Evaluation and verification of requirements artifacts: A Checklist applied to User Stories and Prototype

Thiago S. Martins
Computing Institute
Federal University of Amazonas
Manaus, Amazonas, Brazil
thiago.martins@icomp.ufam.edu.br

Lucas S. Araújo
Computing Institute
Federal University of Amazonas
Manaus, Amazonas, Brazil
lucas.sarmento@icomp.ufam.edu.br

Ana Carolina Oran
Computing Institute
Federal University of Amazonas
Manaus, Amazonas, Brazil
ana.oran@icomp.ufam.edu.br

ABSTRACT

The quality of system requirements is essential to ensure that the software meets user needs. In the context of software engineering, specifying requirements through user stories and prototypes has become a widely adopted practice in the industry, as it facilitates effective communication among requirements, development, and testing teams. However, for these practices to be effective, information essential to developing functionality must be clearly specified and organized. To support this verification process and ensure the completeness of artifacts, this work proposes VUSP (Checklist for the Assessment and Verification of User Stories and Prototypes), focused on the verification of requirements artifacts. The feasibility of VUSP was evaluated in a study involving 38 participants from a Software Engineering course at the Federal University of Amazonas, who used the checklist to verify user stories and prototypes. Participants provided feedback on the tool's usability and effectiveness using the TAM (Technology Acceptance Model), addressing aspects such as perceived ease of use, work relevance, and behavioral intention. The results indicate that VUSP helped identify improvements in documentation and the centralization of essential information for development.

KEYWORDS

Requirements engineering, Requirements specification, User Stories, Prototypes

1 Introduction

The software industry continues to evolve and strives to meet market demands, with an emphasis on the demand for high-quality products [8]. The development of such products requires the use of best practices throughout the development cycle and operational phases [21]. A fundamental method to ensure that a system fulfills its purpose is to define clear and precise requirements, describing what the system must do, along with its constraints and properties [32].

The software requirements specification document presents all the functionalities and capabilities that a software product must provide and establishes any prior constraints that the system must comply with [28]. While the term requirement refers to the objective to be achieved, the specification defines how this objective will be accomplished [25]. To make the specification accessible to different members of a development team with diverse expertise, requirements can be specified through User Stories (US) and prototypes [29, 4]. US descriptions use simple language to simulate the user's perspective when describing functionality, making its scope accessible to the development team [19, 2]. At the same time, prototypes are visual representations that define layout rules to satisfy the corresponding requirements [30].

In the context of the software industry, it is common for teams to face significant challenges related to requirements documentation, mainly when such documentation is carried out through US and prototypes [24]. Often, US are recorded vaguely or incompletely, lacking clear acceptance criteria or containing important gaps regarding business rules, preconditions, and impacts on existing functionalities [26]. Prototypes, in turn, frequently omit expected behaviors, interactive elements, or usability details, which may lead to divergent interpretations among analysts, developers, and testers [30]. Therefore, it is important to use mechanisms that support the verification and improvement of these artifacts during the specification phase, reducing ambiguities and increasing the quality of technical communication [26].

This study introduces VUSP (Checklist for the Assessment and Verification of User Stories and Prototypes), which was created to assist in the evaluation of requirements documentation artifacts, such as US and prototypes. The checklist was based on the work of Oran et al. [26], ReComP, in order to create a single artifact accessible to the entire development team. Furthermore, this paper analyzes the feasibility and user perception when using VUSP, focusing on perceived ease of use, work relevance, and behavioral intention. The experiment also identifies opportunities for improvement and future work derived from the checklist.

This paper is organized as follows: Section 2 presents concepts and related work. Section 3 introduces the VUSP. Section 4 describes the methodology used. Section 5 presents the results and discussions. Finally, Section 6 presents the conclusion drawn from the application of the checklist.

2 Background

2.1 Requirements Specification

According to Doe [13], a requirements specification can be defined as the specification for a particular product, program, or set of programs that perform certain functions in a specific environment. Such a specification represents the agreements between the customer and those responsible for software development [1]. Thus, its role is fundamental in the communication and documentation of customer requirements, and it can also be used to validate the final product [6]. Good requirements specifications involve correct, unambiguous, complete, and consistent descriptions, and are usually written in natural language so that stakeholders can understand and feel comfortable agreeing to them [1].

Requirements communication plays an essential role in software development projects, as it enables coordination among customers,

business areas, and the development team [1]. It refers to the process of conveying the customer's needs to the development team, to guide the implementation of the solution [16]. Effective requirements communication promotes a common understanding between stakeholders and the development team regarding which requirements are relevant and what their meaning is for the system to be developed [26]. This alignment reduces ambiguities and contributes to ensuring that the final product meets the identified needs [7].

2.2 User Stories

Beck and Fowler [5] proposed the use of software requirements through User Stories (US), which are short descriptions of system functionalities from the perspective of a stakeholder. Cohn [11] structured these descriptions following the format "As a <stakeholder>, I want <some goal> so that <some reason>," allowing a simple structuring of the functionality description. Jeffries [18] defined that US can be composed of three elements: (i) the card, which serves as a brief description, as proposed by Beck and Fowler [5]; (ii) the conversation, which represents ongoing communication between the client and programmers and can be complemented with documents that define business rules and quality attributes, such as prototypes; (iii) the confirmation, implemented through acceptance tests, whose purpose is to provide the necessary assurance that the work has been completed as expected. At the end of the iteration, programmers demonstrate the functioning of the acceptance tests, confirming successful delivery.

There are several reasons to analyze the quality of software requirements, particularly User Stories (US) [24]. The absence of detailed specifications through this artifact can lead to the implementation of functionalities that do not adequately meet user expectations. This deficiency in requirements definition compromises clarity regarding what should be developed and hinders the delivery of solutions that are compatible with the needs of the target audience [31]. Furthermore, the quality of design, implementation, testing, and user acceptance directly depends on the quality of the specified requirements [35].

2.3 Prototype

Prototypes are an approach to communicate design interface ideas and to evaluate them from a broad perspective, including the development team, testers, and clients [26]. System prototyping can be understood as a typical and preliminary step aimed at creating an iterative artifact that assists in decision-making and product vision [10].

Prototypes can be classified into three types: low, medium, and high fidelity, varying according to the level of detail and the fidelity with which they should be followed. This difference of levels aids in using prototypes as a communication tool throughout various stages of software projects [27]. Additionally, prototypes can be used in conjunction with other artifacts, such as User Stories (US), to support the understanding of functional requirements implementation and are commonly employed together in agile contexts [29, 3].

2.4 Related Work

Oran et al. [26] present ReComP (Framework for Requirements Communication based on Perspectives), a framework for evaluating and improving artifacts in requirements communication within software projects. Developed using the Design Science Research method, it aims to support communication based on the perspectives of the development team. In two cycles of empirical studies, ReComP reduced 77% of the problems identified by developers in the US and 100% of the problems identified by testers in use cases. The tool received positive acceptance regarding ease of use and usefulness. ReComP demonstrated effectiveness in reducing issues in requirements specifications, but it has limitations, as it evaluates the perspectives of developers and testers separately, without integrating these views into a single artifact, and it does not explicitly address how prototypes relate to the US.

Macedo et al. [12] presented UIProtoCheck, a checklist for semantic inspection of user interface prototypes. While traditional approaches focus on usability problems in already implemented software, UIProtoCheck evaluates prototypes before implementation, verifying whether they correctly represent the functionalities described in the requirements. The checklist analyzes three perspectives: displayed information, input fields, and interaction elements, assessing their correctness, completeness, and consistency. A study with 12 participants showed 67% effectiveness in identifying semantic errors in prototypes, demonstrating its usefulness as a verification tool in agile environments with limited resources.

Fonseca et al. [14] compared two approaches for incorporating usability and user experience requirements in agile methodologies: USARP and ACUX. The study involved 32 participants in a controlled experiment. The results showed that USARP demonstrated higher ease of use, while ACUX showed greater capability in specifying user interactions and system functionalities. Both approaches were praised for their practicality and for providing standards for writing requirements. Participants reported difficulties in interpreting some guidelines. The authors suggest investigating the combined use of both methods and replicating the experiment with a larger sample. These approaches focus solely on usability and user experience aspects, without addressing other elements of requirements specification.

Amna et al. [3] present AmbiTRUS (Ambiguity Tracking and Resolution for User Stories), a framework for ambiguity analysis in the US that addresses issues at different linguistic levels. It is composed of quality criteria for 13 types of ambiguity problems, classified into four linguistic levels and linked to four types of requirements quality issues. The criteria were selected from three established frameworks. In a controlled experiment with master's students, the framework did not demonstrate clear effectiveness in identifying ambiguity problems, but it was considered useful despite its complexity. The authors suggest the development of a tool based on Natural Language Processing (NLP) and evaluation through a usability study. AmbiTRUS provides a detailed approach for identifying ambiguities in the US, but it focuses exclusively on linguistic ambiguities, and its complexity suggests the need for automated support.

Unlike previous works, VUSP presents an integrated approach that connects US and prototypes in a single evaluation checklist, encompassing multiple perspectives of the development team. While ReComP separates its analysis by specific roles, VUSP provides a unified and complementary view of the artifacts. In contrast to UIProtoCheck, which focuses exclusively on the semantic inspection of prototypes, VUSP addresses both US and prototypes in a combined manner. The checklist also overcomes the limitations of USARP and ACUX by addressing aspects beyond usability, including functional and technical elements fundamental for development. Additionally, VUSP presents lower application complexity compared to AmbiTRUS, enabling immediate use without the need for specific automated tools. These differentiating characteristics make VUSP a comprehensive and practical checklist, capable of supporting agile teams in verifying and improving the quality of their requirements artifacts.

3 Evaluated Artifact: The VUSP Checklist

Reading techniques are systematic approaches used in software inspections to guide reviewers in identifying faults in artifacts [12]. They provide guidelines on how to conduct document analysis, increasing the effectiveness of the review. Among these techniques, Checklist-Based Reading (CBR), proposed by Thelin et al. [34], stands out. CBR focuses on detecting faults relevant from the user's perspective, utilizing checklists that contain information guiding reviewers on what to consider during the inspection [12].

As the evaluated artifacts (US and prototypes) can be used at different stages of development by an agile team [11, 10], it is essential to ensure their quality, organization, and completeness. Accordingly, VUSP was developed, a checklist for the evaluation and verification of US and prototypes. It was built based on discussions from related works, taking as its main foundation the structure proposed by Oran et al. [26] and their research with the ReComP framework. Additionally, insights from Jaiswal and Venkataraman [17] assisted in identifying challenges in the creation and evaluation of prototypes, and the research by Macedo et al. [12] supported the development and structuring of the checklist.

To construct the checklist, the checklist principles proposed by Chernak [9] and Laitenberger [20] were used, employing two main components: where to look and how to detect. To define both components, the different contexts in which US and prototypes are used were analyzed, in addition to considering suggestions from industry professionals and perspectives from related works.

To evaluate the quality of the documentation, US and prototypes, different perspectives were considered to facilitate the identification of gaps and inconsistencies in the analyzed artifacts. From this analysis, three main perspectives were identified: (i) informational content, which refers to the clarity and completeness of descriptions; (ii) technical elements and relationships, including functional and non-functional requirements, dependencies, and business rules; and (iii) usability aspects and compliance with visual and interaction standards.

Based on these perspectives, it was possible to structure a set of statements that serve as verification criteria. For each of them, statements were elaborated to support the identification of issues related to Correctness, Completeness, and Consistency of the documentation, according to definitions widely used in Software Engineering:

- Correctness: the documentation accurately describes the functionality according to the system's objectives;
- Completeness: all elements necessary for the development and validation of the functionality are included;
- Consistency: there are no contradictions between the documentation and other artifacts, such as requirements, prototypes, and the design system.

Table 1 presents the relationship between perspectives, definitions, and checklist statements, as well as the expected location of each piece of information.

The questions were organized based on the statements intended to be verified. The analysis was conducted considering not only the presence of the elements but also the clarity of their description and adherence to expected practices in projects focusing on traceability, role-based communication, and preparation for implementation.

The checklist contains 28 statements focused on requirements documentation through US and prototypes, addressing aspects such as covered requirements, level of detail, initial conditions, and other aspects cited in the literature [32]. For each statement, the user can select one or more of the following responses: (i) prototypes – if the information is present in the prototype, (ii) user story – if the information is present in the user story, (iii) not applicable – if the evaluated artifact is not related to the US or prototypes.

In addition to the statements included in the checklist, and with the aim of evaluating the proposed checklist, participants indicated which information they considered useful for performing their tasks through the following question: "Is this information useful?" The purpose of this question was to identify which data each participant considered relevant in the specification to support the execution of their activities. This allows the person responsible for specifying the requirement to identify within their team which information needs to be detailed or clarified further in the documentation.

The implementation of the checklist regarding the responses to be obtained can be observed in Figure 1. It is worth noting that two professionals with prior experience in the software industry participated in the creation of VUSP, working in different organizational contexts. One of the professionals has over 14 years of experience in requirements engineering activities, including elicitation, documentation, and artifact verification. His experience also covers practices in the inspection of technical documents and direct participation in software development projects using agile methodologies. The involvement of these professionals was essential to ensure the checklist's alignment with the practices and real challenges faced by multidisciplinary teams in industrial contexts. The other professional has been working for four years as a mid-level iOS developer and contributed the practical perspective of someone who experiences daily the negative impacts of using incomplete, ambiguous, or disorganized requirements artifacts. The participation of these professionals was crucial to attempt to reflect in the checklist the challenges faced by multidisciplinary teams in real software development environments.

Table 1: Documentation Verification Checklist

Where to Look	ID	Statement	How to Detect
US	1	The documentation includes the user's perspective (As xxx ... I would like to ... to...)	Check for the presence of the standard user story structure; identify role, action, and justification.
US	2	It is clear which Functional Requirement is being addressed by the documentation	Check if there is an explicit association with a functional requirement, with identification (ID, name, or code).
US	3	The documentation describes which non-functional requirements must be met	Look for specific mentions of requirements such as performance, accessibility, security, or usability.
US	4	The documentation describes how the non-functional requirements will be met	Verify if there are plans, parameters, or technical justifications associated with meeting the requirements.
US	5	The documentation includes who is responsible for implementing the functionality/improvement (name and role)	Identify the associated responsible person(s), including their role (e.g., PO, dev, QA).
US, Prototype	6	The documentation includes a detailed description of the improvement and/or functionality to be implemented	Evaluate if there is a clear narrative with beginning, middle, and end, covering the expected behavior.
US, Prototype	7	The documentation maps which user stories it depends on to begin development	Check if there is a dependencies section or links between related functionalities.
US	8	The documentation includes business rules (constraints/assumptions)	Look for explicit rules, value limits, conditions, or expected exceptions.
US, Prototype	9	The documentation defines how the functionality is accessed (how to reach that functionality)	Verify instructions about the access path to the functionality, such as menus or buttons.
US, Prototype	10	The documentation includes initial conditions before the functionality is executed (preconditions)	Evaluate if there are states, prerequisites, or initial configurations described.
US, Prototype	11	The documentation includes final conditions after the functionality is executed (postconditions)	Check for the presence of expected states or results after completing the operation.
US, Prototype	12	The documentation defines the expected behaviors and interactions with the functionality/improvement (navigation, errors, loading, shimmers, etc.)	Identify descriptions of interface behavior in different states.
US	13	The documentation defines which functionality will be impacted by the increment	Check if there is impact mapping on existing functionalities.
US	14	The acceptance criteria and their scenarios are described in the documentation	Analyze the presence of formal criteria (Given-When-Then or equivalents).
US	15	The acceptance criteria are testable and measurable	Verify if the criteria can be objectively validated and quantified.
US	16	The complexity of the functionality/improvement described in the documentation is compatible with the size of a sprint	Evaluate if the scope of the user story is adequate to be developed within the sprint period.
US, Prototype	17	If the functionality/improvement includes access to APIs and services, they are described	Identify references to integrations, endpoints, parameters, and expected responses.
Prototype	18	All colors to be used are specified in the documentation	Check if there is a specific indication of color palette or hexadecimal/RGB codes.
Prototype	19	All fonts to be used are specified in the documentation	Identify references to font families, sizes, weights, and styles.
US, Prototype	20	All interactions with the functionality/improvement are being mapped in the documentation	Verify if all possible user interactions and their results are described.
Prototype	21	The components in the documentation exist in the design system	Compare visual elements with the component library available in the design system.
Prototype	22	The design system components are being indicated	Check if there are explicit references to the names or codes of components in the design system.
Prototype	23	The design system components can be customized according to the documentation	Evaluate if there are instructions about variations or customizations of standard components.
US, Prototype	24	The documentation meets accessibility requirements	Verify compliance with guidelines such as WCAG or references to accessibility practices.
US, Prototype	25	The documentation includes accessibility descriptions	identifies specific descriptions about how to make the functionality accessible to all users.
Prototype	26	The documentation includes responsiveness expectations for larger or smaller screens	Check instructions about interface behavior on different screen sizes.
US, Prototype	27	The documentation covers all usage scenarios, including empty states, errors, and loading	Look for descriptions of behavior in exception cases and special interface states.
US, Prototype	28	The documentation includes all state messages: error, success, loading...	Verify if there are specific texts for each type of message that can be displayed to the user.

Affirmation		Where?		NO	N/A	Is this information useful?	
		user story	prototype			YES	NO
1	The documentation includes the user's perspective (**As** xxx ... **I would like to** ... **in order to** ...)						

Figure 1: VUSP - Implementation

4 Experimental Study Planning and Execution

To guide the process during the research, the following research question was defined: "How can the quality and completeness of requirements specification artifacts (US and prototypes) be verified so that requirements information is clear and accessible to different profiles of members in software development teams?". The objective of this research is to develop and evaluate a checklist that enables the verification of the quality and completeness of requirements specification artifacts, specifically US and prototypes, so that requirements information is clear and accessible to different profiles of members in software development teams.

To meet the objective and answer the research question, VUSP was developed, a checklist designed to assist in the verification of requirements artifacts, ensuring that the information essential for the development of functionalities is clearly specified and organized. To evaluate the feasibility of the checklist as a support tool for requirements specification, it was applied in a course simulating industry experiences, where students are required to use their knowledge to create a real-world project over several sprints, to verify whether the objective was achieved and what improvements could be made in future versions.

Software Engineering Practices (PES) is a 7th-semester course with a workload of 90 hours, aimed at providing students with a practical environment for software development using computational tools. The course seeks to train students in the use of modern technologies applied throughout the software engineering process, including the setup of cloud environments for configuration management and the deployment of solutions ready for consumption by end users.

During the course, the 46 enrolled students were divided into eight multidisciplinary teams, each with approximately 4 to 6 members. They were challenged to develop innovative software, applying techniques and tools widely used in the market, and experiencing all stages of the software development process. For project management, the Scrum framework was adopted, including the execution of its ceremonies. To support this, students underwent introductory training on agile methodologies and the Scrum framework. The roles performed in the projects included: Scrum Master, Requirements Engineer (Product Owner), Developers, Designers, and Testers.

During the study, out of the 46 students enrolled in the course, only 38 students fully completed all artifacts, including the VUSP checklist and evaluation questionnaires. Within this group, the distribution of roles was as follows: 16 Developers, 8 Product Owners, 7 Testers, 5 Designers, and 2 Scrum Masters. Although the incomplete participation of part of the class reduced the sample size compared to the total number of students involved, there were representatives from all teams, and it was possible to identify various issues in the documentation.

During the requirements elicitation phase, the students applied techniques such as Design Thinking, interviews, and questionnaires to identify functional and non-functional requirements. The project documentation consisted of US, UML diagrams, and interface prototypes. From the first sprint, students were encouraged to deliver functional increments and deploy them on the web, avoiding waterfall-based approaches. In total, the students developed their projects over five sprints during the semester.

This study was conducted during the planning meeting of the third sprint of the project, at a time when the first two sprints had already been specified and developed. Thus, the participants were already familiar with the context and requirements of their respective projects.

For a more in-depth evaluation of the VUSP checklist, this study employed a mixed-methods approach, combining quantitative and qualitative instruments.

4.1 Artifacts

All volunteer students signed the Informed Consent Form (ICF), allowing the data collected to be used in the study before performing any activities related to the experiment, clarifying the objectives, ensuring confidentiality, and permitting withdrawal at any time.

In addition to the ICF, participants completed a characterization questionnaire aimed at understanding the context of each study participant, characterizing their level of familiarity and experience with the topics addressed in the study. The questionnaire was divided into five sections: (i) knowledge about software requirements specification; (ii) knowledge about requirements document inspection; (iii) knowledge about prototype inspection; (iv) experience with US; and (v) experience with software development.

Afterwards, the participants used the VUSP checklist (Section 3) to analyze the requirements specifications of the US with prototypes from their projects.

4.2 Participants

The study, including the checklist and evaluation questionnaires, was conducted with 38 undergraduate Software Engineering students from the Federal University of Amazonas, analyzing their experience with the use of VUSP as a tool to improve the quality of US and prototypes in their projects. Data were collected during the classes of the course ICC416 - Software Engineering Practice, offered to students in the first semester of 2025.

All participants already had some experience with requirements specification and inspection of requirements documents in the classroom, 61% had prior contact with requirements specification in industry, and 95% had experience with requirements document inspection. Additionally, 97% had experience with prototypes in the classroom, and 45% had some experience with prototypes in the workplace. Regarding the use of US, 100% of participants had used them in the classroom, at least 92% had experience writing user stories, and 47% had used US in industry. Finally, approximately 81% of the students are currently working in industry, with experience ranging from 3 months to 6 years.

4.2.1 Checklist Evaluation Artifacts. Immediately after using the VUSP checklist, participants were invited to complete an evaluation questionnaire based on the Technology Acceptance Model (TAM), aiming to capture their perceptions of the tool. The TAM is a widely used theoretical model for understanding how users accept and intend to adopt new technologies [15]. The instrument uses a Likert scale, with response options ranging from: (i) strongly disagree, (ii) disagree, (iii) neither agree nor disagree, (iv) partially agree, and (v) strongly agree.

The questionnaire was structured to cover the main constructs of the model, including perceived usefulness, ease of use, relevance, and behavioral intention to use. In addition to closed-ended questions, open-ended questions were included to allow participants to freely report their impressions of the experience with the checklist. These questions enabled the capture of perceptions regarding possible improvements in the requirements of their projects and suggestions for refinement in the structure of VUSP.

This mixed-method evaluation approach, combining quantitative and qualitative data, enabled a deeper analysis of participants' individual perceptions, providing a broader understanding of the value, limitations, and potential application of VUSP in the context of verifying requirements artifacts in educational environments with agile practices.

In addition to the TAM-based questionnaire, during the Sprint retrospective at the end of the development sprint proposed by the course, participants also completed a communication evaluation form. This instrument contained open-ended and reflective questions aimed at identifying challenges faced, perceived improvements in project communication and documentation, as well as suggestions for the future enhancement of the tool.

All artifacts and additional details are available in the supplementary materials [22].

5 Results and Discussions

This section presents the results and discussions generated from the data collected during the study using the artifacts described in Section 4. All findings from participants' responses to the checklist, the TAM questionnaire, and the communication improvement perception questionnaire will be presented, aimed at identifying perceived impacts on the clarity and organization of requirements after using the checklist. All the collected data can be found on the dataset available on the supplementary materials [23].

5.1 Data Collected by the Framework

The analysis of the 28 questions in the VUSP checklist showed that 60% of the information was recorded in User Stories, 30% in prototypes, and approximately 10% was missing, highlighting significant gaps, particularly in non-functional requirements (≈ 40%), acceptance criteria (≈ 55%), and accessibility and responsiveness aspects (≈ 35%). Overall, the checklist revealed that approximately 75% of the expected information was present, leaving around 30% of gaps to be addressed.

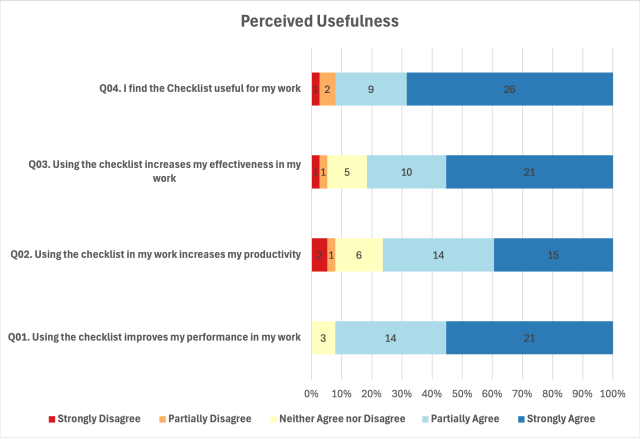

Figure 2: VUSP - Perceived Usefulness

5.2 Analysis of the Perceived Usefulness of VUSP

Figure2 illustrates participants' perceptions according to the following statement:

Q01. Using the checklist improves my performance in my work: The results showed a positive perception regarding performance improvement after using the checklist: 55% of participants strongly agreed that their task performance improved by using VUSP. Additionally, 37% of participants expressed that they agreed on performance improvement, indicating an overall positive perception, although with potential for enhancement. A portion of 8% provided neutral responses, reflecting a lack of clarity about improvements after using VUSP. No participants expressed partial or total disagreement regarding performance improvement post-checklist use, indicating general satisfaction with this aspect. The predominantly positive perception of performance improvement suggests that VUSP achieved its main objective of assisting in the verification of requirements artifacts. The absence of disagreement demonstrates recognition of the checklist's added value, although neutral responses indicate opportunities to make the benefits more evident to all participants.

Q04. I find the checklist useful for my work: The responses to this question show that the majority of users believe VUSP is useful for the activities they perform, with 68% of participants strongly agreeing that the tool helped carry out their tasks, and 24% partially agreeing on the checklist's usefulness, indicating a high positive perception. A minority (5% partially disagree and 3% strongly disagree) believe that the checklist was not useful for their work. The strong approval regarding the usefulness of VUSP indicates its alignment with the practical needs of the participants. The small portion of disagreement provides an opportunity to investigate specific cases where the checklist may not meet certain expectations or work contexts, allowing adjustments to broaden its applicability.

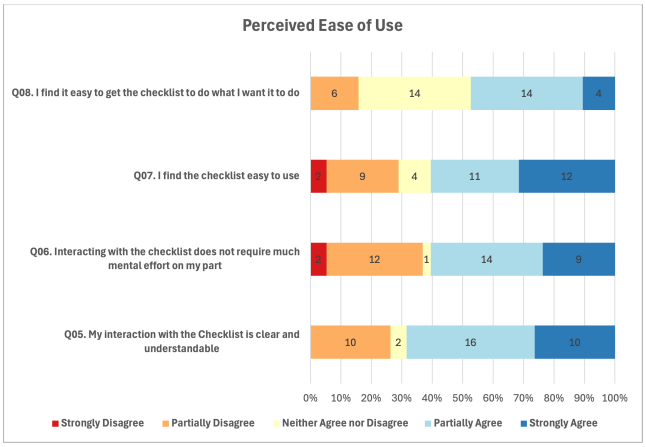

Figure 3: VUSP - Perceived Ease of Use

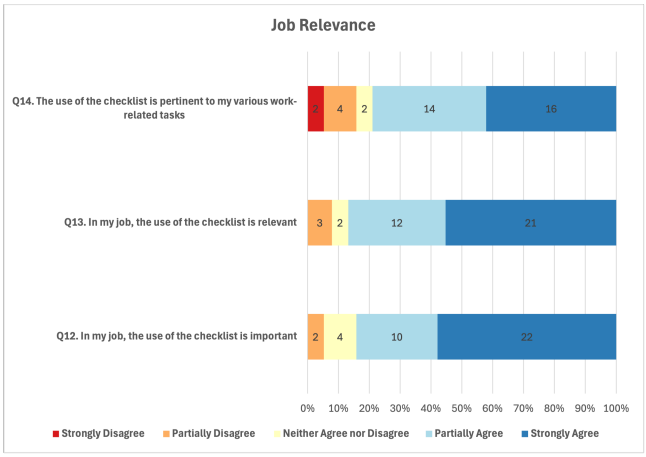

Figure 4: VUSP - Work-Related Relevance

5.3 Analysis of the Perceived Ease of Use of VUSP

Figure3 illustrates the perceived ease of use by participants according to the following question:

Q07. I find the checklist easy to use: This question investigated participants' perception regarding the ease of use of VUSP. The results showed a predominantly positive assessment, with 32% of participants strongly agreeing on the ease of use of the system. Additionally, 29% of respondents partially agreed, indicating an overall favorable perspective, although with potential for optimization. A portion of 10% of participants neither agreed nor disagreed. Conversely, a significant portion expressed disagreement, with 24% partially disagreeing and 5% strongly disagreeing, suggesting that VUSP is not perceived as easy to use by all individuals. The results indicate that the majority of participants recognize the usability of VUSP, meeting the expectations of most users. The neutral and disagreeing responses highlight areas that require improvement, providing valuable insights for enhancing the user experience. It is important to note that the perception of ease of use of VUSP may vary due to individual factors. The more pronounced split in responses points to an area for improvement. Although the majority have a positive perception, a significant proportion of negative responses suggests the need for refinements in the checklist's interface, documentation, or workflow to make it more accessible and intuitive for all user profiles.

5.4 Analysis of the Job Relevance of VUSP

Figure4 illustrates the perceived work-related relevance by participants according to the following questions:

Q12. In my job, the use of the checklist is important: The analysis of the results showed strong agreement among participants. A substantial majority of 58% strongly agreed with the statement, indicating a robust perception of the checklist's importance for performing their tasks. Additionally, 26% of respondents partially agreed, reinforcing the overall perception of value, although with potential caveats. Conversely, a minority of 5% partially disagreed, and another 11% neither agreed nor disagreed with the statement, suggesting that the checklist's importance may not be universally recognized or that some participants may lack clarity about its impact. Total disagreement was 5%, further reinforcing the majority perception of importance. The strong agreement of most participants demonstrates that VUSP is perceived as an important element for the development team. This recognition of its relevance indicates that the checklist aligns with users' workflow. The few neutral and disagreeing responses highlight opportunities to better communicate the tool's value in different application contexts.

Q13. In my job, the use of the checklist is relevant: This statement showed results similar to the previous one, although with slight variations. Strong agreement remained high at 55%, evidencing the perception of the checklist's relevance in the participants' work context. A considerable portion of 32% also partially agreed, reinforcing the view that the checklist is applicable, but with possible limitations or adaptation needs. Neutral responses were 5%, and partial disagreement was 8%, both minor, suggesting that the checklist's relevance is generally recognized. Total disagreement was not reported, indicating consensus on the checklist's relevance in the workplace. The consistently positive results and absence of total disagreement demonstrate that VUSP has recognized practical value, although small adjustments could further enhance its acceptance.

Q14. The use of the checklist is pertinent for my various work-related tasks: The evaluation of this question revealed a more balanced distribution of responses. Strong agreement remained substantial at 42%, but lower than in previous questions, suggesting that while the checklist is considered relevant, its applicability may be more restricted to specific tasks. Partial agreement from 37% of participants indicated a significant perception of relevance, although with possible limitations or the need for adaptations for different tasks. Neutral responses at 11% were higher than in previous questions, indicating greater uncertainty among participants regarding the checklist's relevance to their activities. Total and partial disagreements were minor, remaining at 5% each, reaffirming the overall perception of relevance, although with caveats. The balanced distribution of responses regarding relevance for various tasks suggests that VUSP may have variable applicability

depending on the scope of work for the different roles performed by participants during the Sprint. Enhancing the checklist to address a broader range of professional activities could increase its versatility and perceived value in different usage contexts.

5.5 Analysis of the Behavioral Intention to Use VUSP

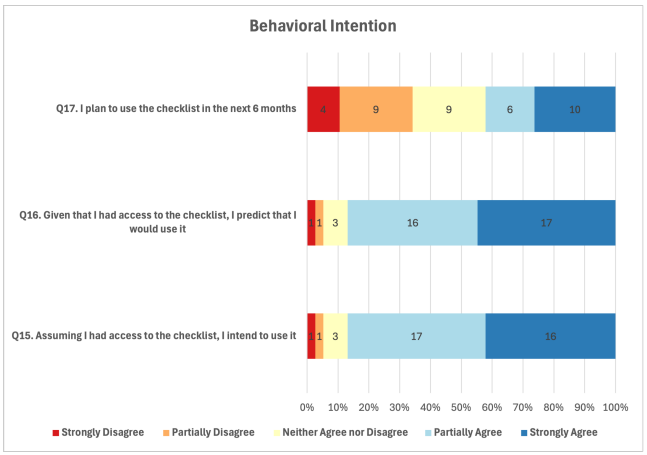

Figure 5: VUSP - Behavioral Intention

Figure5 illustrates the intention and predicted use of VUSP by participants, considering a hypothetical scenario of unrestricted access to the tool according to the following questions:

Q15. Assuming I had access to the checklist, I intend to use it: The analysis of this question reveals considerable interest among participants. A substantial portion of 46% partially agreed with the statement, indicating a propensity to use the checklist if it were available. Strong agreement by 43% of participants also demonstrates significant interest among a considerable number of respondents. Conversely, a minority of 3% expressed partial disagreement or adopted a neutral stance (8%), suggesting that some participants may lack clarity about the checklist's potential use or that there are factors influencing their intention. The high level of behavioral intention reflects the perceived practical value of VUSP. This positive result indicates that, once initial access barriers are overcome, the checklist would have good acceptance in a real work environment, confirming its relevance as a support tool for requirements verification.

Q16. Given that I had access to the checklist, I predict that I would use it: This question presents results aligned with the previous one, although with slight variations. Strong agreement by 45% of participants remains high, indicating that a significant number of participants foresee using the checklist if it were accessible. Partial agreement from 42% also demonstrates a projected use by a considerable portion of respondents. Neutral responses at 8% and partial disagreement by 2% of participants are minor, suggesting that most participants have a clear view of the checklist's potential use. The absence of total disagreement reinforces the idea that, in general, the checklist is seen as a tool that participants would use if available. The consistency between intention and predicted use responses suggests participants' conviction regarding the practical value of VUSP. This pattern indicates that, beyond initial interest, users envision continued application of the tool in their work processes.

5.6 Qualitative Results

The data in this section were obtained from participants' opinions (referenced as PP1 to PP38) through the evaluation questionnaire administered after the end of the development cycle.

Regarding the previous Sprint, do you consider that there was an improvement in the project documentation? Yes/No? Why? The evaluation of the responses shows positive results, with the majority of participants recognizing significant improvements. The main improvements highlighted are: (i) more detailed and well-written User Stories, resulting in fewer implementation problems (28.9% of participants), as indicated by statements such as "we improved our user stories, and as a result the devs are facing fewer implementation issues" (PP2) and "the user stories are more detailed and the prototypes are clearer" (PP28); (ii) prototypes developed in advance and with greater detail (15.8% of participants), evidenced by comments like "the prototypes were created in advance" (PP5) and "we created a Figma with the prototypes to standardize the documentation" (PP34); (iii) more precise activity descriptions (10.5% of participants), as mentioned in "there was an improvement in documentation and activity descriptions" (PP4) and "we detailed the activity cards more thoroughly" (PP35); and (iv) implementation of clearer Definition of Done criteria (7.9% of participants), expressed in statements such as "we are analyzing which are the minimum functionalities required to consider the user stories as done" (PP12) and "we included Definition of Done criteria" (PP22). Some participants (5.3%) mentioned that, despite the current quality, there is still room for improvement and the possibility of incorporating suggestions from the checklist, as indicated in the remark "it would be good to use some suggestions from the checklist" (PP1).

Comparing the previous Sprint and the use of the checklist in the project, do you consider that there was any impact on development? Yes/No? Which? The responses show a predominantly positive perception, with several specific benefits identified by the participants. The most highlighted aspects include: (i) better detailing of User Stories, reducing the need for rework by developers (18% of participants), as indicated by a participant who stated: "I was able to detail the User Stories a bit better, which made the devs know more precisely what was to be done, without much rework" (PP3); (ii) greater awareness of functionality coverage through documentation (10% of participants), exemplified by the comment: "There was an impact in the sense that we became more aware of functionality coverage through documentation" (PP6); (iii) identification of deficiencies in User Stories and prototypes (15.8% of participants), as mentioned: "It helped to notice some deficiencies in the User Stories or in documentation such as in prototypes" (PP11); (iv) ease in task prioritization and documentation organization (7% of participants), as indicated by: "Very good for prioritizing things and listing important items without leaving them forgotten" (PP18) and "It facilitated the organization of the sprint documentation" (PP16); (v) greater calmness and certainty during development, as reported (7% of participants): "We developed with more calmness

and certainty about what we should do" (PP23). Several participants mentioned plans to incorporate VUSP elements in future Sprints, aiming to improve documentation with more complete information: "We are considering including in the documentation some items questioned by the checklist that we consider relevant" (PP12). Some participants expressed interest in applying it more fully in upcoming iterations (5% of participants): "The checklist is very good and we will take it into account in the next sprint" (PP5). The evaluation highlights how VUSP contributes to more structured and complete documentation, resulting in a more efficient and assertive development process.

5.7 Perceived Usefulness

The analysis of the data collected through the question "Is this information useful?", applied after each checklist item, reveals considerable differences in the perceived usefulness of requirements documentation fields among the roles performed by the study participants. According to Testers, 98% of the fields are considered necessary for the creation and execution of test cases, demonstrating the importance of complete documentation for this profile. Among Developers, approximately 65% of the information was deemed useful, while 35% was evaluated as unnecessary, including error-handling scenarios, final conditions, field masks, the person responsible for executing the functionality, navigation rules between screens, and non-functional requirements. For Product Owners, around 78% of the fields were considered necessary, with less emphasis on technical implementation details. Designers showed interest in approximately 70% of the fields, prioritizing those related to interface and user experience. Finally, Scrum Masters considered about 75% of the fields useful, focusing mainly on those related to requirements management and activity description, thus confirming that the perceived usefulness of documentation varies significantly depending on the role performed in the project.

By applying Cronbach's alpha coefficient, a value of 0.782 was obtained, which is considered acceptable and satisfactory according to the criteria established by Taber [33], who classifies values between 0.7 and 0.8 as indicating good internal consistency.

5.8 Discussions

The evaluation of the VUSP checklist, conducted as a feasibility test, showed significantly positive results among the 38 research participants. The data reveal strong acceptance regarding the checklist's usefulness, with 55% of participants fully agreeing that it improved their performance and 68% considering it useful for their work. The qualitative analysis confirmed tangible benefits, such as more detailed user stories, more complete prototypes, and a reduction in implementation issues.

Regarding usability, the results were more heterogeneous: 61% of participants considered the VUSP easy to use, while 29% reported some degree of difficulty. This distribution indicates opportunities for improving the user experience and clarifying instructions. However, the perceived relevance in the work environment was highly positive, with 84% recognizing its importance for the professional context.

A particularly promising indicator was the high intention of future use, with 89% expressing interest in using the checklist if it were available. The practical benefits identified included reduced rework, proactive identification of deficiencies in documentation, facilitation of task prioritization, and increased confidence during development.

Despite the predominantly positive feedback, opportunities for improvement were identified, particularly regarding the refinement of instructions and the expansion of aspects covered by the verification process. To address criticisms related to the checklist's ease of use, reflected by 29% of negative or neutral responses in the survey, it is considered to adapt the VUSP for use with AI tools (ChatGPT, Gemini, etc.) to reduce usage complexity and promote broader adoption in the industry for requirements specification.

The results suggest that VUSP represents a significant contribution to requirements engineering, providing a structured mechanism that can be effectively integrated into agile development practices, potentially becoming a standard tool for teams seeking to verify the quality of their requirements artifacts. Although related to the agile development process, VUSP is not tied to specific management tools (Jira, Trello, Figma, etc.), as each technical team may have particular needs, and such dependency could limit the adoption of the checklist.

5.9 Threats to Validity and Limitations

As with all studies, this research is subject to threats that may influence the validity of the results obtained. In this section, the existing threats are discussed along with the measures taken to mitigate them, where possible:

(1) Academic versus real-world context — The projects developed during the course, although simulating industry situations, do not have the same constraints as real projects. To mitigate this threat, the Software Engineering Practices course was structured to replicate processes and challenges found in professional environments, including deadlines, quality requirements, and teamwork.
(2) Construct validity — The reliance on subjective perceptions through the TAM may not precisely reflect the checklist's effectiveness. To minimize this risk, the study was complemented with qualitative questions that allowed participants to elaborate on specific impacts observed in the documentation and development process.
(3) Hawthorne effect — Participants, aware that they were being studied, may have altered their behavior or evaluations. This threat was mitigated by conducting anonymous assessments, without personal questions, focusing only on the requirements artifacts used in the activities.
(4) Limited period of use — Using the checklist for only one sprint may not fully capture its long-term benefits. This limitation will be addressed in future studies with longer observation periods.

In addition to the threats to the validity of the study results, it is important to highlight that the VUSP presents three main limitations:

(1) The checklist does not evaluate the quality of requirements elicitation, nor does it verify whether this process was conducted correctly. VUSP focuses on assessing and improving the communication of software requirements among development team members, considering the artifacts used.

(2) The evaluation guidelines are limited to User Stories and prototypes, not addressing other types of requirements artifacts that may be used in different development contexts.
(3) The checklist was tested only in an academic environment, making its validation in industrial settings with different organizational contexts, application domains, and project scales necessary.

6 Conclusion

This work aimed to propose and evaluate VUSP, a checklist for the assessment and verification of User Stories and prototypes, developed to help development teams improve the quality of requirements documentation. The proposal arose from the need to support development teams in identifying gaps in requirements documents, promoting more effective communication among the various roles involved in the process. The paper also presents a study that assessed the feasibility and effectiveness of the checklist based on data collected from participants enrolled in the Software Engineering Practice course during the first semester of 2025 at the Federal University of Amazonas, who conducted self-assessments of their skills in the areas relevant to the study.

The results collected through the Technology Acceptance Model (TAM) revealed a positive perception regarding the checklist's usefulness, with 68% of participants considering it helpful for their work. The analysis also showed that 61% of participants found VUSP easy to use, although opportunities for improvement were identified in this aspect. The qualitative evaluation confirmed practical benefits, such as more detailed documentation and reduced rework, with 89% of participants expressing an intention to use the checklist in future projects if they had access to it. Based on these findings, necessary improvements were identified, including refinement of terminology, simplification of instructions, and expansion of the types of artifacts analyzed.

Regarding the research question presented in Section 4, VUSP can serve as a structured inspection tool, supported by 28 items that guide the evaluator on where to look and how to detect relevant information in User Stories and Prototypes, enabling the identification of gaps, inconsistencies, and ambiguities in the documentation. Moreover, VUSP not only checks for the presence of information but can also enhance its comprehension and relevance according to the role performed, promoting communication among team functions. This aspect was reflected in the improvements reported by the teams in their documentation, as described in Section 5.6.

For the software industry, the checklist assists in evaluating User Stories and prototypes, identifying gaps, and aligning individual competencies with system objectives, contributing to product quality. In academia, VUSP can serve as a support tool for instructors, helping students both to inspect requirement artifacts and to create more complete requirement documents, ensuring that project teams have the necessary information. As next steps, the plan is first to implement automation of the checklist using artificial intelligence, enabling more efficient and consistent inspections. Furthermore, the aim is to expand the sample to include more industry professionals, integrate real-world contexts into evaluations, and add customization options. With an iterative approach and continuous user feedback, VUSP can evolve into a more robust tool, strengthening the connection between academia and industry in the field of requirements engineering.

ARTIFACTS AVAILABILITY

Supplementary material or artifacts associated with this research as well as interview results can be accessed through the link below: [22] [23]

ACKNOWLEDGMENTS

We would like to thank the financial support granted by Project No 017/2024 Divulga CT&I FAPEAM; Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES-PROEX) - Finance Code 001; FAPEAM - through the POSGRAD project 2025/2026;

REFERENCES

[1] Alaa Abdalazeim and Farid Meziane. 2021. A review of the generation of requirements specification in natural language using objects uml models and domain ontology. *Procedia Computer Science*, 189, 328–334. AI in Computational Linguistics. doi:https://doi.org/10.1016/j.procs.2021.05.102.
[2] Anis R. Amna and Geert Poels. 2022. Ambiguity in user stories: a systematic literature review. *Information and Software Technology*, 145, 106824. doi:https://doi.org/10.1016/j.infsof.2022.106824.
[3] Anis R. Amna, Yves Wautelet, Stephan Poelmans, Samedi Heng, and Geert Poels. 2025. The ambitrus framework for identifying potential ambiguity in user stories. *Journal of Systems and Software*, 223, 112357. doi:https://doi.org/10.1016/j.jss.2025.112357.
[4] B. Anda, K. Hansen, and G. Sand. 2009. An investigation of use case quality in a large safety-critical software development project. *Information and Software Technology*, 51, 12, 1699–1711. doi:10.1016/j.infsof.2009.04.005.
[5] Kent Beck and Martin Fowler. 2000. *Planning Extreme Programming*. (1st ed.). Addison-Wesley Longman Publishing Co., Inc., USA. ISBN: 0201710919.
[6] R. Beg, Q. Abbas, and A. Joshi. 2008. A method to deal with the type of lexical ambiguity in a software requirement specification document. In *2008 First International Conference on Emerging Trends in Engineering and Technology*. IEEE, Nagpur, India, 1212–1215. doi:10.1109/ICETET.2008.160.
[7] Elizabeth Bjarnason and Helen Sharp. 2017. The role of distances in requirements communication: a case study. *Requirements Engineering*, 22, 1, 1–26. doi:10.1007/s00766-015-0233-3.
[8] Giuliano Casale et al. 2016. Current and future challenges of software engineering for services and applications. *Procedia Computer Science*, 97, 34–42.
[9] Y. Chernak. 1996. A statistical approach to the inspection checklist formal synthesis and improvement. *IEEE Transactions on Software Engineering*, 22, 12, 866–874. doi:10.1109/32.553635.
[10] Raffaele Fabio Ciriello, Alexander Richter, Gerhard Schwabe, and Lars Mathiassen. 2023. The multiplexity of diagrams and prototypes in requirements development. *Information and Organization*, 33, 3, 100476. doi:https://doi.org/10.1016/j.infoandorg.2023.100476.
[11] Mike Cohn. 2004. *User Stories Applied: For Agile Software Development*. Addison-Wesley, Boston, 4.
[12] Gretchen T. De Macedo, Awdren Fontão, and Bruno Gadelha. 2023. Uiprotocheck: a checklist for semantic inspection of user interface prototypes. In *Proceedings of the XXXVII Brazilian Symposium on Software Engineering* (SBES '23). Association for Computing Machinery, Campo Grande, Brazil, 485–490. ISBN: 9798400707872. doi:10.1145/3613372.3613376.
[13] J. Doe. 2011. Recommended practice for software requirements specifications (ieee). https://www.midori-global.com/downloads/jpdf/jira-software-requirement-specification.pdf. Accessed: 2021-03-11. (2011).
[14] Clara Fonseca, Luciana Zaina, and Anna Marques. 2024. Avaliação de métodos para elicitação e especificação de requisitos de usabilidade com histórias de usuário: um experimento controlado. In *Anais do XXXVIII Simpósio Brasileiro de Engenharia de Software*. SBC, Curitiba/PR, 257–268. doi:10.5753/sbes.2024.3413.
[15] Richard P. Bagozzi Fred D. Davis and Paul R. Warshaw. 1998. User acceptance of computer technology: a comparison of two theoretical models. *Management Science, Volume 35, Issue 8*.
[16] Samuel A. Fricker, Kurt Schneider, Farnaz Fotrousi, and Christoph Thuemmler. 2016. Workshop videos for requirements communication. *Requirements Engineering*, 21, 4, 521–552. doi:10.1007/s00766-015-0231-5.
[17] S. Jaiswal and S. Venkataraman. 2025. Development and validation of a framework of metrics to evaluate prototypes. In *Design, User Experience, and Usability.*

HCII 2025. Lecture Notes in Computer Science. Vol. 15795. M. Schrepp, (Ed.) Springer, Cham. doi:10.1007/978-3-031-93224-3_6.

[18] Ron Jeffries. 2001. Essential XP: card, conversation, and confirmation. *XP Magazine*, 30.

[19] J. Jia, X. Yang, R. Zhang, and X. Liu. 2019. Understanding software developers' cognition in agile requirements engineering. *Science of Computer Programming*, 178, 1–19. doi:10.1016/j.scico.2019.03.005.

[20] Oliver Laitenberger, Colin Atkinson, Maud Schlich, and Khaled El Emam. 2000. An experimental comparison of reading techniques for defect detection in uml design documents. *Journal of Systems and Software*, 53, 2, 183–204. doi:https://doi.org/10.1016/S0164-1212(00)00052-2.

[21] Nayane Maia, Ana Oran, and Bruno Gadelha. 2025. Sweetcomp: a framework for software testing competency assessment. In *Proceedings of the 27th International Conference on Enterprise Information Systems - Volume 2: ICEIS*. INSTICC. SciTePress, 185–192. ISBN: 978-989-758-749-8. doi:10.5220/0013276600003929.

[22] Thiago Martins, Ana Carolina Oran, and Lucas Araujo. 2025. Evaluantion and verification of requirements artifacrts. a checklist applied to user stories and prototypes - artifacts of study. (Oct. 2025). doi:10.6084/m9.figshare.30347326.

[23] Thiago Martins, Ana Carolina Oran, and Lucas Araujo. 2025. Vusp dataset. (Oct. 2025). doi:10.6084/m9.figshare.30333919.

[24] Juliana Medeiros, Alexandre Vasconcelos, Carla Silva, and Miguel Goulão. 2018. Quality of software requirements specification in agile projects: a cross-case analysis of six companies. *Journal of Systems and Software*, 142, 171–194. doi:https://doi.org/10.1016/j.jss.2018.04.064.

[25] Jakub Mikołaj Mund. 2017. *Measurement-based quality assessment of requirements specifications for software-intensive systems*. Ph.D. Dissertation. Technische Universität München.

[26] Ana Carolina Oran, Gleison Santos, Bruno Gadelha, and Tayana Conte. 2021. A framework for evaluating and improving requirements specifications based on the developers and testers perspective. *Requirements Engineering*, 26, 4, 481–508.

[27] Jennifer Preece, Helen Sharp, and Yvonne Rogers. 2015. *Interaction Design: Beyond Human-Computer Interaction*. (4th ed.). Wiley, Hoboken.

[28] M.R. Raja Ramesh and Ch. Satyananda Reddy. 2021. Metrics for software requirements specification quality quantification. *Computers & Electrical Engineering*, 96, 107445. doi:https://doi.org/10.1016/j.compeleceng.2021.107445.

[29] G. Reggio, M. Leotta, F. Ricca, and D. Clerissi. 2018. Dusm: a method for requirements specification and refinement based on disciplined use cases and screen mockups. *Journal of Computer Science and Technology*, 33, 5, 918–939. doi:10.1007/s11390-018-1866-8.

[30] F. Ricca, G. Scanniello, M. Torchiano, G. Reggio, and E. Astesiano. 2014. Assessing the effect of screen mockups on the comprehension of functional requirements. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24, 1–38. doi:10.1145/2629457.

[31] Henrique F. Soares, Nicolli S.R. Alves, Thiago S. Mendes, Manoel Mendonça, and Rodrigo O. Spínola. 2015. Investigating the link between user stories and documentation debt on software projects. In *2015 12th International Conference on Information Technology - New Generations*, 385–390. doi:10.1109/ITNG.2015.68.

[32] Ian Sommerville. 2015. *Software Engineering*. (10th ed.). "A way to make systems do what we want is through requirements". Pearson, 120.

[33] Keith S. Taber. 2018. The use of cronbach's alpha when developing and reporting research instruments in science education. *Research in Science Education*, 48, 6, 1273–1296. doi:10.1007/s11165-016-9602-2.

[34] Thomas Thelin, Per Runeson, Claes Wohlin, Thomas Olsson, and Carina Andersson. 2004. Evaluation of usage-based reading—conclusions after three experiments. *Empirical Software Engineering*, 9, 1, 77–110. doi:10.1023/B:EMSE.0000013515.86806.d4.

[35] Karl Wiegers and Joy Beatty. 2013. *Software Requirements*. (3rd ed.). Microsoft Press, Redmond.