Studio is a graphical tool based on ProB (specification formalisms validation tool) for domain-specific visualization. Dynamic Modeling Laboratory(DML) is a comprehensive tool for Modeling and Assessment complex integrated systems.	

TABLE 10. *(Continued.)* **Datasets and tools.**

	Tool URL	http://www.event-b.org/ https://wiki.event-b.org/index.php/IUML-B https://www.eclipse.org/rmf/pror/ https://wiki.event-b.org/index.php/BMotion_Studio https://www.3ds.com/products-services/catia/products/dymola/model-design-tools/	
	Performance	89% obligations approved with automatic provers. 98% proved with MT solvers. The Arterial Pressure was correctly controlled to a steady value of 72%.	
S31	Tool Name	Semantic and Ontological Variability Analysis (SOVA)	
	Description	A Tool for Semantic and Ontological Variability Analysis	
	Tool URL	https://sova.cc/business-intelligence.html	
	Dataset	Name	E-shop Functional Requirements
		URL	In the paper
		Performance	With SOVA featured diagrams, it takes participants on average 126.73 compared to 188.24 with textual requirements with a lower level of difficulty and a higher confidence level.
S41	Tool Name	ScenarioTools	
	Description	The modeling in ScenarioTools is based on the Scenario Modeling Language (SML), an extended variant of Live Sequence Charts (LSCs).	
	Tool URL	Eclipse-based tool suite	
S42, S49	Tool Name	The Scenario Modeling Language for Kotlin(SML)	
	Description	Formulate requirements intuitively and provide engineers with automated checks for requirements validation.	
	Tool URL	-	
	Performance	Automated tests increase efficiency. The model is integrated with Behavior-Driven Development (BDD), test-driven scenario specification, and Cucumber	
S46	Tool Name	CentraL Artifact for Requirement Engineering and model-based Testing (CLARET)	
	Description	Generate use cases from natural language processing and structured use case diagrams	
	Tool URL	https://splab-ufcg.github.io/claret/	
	Performance	measured created use cases and test cases'time, requirements coverage, completeness, and consistency.	
S51	Tool Name	Commonsense Knowledge, Ontology and Text mining for Implicit Requirements (COTIR)	
	Description	It depends on finding textual similarities in SRS, identify a basis for analogy and discover knowledge	
	Tool URL	-	
	Performance	F-measure 0f 70.3 over 3 datasets	
S53	Tool Name	OpenReq Analytics	
	Description	Analyze feedback from users on Twitter for a specific app to find sentiment, reuse, or find dependencies between requirements	
	Tool URL	https://github.com/OpenReqEU	
S54	Tool Name	SReYantra: Automated Software Requirement Inter-dependencies	
	Description	Finding interdependencies in requirements	
	Tool URL	-	
S59	Tool Name	TestAlgo	
	Description	Generate test cases and process models. Requirements are extracted to action triads (test cases format), fed into the optimization engine to generate test cases.	
	Tool URL	-	
S62	Tool Name	TestMEReq	
	Description	It is an automated tool requirements validation that includes correctness, completeness, and consistency. In addition, the tool generates Essential Use Cases (EUC), mock-up prototypes, and use cases.	
	Tool URL	-	
	Performance	The tool was tested over 15 applications; the correctness was above 50%	

TABLE 10. *(Continued.)* **Datasets and tools.**

S64	Tool Name	Requirements Quality Analyzer (RQA)	
	Description	Extract quality metrics (correctness in the paper)	
	Tool URL	https://www.reusecompany.com/rqa-quality-studio	
	Dataset	1035 textual software requirements from the domain of the aerospace industry	
	Performance	The tool reported an effectiveness of around 85% from different machine learning methods.	
S65	Tool Name	Requirements-based Tests (ASSERT)	
	Description	ASSERT also automatically generates a complete set of requirements-based test cases	
	Tool URL	Eclipse plugin	
	Performance	Formal model	
S39	Other Datasets	Description	URL
		Eighteen structured requirements documents enriched with FN annotation (FN-RE).	https://zenodo.org/record/1291660#.YNRmWyBR1PY
S45		72 annotated forum comments	https://drive.google.com/open?id=1oYHct7WWgycLyJgHhiaVTpIiLFHi8Qo
S57		1,000 reviews annotated with 1,521 opinions	https://github.com/jsdabrowski/CAiSE-20/

modeling and assessment techniques (e.g., simulations). Contrary to the state of practice, inspection methods (7.6%) are less studied in the literature, whereas testing-oriented (11.4%) methods are generally less informative to end users. Formal models are often used in critical systems, but their usage is marginally less than the other methods; thus, it is challenging to establish an explicit shared understanding. Moreover, the studied requirements quality characteristics showed the usage of application-specific characteristics; therefore, they still necessitate proper knowledge-oriented validation techniques to handle their underlying semantics. Approximately three-quarters of the studies (74.24%) did not report the applicability of the validation technique over specific software process models (agile vs. nonagile); agile validation methods are scarce. Therefore, the primary studies showed that current validation techniques are considered semi-automated and software domain dependent. Consequently, selecting a validation technique depends on the nature of the problem, stakeholder experience, and availability of tools. Most validation techniques are related to expressing the requirements for detection errors and incorporating clients' feedback quickly.

B. FUTURE RESEARCH DIRECTIONS

This section presents research directions to resolve issues discussed in the previous section.

1) COMBINING VALIDATION METHODS

We anticipate fruitful results when integrating formal and prototype models that support prototype generation to provide a sound output [S11], [S29]. Formal models support automated proving, whereas prototypes provide instant

customer feedback. Additionally, testing-oriented techniques may enhance inspection techniques [S03]. For example, the validation model of the automatic train protection system depends on a prototype generated from user scenarios [S03], which is then followed by a report to visualize the test results. Reviews were then used to validate these requirements. However, the requirements specification templates in [S03] do not solve behavior requirements issues in self-adaptive or embedded systems [62]. Hence, the challenge is to specialize in requirements validation techniques as per the target application of interest.

2) QUALITY CHARACTERISTICS UNIFICATION

With a nonuniform list of characteristics, the research challenge is to find a clustering technique to analyze quality characteristics, including their synonyms or antonyms, or related ontological concepts, in such a way that could be further applied to a particular software application domain. For example, language patterns transfer requirements into semi-formal specification domain ontologies, enabling a semiformal validation of the list of requirements qualities. A starting point could be adopting model-based RE to specify large system extents supporting interdependency objectives and allowing requirements reusability [62]. Moreover, because stakeholders have different viewpoints (concerns), mining concerns are essential for extracting semantic information to reveal latent concerns [S06]. Researchers can then build a unified framework that handles multilevel quality attributes [39].

3) VALIDATION FOR SOFTWARE VARIANTS

Given a software requirement that can run on different platforms, it is crucial to have a validation method that reveals similar results regardless of the software variant. A starting point to this problem is the one that represents the Cross-Fabrik meta-model, which comprises feature models, class models, and their relationships [63]; however, it was not included in our research because its focus is on development rather than on RE. Next, the specification of the variability applications running on more than one platform is developed using a feature model and product definition. Finally, a simulation process is used to validate the software variants across multiple devices.

4) REQUIREMENTS VALIDATION PROCESS INTEGRATION

Validation is not evidently separated from other software requirements processes; thus, a comprehensive requirements validation technique should work in silos with other related processes (e.g., elicitation), providing a comprehensive and holistic RE model. The research field of validation is fascinating the verification methods; hence, the research could be on the relationship between these factors to enhance the overall software quality. It should be customized according to the required software development process models. Moreover, it should dynamically change on the basis of the stakeholders' capacities and existing project capacity. A starting point could be the modified V&V [S18], [S55] and software requirements theory [3].

APPENDIX A
LIST OF PRIMARY STUDIES

See Table 9.

APPENDIX B
DATASETS AND TOOLS

See Table 10.

REFERENCES

[1] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*, 1st ed. Hoboken, NJ, USA: Wiley, 2009.
[2] D. M. Fernández *et al.*, "Naming the pain in requirements engineering," *Empirical Softw. Eng.*, vol. 22, no. 5, pp. 2298–2338, Oct. 2017, doi: 10.1007/s10664-016-9451-7.
[3] S. Wagner *et al.*, "Status quo in requirements engineering: A theory and a global family of surveys," *ACM Trans. Softw. Eng. Methodol.*, vol. 28, no. 2, pp. 1–48, Apr. 2019, doi: 10.1145/3306607.
[4] F. Dalpiaz, P. Gieske, and A. Sturm, "On deriving conceptual models from user requirements: An empirical study," *Inf. Softw. Technol.*, vol. 131, Mar. 2021, Art. no. 106484, doi: 10.1016/j.infsof.2020.106484.
[5] *ISO/IEC/IEEE International Standard—Systems and Software Engineering—Life Cycle Processes—Requirements Engineering*, Standard ISO/IEC/IEEE 29148:2018(E), 2018, pp. 1–104, doi: 10.1109/IEEESTD.2018.8559686.
[6] E. M. Schön, J. Thomaschewski, and M. J. Escalona, "Agile requirements engineering: A systematic literature review," *Comput. Stand. Interface*, vol. 49, pp. 79–91, Jan. 2017, doi: 10.1016/j.csi.2016.08.011.
[7] B. H. C. Cheng and J. M. Atlee, "Research directions in requirements engineering," in *Proc. Future Softw. Eng. (FoSE)*, May 2007, pp. 285–303, doi: 10.1109/FOSE.2007.17.
[8] H. Dar, M. I. Lali, H. Ashraf, M. Ramzan, T. Amjad, and B. Shahzad, "A systematic study on software requirements elicitation techniques and its challenges in mobile application development," *IEEE Access*, vol. 6, pp. 63859–63867, 2018, doi: 10.1109/ACCESS.2018.2874981.
[9] B. Nuseibeh and S. Easterbrook, "Requirements engineering: A roadmap," in *Proc. Conf. Future Softw. Eng. (ICSE)*, vol. 1, 2000, pp. 35–46, doi: 10.1145/336512.336523.
[10] A. Bennaceur, T. T. Tun, Y. Yu, and B. Nuseibeh, "Requirements engineering," in *Handbook of Software Engineering*, 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01758502 and https://hal.archives-ouvertes.fr/hal-01758502/file/reChapter_revision_21082017.pdf
[11] K.-J. Stol and B. Fitzgerald, "The ABC of software engineering research," *ACM Trans. Softw. Eng. Methodol.*, vol. 27, no. 3, pp. 1–51, Oct. 2018, doi: 10.1145/3241743.
[12] M. Shameem, B. Chandra, C. Kumar, and A. A. Khan, "Understanding the relationships between requirements uncertainty and nature of conflicts: A study of software development team effectiveness," *Arabian J. Sci. Eng.*, vol. 43, pp. 8223–8238, Jun. 2018, doi: 10.1007/s13369-018-3375-z.
[13] L. Canchari and A. Dávila, "Requirements validation in the information systems software development: An empirical evaluation of its benefits for a public institution in lima," in *Proc. Int. Conf. Softw. Process Improvement*, in Advances in Intelligent Systems and Computing, vol. 1071, 2020, pp. 23–35, doi: 10.1007/978-3-030-33547-2_3.
[14] K. Pohl, *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam-Foundation Level-IREB Compliant*. San Rafael, CA, USA: Rocky Nook, 2016.
[15] Z. S. H. Abad, V. Gervasi, D. Zowghi, and B. H. Far, "Supporting analysts by dynamic extraction and classification of requirements-related knowledge," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng. (ICSE)*, May 2019, pp. 442–453, doi: 10.1109/ICSE.2019.00057.
[16] S. Chakraborty, C. Rosenkranz, and J. Dehlinger, "Getting to the shalls: Facilitating sensemaking in requirements engineering," *ACM Trans. Manage. Inf. Syst.*, vol. 5, no. 3, pp. 1–30, Jan. 2015, doi: 10.1145/2629351.
[17] J. Pivatelli and J. C. S. do Prado Leite, "The clash between requirements volatility and software contracts," in *Proc. 31st Brazilian Symp. Softw. Eng. (SBES)*, 2017, pp. 144–153, doi: 10.1145/3131151.3131159.

[18] J. L. King and C. P. Simon, "Complications with complexity in requirements," *ACM Trans. Manage. Inf. Syst.*, vol. 5, no. 3, pp. 1–12, Jan. 2015, doi: 10.1145/2629375.

[19] A. T. S. Calazans, R. Á. Paldês, E. D. Canedo, E. T. S. Masson, F. de Albuquerque Guimarães, K. M. F. Rezende, F. de Souza Gonçalves, and A. M. Mariano, "Quality requirements and the requirements quality: The indications from requirements smells in a financial institution systems," in *Proc. 33rd Brazilian Symp. Softw. Eng.*, Sep. 2019, pp. 472–480, doi: 10.1145/3350768.3350782.

[20] M. Kalinowski, E. Mendes, D. N. Card, and G. H. Travassos, "Applying DPPI: A defect causal analysis approach using Bayesian networks," in *Proc. Int. Conf. Product Focused Softw. Process Improvement*, 2010, pp. 92–106.

[21] R. R. Young, *The Requirements Engineering Handbook*. Norwood, MA, USA: Artech House, 2004.

[22] D. Firesmith, "Specifying good requirements," *J. Object Technol.*, vol. 2, no. 4, pp. 77–87, 2003.

[23] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering. Version 2.3," School Comput. Sci. Math., Keele Univ., Newcastle, U.K., Tech. Rep. EBSE-2007-01, 2007. [Online]. Available: https://goo.gl/L1VHcw

[24] I. Sommerville and P. Sawyer, *RE: A Good Practice Guide*. Hoboken, NJ, USA: Wiley, 1997.

[25] M. J. Machina and M. Siniscalchi, "Ambiguity and ambiguity aversion," in *Handbook of the Economics of Risk and Uncertainty*, vol. 1, M. Machina and K. Viscusi, Eds. Amsterdam, The Netherlands: North-Holland, 2014, ch. 13, pp. 729–807. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444536853000131, doi: 10.1016/B978-0-444-53685-3.00013-1.

[26] S. Sepúlveda, A. Cravero, and C. Cachero, "Requirements modeling languages for software product lines: A systematic literature review," *Inf. Softw. Technol.*, vol. 69, pp. 16–36, Jan. 2016, doi: 10.1016/j.infsof.2015.08.007.

[27] N. A. Moketar, M. Kamalrudin, M. M. Yusof, and S. Sidek, "A review on requirements validation for software development," *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 11, pp. 3182–3193, 2018.

[28] N. Tripathi, E. Klotins, R. Prikladnicki, M. Oivo, L. B. Pompermaier, A. S. Kudakacheril, M. Unterkalmsteiner, K. Liukkunen, and T. Gorschek, "An anatomy of requirements engineering in software startups using multivocal literature and case survey," *J. Syst. Softw.*, vol. 146, pp. 130–151, Dec. 2018, doi: 10.1016/j.jss.2018.08.059.

[29] V. Gupta, J. M. Fernandez-Crehuet, T. Hanne, and R. Telesko, "Requirements engineering in software startups: A systematic mapping study," *Appl. Sci.*, vol. 10, no. 17, p. 6125, Sep. 2020, doi: 10.3390/app10176125.

[30] L. E. G. Martins and T. Gorschek, "Requirements engineering for safety-critical systems: A systematic literature review," *Inf. Softw. Technol.*, vol. 75, pp. 71–89, Jul. 2016, doi: 10.1016/j.infsof.2016.04.002.

[31] S. Z. Hlaing and K. Ochimizu, "An integrated cost-effective security requirement engineering process in SDLC using FRAM," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2018, pp. 852–857, doi: 10.1109/CSCI46756.2018.00170.

[32] S. Besrour, L. B. A. Rahim, and P. D. D. Dominic, "A quantitative study to identify critical requirement engineering challenges in the context of small and medium software enterprise," in *Proc. 3rd Int. Conf. Comput. Inf. Sci. (ICCOINS)*, Aug. 2016, pp. 606–610, doi: 10.1109/ICCOINS.2016.7783284.

[33] W. Zogaan, P. Sharma, M. Mirahkorli, and V. Arnaoudova, "Datasets from fifteen years of automated requirements traceability research: Current state, characteristics, and quality," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 110–121.

[34] 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.

[35] D. S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering," in *Proc. Int. Symp. Empirical Softw. Eng. Meas.*, Sep. 2011, pp. 275–284.

[36] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proc. 18th Int. Conf. Eval. Assessment Softw. Eng. (EASE)*, 2014, pp. 1–10.

[37] E. Hull, K. Jackson, and J. Dick, "Defining requirement engineering," in *Requirements Engineering*. London, U.K.: Springer, 2010, pp. 6–8.

[38] D. Zowghi and V. Gervasi, "On the interplay between consistency, completeness, and correctness in requirements evolution," *Inf. Softw. Technol.*, vol. 45, no. 14, pp. 993–1009, 2003.

[39] V. Gervasi, A. Ferrari, D. Zowghi, and P. Spoletini, "Ambiguity in requirements engineering: Towards a unifying framework," in *From Software Engineering to Formal Methods and Tools, and Back: Essays Dedicated to Stefania Gnesi on the Occasion of Her 65th Birthday*, M. H. ter Beek, A. Fantechi, and L. Semini, Eds. Cham, Switzerland: Springer, 2019, pp. 191–210, doi: 10.1007/978-3-030-30985-5_12.

[40] L. E. G. Martins and T. Gorschek, "Requirements engineering for safety-critical systems: An interview study with industry practitioners," *IEEE Trans. Softw. Eng.*, vol. 46, no. 4, pp. 346–361, Apr. 2020, doi: 10.1109/TSE.2018.2854716.

[41] R. Rabiser, M. Vierhauser, and P. Grünbacher, "Assessing the usefulness of a requirements monitoring tool: A study involving industrial software engineers," in *Proc. 38th Int. Conf. Softw. Eng. Companion*, May 2016, pp. 122–131, doi: 10.1145/2889160.2889234.

[42] J. Medeiros, A. Vasconcelos, C. Silva, and M. Goulão, "Quality of software requirements specification in agile projects: A cross-case analysis of six companies," *J. Syst. Softw.*, vol. 142, pp. 171–194, Aug. 2018.

[43] R. Cursino, J. Farias, M. Lancastre, and W. Santos, "Agile requirements validation in Brazilian software development companies: A survey," in *Proc. Brazilian Workshop Agile Methods*, in Communications in Computer and Information Science, vol. 981, 2019, pp. 3–18, doi: 10.1007/978-3-030-14310-7_1.

[44] O. Laitenberger, T. Beil, and T. Schwinn, "An industrial case study to examine a non-traditional inspection implementation for requirements specifications," *Empirical Softw. Eng.*, vol. 7, no. 4, pp. 345–374, 2002.

[45] A. Jarzębowicz and W. Ślesiński, "Assessing effectiveness of recommendations to requirements-related problems through interviews with experts," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, Sep. 2018, pp. 959–968, doi: 10.15439/2018F85.

[46] K. Mokos and P. Katsaros, "A survey on the formalisation of system requirements and their validation," *Array*, vol. 7, Sep. 2020, Art. no. 100030, doi: 10.1016/j.array.2020.100030.

[47] F. Chalvatza, S. Karkalas, and M. Mavrikis, "Communicating learning analytics: Stakeholder participation and early stage requirement analysis," in *Proc. 11th Int. Conf. Comput. Supported Educ. (CSEDU)*, vol. 2, 2019, pp. 339–346, doi: 10.5220/0007716503390346.

[48] A. Albaghajati and J. Hassine, "A use case driven approach to game modeling," *Requirements Eng.*, vol. 26, no. 3, pp. 1–34, 2021.

[49] R. Rabiser and A. Zoitl, "Towards mastering variability in software-intensive cyber-physical production systems," *Proc. Comput. Sci.*, vol. 180, pp. 50–59, Jan. 2021, doi: 10.1016/j.procs.2021.01.128.

[50] A. Fantechi, S. Gnesi, S. Livi, and L. Semini, "A spaCy-based tool for extracting variability from NL requirements," in *Proc. 25th ACM Int. Syst. Softw. Product Line Conf.*, Sep. 2021, pp. 32–35, doi: 10.1145/3461002.3473074.

[51] A. Gupta and C. Gupta, "A collaborative effort-benefit-value analysis model to support requirements reuse for software requirements prioritization," *Int. J. Softw. Innov.*, vol. 9, no. 1, pp. 37–51, Jan. 2021.

[52] V. Garousi, M. Felderer, and F. N. Kılıçaslan, "A survey on software testability," *Inf. Softw. Technol.*, vol. 108, pp. 35–64, Apr. 2019, doi: 10.1016/j.infsof.2018.12.003.

[53] A. Beer and M. Felderer, "Measuring and improving testability of system requirements in an industrial context by applying the goal question metric approach," in *Proc. 5th Int. Workshop Requirements Eng. Test.*, Jun. 2018, pp. 25–32, doi: 10.1145/3195538.3195542.

[54] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, and M. Young, "Machine learning: The high-interest credit card of technical debt," in *Proc. NIPS Workshop Softw. Eng. Mach. Learn. (SEML)*, 2014, pp. 1–9.

[55] A. Naeem, Z. Aslam, and M. A. Shah, "Analyzing quality of software requirements; a comparison study on NLP tools," in *Proc. 25th Int. Conf. Automat. Comput. (ICAC)*, Sep. 2019, pp. 1–6, doi: 10.23919/IConAC.2019.8895182.

[56] H. R. Herdika and E. K. Budiardjo, "Variability and commonality requirement specification on agile software development: Scrum, XP, Lean, and Kanban," in *Proc. 3rd Int. Conf. Comput. Informat. Eng. (ICIE)*, Sep. 2020, pp. 323–329, doi: 10.1109/IC2IE50715.2020.9274564.

[57] T. Pfofe, T. Thüm, S. Schulze, W. Fenske, and I. Schaefer, "Synchronizing software variants with variantsync," in *Proc. 20th Int. Syst. Softw. Product Line Conf.*, Sep. 2016, pp. 329–332, doi: 10.1145/2934466.2962726.

[58] J. Krüger and T. Berger, "An empirical analysis of the costs of clone- and platform-oriented software reuse," in *Proc. 28th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Nov. 2020, pp. 432–444, doi: 10.1145/3368089.3409684.
[59] F. Gazzawe, "Requirement artefacts: Finding the missing link, framework to support requirement traceability for software developers," M.S. thesis, Loughborough Univ., Loughborough, U.K., 2021, doi: 10.26174/thesis.lboro.13880324.v1.
[60] B. Hutchinson, A. Smart, A. Hanna, E. Denton, C. Greer, O. Kjartansson, P. Barnes, and M. Mitchell, "Towards accountability for machine learning datasets: Practices from software engineering and infrastructure," in *Proc. ACM Conf. Fairness, Accountability, Transparency (FAccT)*, 2021, pp. 560–575, doi: 10.1145/3442188.3445918.
[61] B. Shahzad, I. Javed, A. Shaikh, A. Sulaiman, A. Abro, and M. A. Memon, "Reliable requirements engineering practices for COVID-19 using blockchain," *Sustainability*, vol. 13, no. 12, pp. 1–25, 2021, doi: 10.3390/su13126748.
[62] H. Scherer, A. Albers, and N. Bursac, "Model based requirements engineering for the development of modular kits," *Proc. CIRP*, vol. 60, pp. 145–150, Jan. 2017, doi: 10.1016/j.procir.2017.01.032.
[63] G. Cavarlé, A. Plantec, S. Costiou, and V. Ribaud, "A feature-oriented model-driven engineering approach for the early validation of feature-based applications," *Sci. Comput. Program.*, vol. 161, pp. 18–33, Sep. 2018, doi: 10.1016/j.scico.2018.01.001.
[64] M. Autili, P. Inverardi, and M. Tivoli, "CHOREOS: Large scale choreographies for the future Internet," in *Proc. Softw. Evol. Week-IEEE Conf. Softw. Maintenance, Reeng., Reverse Eng. (CSMR-WCRE)*, Feb. 2014, pp. 391–394, doi: 10.1109/CSMR-WCRE.2014.6747202.

ISSA ATOUM received the M.Sc. degree in computer science from Philadelphia University, Jordan, and the Ph.D. degree in software engineering from Universiti Malaysia Sarawak, Malaysia. He was the Head of the Software Engineering Department, from 2018 to 2019. He is currently an Associate Professor with the Software Engineering Department, The World Islamic Sciences and Education University, Jordan. His research interests include software engineering, software quality, requirements engineering, and machine learning.

MAHMOUD KHALID BAKLIZI received the degree in computer science from Yarmouk University, Jordan, in 2002, the master's degree in computer information systems from The Arab Academy for Banking and Financial Sciences, Jordan, in 2008, and the Ph.D. degree from Universiti Sains Malaysia (USM), in 2014, while serving as a full-time Faculty Member of WISE University, Jordan. He is currently an Associate Professor and the Director of engineering and maintenance with WISE University.

IZZAT ALSMADI received the master's and Ph.D. degrees in software engineering from North Dakota State University, in 2006 and 2008, respectively. He is an Associate Professor with the Department of Computing and Cyber Security, Texas A&M University-San Antonio. He has more than 100 conference and journal publications. He is a lead author and an Editor of several books including: *The NICE Cyber Security Framework: Cyber Security Intelligence and Analytics* (Springer, 2019), *Practical Information Security: A Competency-Based Education Course* (Springer, 2018), and *Information Fusion for Cyber-Security Analytics (Studies in Computational Intelligence)* (Springer, 2016). His research interests include: cyber intelligence, cyber security, software security, software testing, social networks, and software defined networking.

AHMED ALI OTOOM received the dual master's degrees in computer science and information technology management from the Naval Postgraduate School, USA, and the Ph.D. degree in computer science from Amman Arab University for Graduate Studies, Jordan. He is currently an Assistant Professor with Irbid National University, Jordan. He has a long and rich industrial and business experience in software development and project management. He is a Lecturer of different topics in computer science and cybersecurity. He has several published articles in software engineering and cybersecurity management. He has got high-quality professional training and has some professional certificates in project management.

TAHA ALHERSH received the B.Sc. degree in computer science from The University of Jordan, the M.Sc. degree in intelligent systems from Universiti Utara Malaysia, and the Ph.D. degree from the University of Mannheim, under the chair of artificial intelligence. His research interests include medical informatics, machine learning, and computer vision.

JAFAR ABABNEH received the B.Sc. degree in telecommunication engineering, in 1991, the M.Sc. degree in computer engineering, in 2005, and the Ph.D. degree, in 2009. He is an Associate Professor. In 2009, he joined WISE University as the Head of computer information and network systems in information technology (IT). He was the Dean of the IT Faculty, WISE University, from August 2015 to November 2020. He has published many research papers, book chapters, and books in international refereed journals and conferences. His research interests include congestion in networks, network performance (WSNs), artificial intelligence, security systems, data mining and retrieving information, and E-learning.

JAMEEL ALMALKI received the B.Sc. degree in computer science from KAU, Saudi Arabia, the master's degree in IT from UTS, Australia, and the Ph.D. degree in software engineering from Flinders University, Australia. He is currently an Assistant Professor and the Vice Dean of the College of Computers for Academic Affairs, Umm Al-Qura University, Al Lith, Saudi Arabia. He also has over three years of working experience as a Network Engineer and Administrator before becoming an Academic. His research interests include distributed collaborative software development, software engineering, and the Internet of Things.

SAEED MASOUD ALSHAHRANI received the B.Sc. degree in computer science from King Khalid University, Saudi Arabia, the master's degree in computer science from La Trobe University, Australia, and the Ph.D. degree in computer science from Flinders University, Australia. He is currently an Assistant Professor and the Head of the Information Systems Department, College of Computing and Information Technology, Shaqra University, Saudi Arabia. He has over ten years of working experience in the academic sector. His research interests include assistive technology for disabled people, big data, software engineering, and information systems. He is a member of Saudi Computer Society.