Requirements Documentation Containing Natural Language: A Systematic Tertiary Literature Review

Larissa Pereira Gonçalves[0000−0002−3355−9527], Edna Dias Canedo[0000−0002−2159−339X], Daniel Alves da Silva[0000−0001−7617−6609], Carlos Eduardo Lacerda Veiga[0000−0002−2677−1458], Rafael Timóteo de Sousa Junior[0000−0003−1101−3029], and Fábio Lúcio Lopes de Mendonça[1][0000−0001−7100−7304]

University of Brasilia (UnB), Brasilia – DF, Brazil
larissa.goncalves@redes.unb.br, ednacanedo@unb.br, daniel.alves@redes.unb.br, carlos.veiga@agu.gov.br,desousa@unb.br, fabio.mendonca@redes.unb.br

Abstract. Context: Requirements documentation in natural language has diverse artifacts, but few studies address their suitability to types of requirements or ease of communication. Methods: We conducted a systematic tertiary literature review (STLR) and identified 22 relevant review papers that address natural language artifacts used by practitioners to document software requirements. We also investigated which types of requirements are addressed by artifacts and if there are guidelines for each. Results: A variety of artifacts used for this purpose were identified, of which the most referenced in the literature were diagrams, use cases, conceptual models, user stories, and prototypes. The analysis highlighted that artifacts are applied differently to functional and non-functional requirements. In general, diagrams, use cases, scenarios, and prototypes can be used for both types of requirements, depending on the content (usability, security, etc.). However, user stories and derived artifacts are more recommended for functional requirements and have limitations for non-functional requirements. Conclusion: Furthermore, the study explored different guidelines, structures, and formats used in documentation artifacts, reflecting the diversity in requirements documentation practices in software projects.

Keywords: Requirements documentation · Specification · Software requirements · Natural language artifacts · Systematic Review.

1 Introduction

Growing software complexity presents a significant challenge for stakeholders and software professionals who must find an equilibrium between cost, time, and quality, while continuously innovating to meet customer expectations [7].

For instance, less documentation can impact future changes and maintenance, as it helps stakeholders to use, understand and evolve a system [1]. The

Proceedings of the 27th Workshop on Requirements Engineering (WER24), August 07-09, 2024, Buenos Aires, Argentina.
DOI: 10.29327/1407529.27-20

term "software documentation" refers to the set of information that describes the functionalities, architecture, and use of a software system. In traditional approaches, all or most of the documentation is concentrated in a single document, while agile methodologies adopt documentation as a collection of smaller artifacts, making requirements easier to read [26].

During documentation phase, some requirements (mainly non-functional) can be under-specified or undocumented. Inefficient management of requirements changes, especially when documentation is not clear, may result in project failure or schedule and scope deviation [4]. A lack of standards across documents can also happen due to a large set of available templates for recording requirements.

Therefore, the goal of this study is to capture the current state of the art in academic literature regarding artifacts for software requirements documentation. We investigate what are the most used artifacts containing natural language and their suitability to each type of requirement (functional or non-functional). To reach this goal, we conducted a Systematic Literature Review (SLR) and identified 22 relevant papers.

The main contributions of this paper are the understanding of patterns in software requirement specification (SRS) artifacts chosen in literature and their suitability regarding functional and non-functional requirements. The paper is organized as follows: Section 2 presents the methodology for SLR, followed by Section 3 which describes its results. Section 4's discussion delves into important topics about the results and section 5 describes our limitations and possible threats to validity. Our conclusions on the study are described in section 6.

2 Methodology

Our Tertiary Systemic Literature Review (TSLR) followed the Kitchenham and Charters protocol [17], in order to answer how are software requirements documented by practitioners in a way that provides a common understanding of the software and allows collaboration between different actors. We divided our work in two research questions:

- **RQ.1**What are the artifacts used in the literature for documenting requirements in natural language?
- **RQ.2** How are different documentation artifacts used to address functional and non-functional requirements in software projects?

Five databases were elected to perform the research, following Kitchenham's guidelines: [18]: ACM, IEEEXplore, SCOPUS, Springer Link and Web of Science. PICOC framework [35] was applied to choose the search string (Supplementary Material available at https://zenodo.org/records/11217791, file ´´SLR Requirements Documentation - PICOC terms.pdf").

Search string:
(“Requirements") AND (“specification" OR “documentation" OR “natural language") AND (“artifact" OR “model" OR “guidelines" OR

"format") AND ("systematic review" OR "review" OR "systematic mapping").

Inclusion and exclusion criteria were defined according to Kitchenham's guidelines [16]. Inclusion criteria addressed the main theme of studies, which followed our work's purpose and generated artifacts: (IC 1) studies must address artifacts, models, guides, or formats related to documenting requirements in natural language; (IC 2) studies must present at least one type of requirements documentation; (IC 3) studies must present information about existing documentation artifacts and their applications to illustrate functional and non-functional requirements.

Exclusion criteria (EC) were : (EC 1) studies that are not exclusively related to requirements documentation or do not directly address the mentioned research questions; (EC 2) Studies that are not written in a language other than that understood by the authors (Portuguese, Spanish and English); (EC 3) Studies that are not a literature review (e.g., book chapters, research studies, case studies, dissertations, theses, works in progress, position papers, duplicated works); (EC 4) Published before 2013; (EC 5) The focus of the study is not the analysis of natural language documentation artifacts, but other procedures (i.e.: automation, classification with NLP, application of machine learning, elicitation techniques, etc.); (EC 6) The main subject of the study is not related to software engineering or related areas, but other areas of research.

Despite previous filters already applied by exclusion criteria, it was necessary to check the quality of the selected studies, with the following quality assessment: (QA 1) Were the research questions clearly defined? (QA 2) Is the methodology clear, adequate, and replicable for the search, selection, and analysis of included studies? (QA 3) Were the limitations of the review highlighted and explained? (QA 4) Do the studies consider the practical applicability of the results and their implications for the research area, providing useful and relevant recommendations? (QA 5) Do the studies make an original contribution to knowledge in the area, whether by identifying gaps in the literature, synthesizing conflicting results, or proposing new perspectives or research directions?

Each selected study was evaluated with the 5 questions, and we included in our analysis only studies that met all proposed criteria.

To answer the proposed questions, data extraction comprised data relating to the publication and data related to the research questions (Supplementary Material available at https://zenodo.org/records/11217791, file " Table-Tertiary SLR on requirements documentation.xlsx").

2.1 SLR Conducting

To carry out this review, Parsifal[1] was the main tool used. Figure 1 illustrates the studies that remained after each conduction step.

The study collection period through the search string was until December 2023, resulting in a total of 574 papers (3 from ACM, 150 from IEEEXplore,

[1] https://parsif.al

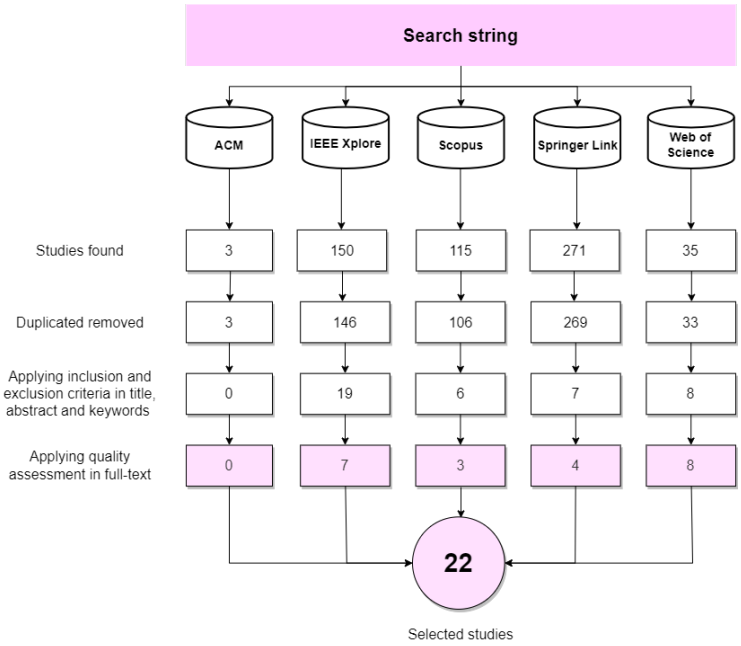

Fig. 1. Remaining studies after each SLR step.

35 from Web of Science, 115 from SCOPUS and 271 from Springer Link), as seen in Figure 1. 17 duplicated papers were removed, and 557 remained, (3 from ACM, 143 from IEEEXplore, 33 from Web of Science, 106 from SCOPUS and 269 from Springer Link) to apply inclusion and exclusion criteria on title and abstract. Although the studies analyzed addressed the requirements topic, 517 were removed due to non-compliance with one or more criteria. Finally, the full text was read for more in-depth analysis and application of the quality assessment of 40 studies (zero from ACM, 19 from IEEEXplore, 8 from Web of Science, 6 from SCOPUS and 7 from Springer Link). 17 papers were removed and 22 papers remained (zero from ACM, 7 from IEEE, 8 from Web of Science, 3 from SCOPUS and 4 from Springer Link).

3 SLR Results

3.1 RQ.1 - What are the artifacts used in the literature for the documentation of natural language requirements?

In the selected studies, 12 types of artifacts used in the documentation of requirements and present in more than one study were identified 1. Among these, UML diagrams, use cases, scenarios, conceptual models, user stories, and prototypes were, respectively, the most recurrent artifacts 2. Although prototypes

are more common during elicitation phase, they can be used for documentation as well [24].

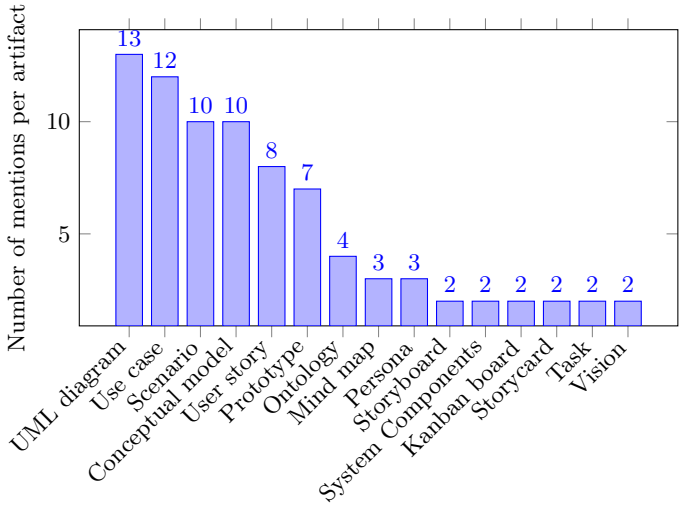

Fig. 2. Number of mentions of each documentation artifact in the selected studies.

Following the sequence, ontologies, mind maps, storyboards, personas, system components, kanban boards, storycards, tasks, and visions were cited in more than one paper. Although they are not cited as much as the first six, they may indicate different documentation in certain contexts. It is important to note that there are many types of ontologies, but we are considering those presented in natural language [8].

Artifacts mentioned only once were not included in the graphic, but are mentioned as follows: tables, formal specification, HTML reports, figures, storyboard tasks, videos, documented user experiences, wall, pin board, event-driven process chain (Event-driven Process Chain - EPC), domain models, tags, user wish list, user journey, product backlog, roadmap, the definition of "done", sprint burndown chart, and generic terms such as "product requirement", "market requirement" and "technical requirement" (Supplementary material available at https://zenodo.org/records/11217791, file " Table-Tertiary SLR on requirements documentation.xlsx"). Following Schon et. al (2017), we will highlight artifacts that are present in at least 20% of the studies [38]. For the percentage, we consider the number of mentions divided per 22 (total number of studies).

Diagrams are organized visual representations, used to represent various situations and relationships at a sequential, hierarchical, or structural level [20]. The unified modeling language (UML) allows the creation of several instances, such as class diagrams, use case diagrams, activity diagrams, and sequence diagrams [20]. The BPMN notation is considered the standard in business process

modeling [9]. The presence of a unified language (UML) and the possibility of customization even creating new extensions [19] - such as security in IoT [11] - are characterized as the reasons why this artifact was the most cited in the literature found (59.1%).

In our findings, use cases were mentioned almost as much as diagrams (54.5 %). Use cases are widely used in documenting software requirements and are adaptable, enabling the inclusion of various behavior flows [39]. Despite the risk of presenting too much information and the effort involved in describing many scenarios [25], they are considered more technical artifacts than user stories [38], with the possibility of reusing the documented requirements for generating manuals and effort estimate [29].

Scenarios and conceptual models were mentioned in 45.4% of the studies. Scenarios describe dynamic and context-dependent interactions between the system and external actors: user, network, and other devices. They are quite significant for checking if all relevant cases are covered [14], in addition to facilitating test automation [14]. Conceptual models are used as a bridge between stakeholders or users and the technical team [40]. Next, user stories appear in 36.3 % of studies evaluated. However, they are described as the most used artifact for agile software development in one of the studies [38], and not in another [41]. User stories are artifacts that document needs from the users' point of view, written in natural language and generally following a semi-structured [22] format, such as "I [user type], I want [something] for [goal]" [14].

In contrast to textual artifacts, prototypes are also widely used in requirements documentation (31.8 %), mainly related to visual aspects. However, there are different definitions of the prototype in the literature [38], covering both informal paper drawings (low fidelity), wireframes (medium to high fidelity) and also interactive HTML mockups [29]. Lower fidelity prototypes were recommended when the purpose is communication between stakeholders and agile teams [23], as well as discussions with users [32].

3.2 RQ.2 - How are different documentation artifacts used to address functional and non-functional requirements in software projects?

Documentation artifacts in software development have characteristics that enhance their suitness for recording and tracking certain types of requirements. In this way, Table 1 shows which types of requirements are illustrated by the artifacts found in this tertiary review, according to the selected studies.

Artifacts (%)	Type of requirement addressed	Multimedia classification	References
UML Diagram (59,1%)	Non reported.	Hybrid	[3,36,6,21,12,38,40,2,10,27,28,33,5].
Use case (54,5%)	FRs e NFRs.	Textual	[25,14,21,12,38,40,2,27,28,33,39,41].
Scenario (45,4%)	FRs e NFRs.	Textual	[3,25,14,30,27,28,33,39,5,38,10].
Conceptual model (45,4%)	FRs e NFRs.	Hybrid	[6,3,40,34,38,2,10,27,41,12].
User story (36,3%)	Recommended for FRs, can be adapted for NFRs	Textual	[22,38,3,25,15,14,27,41].
Prototype (31,8%)	FR, NFR.	Visual	[22,3,15,38,33,41,25]
Ontology (18,1%)	FRs e NFRs.	Hybrid	[3,14,10,5].
Mind map (13,6%)	FR e NFR.	Hybrid	[15,38,2].
Storycard (13,6%)	Recommended for FRs, can be used for NFRs.	Textual	[25,38,41].
Persona (13,6%)	NFR.	Visual	[38,3,41].
Storyboard (13,6%)	FRs.	Hybrid	[38,22].
System components (9%)	NFR.	Textual	[25,12].
Kanban board (9%)	Non reported.	Textual	[38,37].
Tasks (9%)	FR e NFR.	Textual	[38,41].
Vision (4,5%)	FR e NFR.	Textual	[38].

Table 1: Artifacts and documentation type found.

In general, we observed that artifacts with similar characteristics tend to be used in a similar way when approaching functional and non-functional requirements 1. In this way, we divided the artifacts into 3 categories in relation to the predominance of multimedia elements they present: Hybrid format, predominantly textual, and predominantly visual.

In general, Hybrid artifacts incorporate small texts, graphic elements, and/or images, often showcasing sequences of events or hierarchies with a higher level of abstraction. Predominantly textual artifacts consist of structured, semi-structured, or unstructured texts written in natural language. Lastly, predominantly visual

artifacts typically pertain to predicting interfaces and behaviors that will be viewed by one or more groups of users.

4 Discussion

The most cited artifacts found in our study partially corroborate the previous pape by Schon et. al [38]. In fact, user stories, prototypes, use cases and scenarios remain the most discussed artifacts in the literature. However, diagrams and conceptual models did not predominate as strongly as ours (only 11%). In part, this can be explained by the year of publication of the works, since most of the reviews found in our approach that mentioned diagrams are after 2017 [3,6,21,12,40,2,10,27,28,5], year of publication of the previous work [38]. Another possible point of divergence is in relation to the scope, since the work of Schon et. al [38] performs an explicit filter for agile documentation, and our work does not have this segmentation.

Regarding the least cited artifacts, it is interesting to highlight some contradictory points. Firstly, tasks, storyboards, and story cards are most used by agile teams during task distribution and schedule execution, to the detriment of requirements documentation. However, the artifacts were included because, in the studies evaluated, they were actively used in documentation. The Kanban board, normally used to divide and visualize tasks, was presented as a way of recording and tracking requirements [38], as well as tasks, storyboards, and story cards [38]. Some artifacts were also mentioned as complementary to "main" artifacts, such as personas [38,3] and story cards [25]. As pointed by Schon et. al [38], it is important to highlight that the literature may not reflect the totality of the most used artifacts, since some classic artifacts (such as product backlog) are rarely mentioned in [38] studies.

Hybrid artifacts:

Due to their higher level of abstraction, hybrid artifacts can address both functional and non-functional requirements. However, abstract models can have two conflicting roles: they must be general enough to facilitate communication between different actors with varied knowledge but also have enough formality to contain all the necessary information [40]. Even so, diagrams, conceptual models, and mind maps can be used to document functional and non-functional requirements [30,27]. Mornie et. al [27] list the use of class and sequence diagrams as facilitators for the software team's decision-making regarding the best way to implement the functional requirements requested by stakeholders [27].

Nguyen et. al [30] bring together examples of different types of diagrams for modeling static and dynamic aspects in a system, with a focus on non-functional security requirements [30]. In contrast, Mustafa et. al [28] explained that UML diagrams are extensively used to address functional requirements, but are not suitable for use as input material for generating non-functional test cases.

Predominantly textual artifacts:

Predominantly textual artifacts can be classified as those that mainly use natural language to record and document software requirements. They are quite

variable among themselves on level of detail and types of information presented. Scenarios, for example, are short context-dependent descriptions of interactions. They can be described in the textual form [14], as alternative or main scenarios in use cases [39] or represented as diagrams [30]. They are commonly used to illustrate and record functional requirements raised during the elicitation stage, but they can be used to represent usability [32] or security requirements.

Regarding User stories, there is a certain consensus that they serve to record functional requirements from the user's point of view, and are incomplete or inadequate to describe technical or very complex aspects [25,38,27,22]. Despite this, they are artifacts widely used in software requirements documentation.

The selected literature suggests extending the user story format to include more detail, hierarchy, and tracking, as well as combining HUs with additional requirements documentation [15,27,3]. The emergence of epics, tasks, story cards, and storyboards stems from the need for segmentation, tracking, and/or inclusion of other information - story cards, for example, allow you to capture time estimation, development accountability and goals, prioritization, and even requirements needed for the next elements from customer interactions [38]. Likewise, dividing requirements into smaller units makes reading and understanding easier, but can compromise the unity of documentation [25,38].

In turn, use cases are considered more complete user stories from a technical point of view [38] and existing use case templates focus on points of variability between scenarios and their traceability [39]. Use cases can be used to address functional [13] and non-functional [31] requirements. However, the complexity supported by use cases was also presented as a negative point by Medeiros et. al [25]. Ontologies are used to mitigate ambiguity problems in [3] requirements and can address functional and non-functional requirements. Bellendorf et. al [5] used ontologies for interoperability requirements since the same term can have different uses within the context of each type of system [5]. In another study, ontologies are used as additional information for generating test cases [10]. Finally, the vision artifact, despite being mentioned in two studies, was not attributed to the approach to functional or non-functional requirements.

Predominantly visual artifacts:

Visual artifacts are associated with representation of software interface elements that will be viewed by end users. They are closely related to functional requirements due to the proximity of interaction with the end user, however, they can be used to address non-functional aspects. Prototypes can elucidate functional flows requested by stakeholders and users, as well as assist in specifying usability requirements [38]. Ogata et. al suggests combining functional prototypes with UML diagrams to address functional and non-functional requirements more completely [33]. Personas, in general, are mentioned as artifacts that mainly address usability requirements [38,41].

In general, a notable trend in the analysis of the studies is the attention given to non-functional security requirements [34,39,30,25,38,10,28,5,3], usability [15,39,30,25,38,5] and performance [34,25,30,2,5]. One explanation is the increasing complexity of software systems and the threat of cyber attacks that violate

personal and sensitive data, demanding increasingly greater security requirements. Additionally, modern users have higher expectations regarding usability and performance. The regulations and standards to which companies are subject also reinforce the adoption of preventive and protective measures.

5 Limitation and threats to validity

Despite the solid methodology, we still found some limitations while conducting our study. As an example, even though we set time constraints, some of the cited studies can be outdated or not reflect reality, and therefore, might not capture the full picture of artifacts in requirements engineering. It's possible too that literature itself does not represent the current practices of software engineering. Also, different types of software can also deviate from the pattern found here, due to specific needs, such as IOT devices or chatbots, for example.

Regarding the quality of the studies, during the selection phase, we applied extensive inclusion/exclusion criteria and quality assessment measures, following similar systematic reviews of the software area. However, there is the possibility that the filters were too restrictive or too permissive. We also were able to highlight patterns and classify artifact representation dominance, but since this is the first work to address the theme, to our knowledge, it is interesting to have future works trying to replicate the results found here and see if they are maintained.

6 Conclusion

Our study presented the findings from our Tertiary SLR about requirements documentation artifacts and their characteristics. In the selected papers, we found that UML diagrams, use cases, scenarios, conceptual models, user stories, and prototypes were present in many studies. We were also able to identify patterns in the artifacts' representation predominance (hybrid, visual, or textual) and address the types of requirements each artifact is suited to represent. In general, predominantly visual artifacts are mostly used to mimicry a software expected behavior regarding design and usability, because they are actually shown to the user. Textual requirements are more suitable for non-visible or non-perceptive features and they can be associated with complex requirements, that require longer explanations. Hybrid artifacts (visual and textual) are the most versatile and can be used for a wide range of situations. It is common that more than one artifact is used for SRS. Future studies should try to reproduce this methodology see if results still apply, and investigate the most used artifacts for each software context, to see if distinct software characteristics can influence on SRS artifact choice.

7 Acknowledgments

The authors would like to thank the technical and computational support from the LATITUDE Laboratory, at the University of Brasília, to TED 01/2019 from the "Advocacia Geral da União - AGU" (AGU 697.935/2019), to TED 01/2021 from "Secretaria Nacional de Assistência Social" – SNAS/DGSUAS /CGRS to the SISTER City Project – Secure and Real Time Smart Systems for Smart Cities (Grant 625/2022), to the Project "Project Control and Unification System for the Federal District Government – Sispro-DF" (Grant 497/2023), to the project "Methodology to Support the Elicitation of Ethical and Privacy Requirements" (Grant e 514/2023) Deanship of Research and Innovation – DPI/UnB and FAP/DF.

References

1. Aghajani, E., Nagy, C., Linares-Vásquez, M., Moreno, L., Bavota, G., Lanza, M., Shepherd, D.C.: Software documentation: the practitioners' perspective. In: Rothermel, G., Bae, D. (eds.) ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020. pp. 590–601. ACM (2020). https://doi.org/10.1145/3377811.3380405
2. Amjad, A., Azam, F., Anwar, M.W., Butt, W.H., Rashid, M.: Event-driven process chain for modeling and verification of business requirements-a systematic literature review. IEEE Access **6**, 9027–9048 (2018). https://doi.org/10.1109/ACCESS.2018.2791666, https://doi.org/10.1109/ACCESS.2018.2791666
3. Amna, A.R., Poels, G.: Systematic literature mapping of user story research. IEEE Access **10**, 51723–51746 (2022). https://doi.org/10.1109/ACCESS.2022.3173745, https://doi.org/10.1109/ACCESS.2022.3173745
4. Behutiye, W., Seppänen, P., Rodríguez, P., Oivo, M.: Documentation of quality requirements in agile software development. In: Li, J., Jaccheri, L., Dingsøyr, T., Chitchyan, R. (eds.) EASE '20: Evaluation and Assessment in Software Engineering, Trondheim, Norway, April 15-17, 2020. pp. 250–259. ACM (2020), https://doi.org/10.1145/3383219.3383245
5. Bellendorf, J., Mann, Z.Á.: Specification of cloud topologies and orchestration using TOSCA: a survey. Computing **102**(8), 1793–1815 (2020). https://doi.org/10.1007/S00607-019-00750-3, https://doi.org/10.1007/s00607-019-00750-3
6. Bozyiğit, F., Özlem Aktaş, Kılınç, D.: Linking software requirements and conceptual models: A systematic literature review. Engineering Science and Technology, an International Journal **24**(1), 71–82 (2021). https://doi.org/https://doi.org/10.1016/j.jestch.2020.11.006, https://www.sciencedirect.com/science/article/pii/S2215098620342580
7. Brhel, M., Meth, H., Maedche, A., Werder, K.: Exploring principles of user-centered agile software development: A literature review. Inf. Softw. Technol. **61**, 163–181 (2015). https://doi.org/10.1016/J.INFSOF.2015.01.004
8. Castañeda, V., Ballejos, L.C., Caliusco, M.L.: Improving the quality of software requirements specifications with semantic web technologies. In: WER (2012)

9. Chergui, M.E., Benslimane, S.M.: A valid BPMN extension for supporting security requirements based on cyber security ontology. In: Abdelwahed, E.H., Bellatreche, L., Golfarelli, M., Méry, D., Ordonez, C. (eds.) Model and Data Engineering - 8th International Conference, MEDI 2018, Marrakesh, Morocco, October 24-26, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11163, pp. 219–232. Springer (2018), https://doi.org/10.1007/978-3-030-00856-7_14
10. Clark, A.G., Walkinshaw, N., Hierons, R.M.: Test case generation for agent-based models: A systematic literature review. Inf. Softw. Technol. **135**, 106567 (2021). https://doi.org/10.1016/J.INFSOF.2021.106567, https://doi.org/10.1016/j.infsof.2021.106567
11. Compagnucci, I., Corradini, F., Fornari, F., Polini, A., Re, B., Tiezzi, F.: A systematic literature review on iot-aware business process modeling views, requirements and notations. Softw. Syst. Model. **22**(3), 969–1004 (2023), https://doi.org/10.1007/s10270-022-01049-2
12. Duran, M.B., Mussbacher, G.: Reusability in goal modeling: A systematic literature review. Inf. Softw. Technol. **110**, 156–173 (2019). https://doi.org/10.1016/J.INFSOF.2019.03.004, https://doi.org/10.1016/j.infsof.2019.03.004
13. Farid, W.M.: The NORMAP methodology: Lightweight engineering of non-functional requirements for agile processes. In: Leung, K.R.P.H., Muenchaisri, P. (eds.) 19th Asia-Pacific Software Engineering Conference, APSEC 2012, Hong Kong, China, December 4-7, 2012. pp. 322–325. IEEE (2012), https://doi.org/10.1109/APSEC.2012.23
14. Farooq, M.S., Omer, U., Ramzan, A., Rasheed, M.A., Atal, Z.: Behavior driven development: A systematic literature review. IEEE Access **11**, 88008–88024 (2023). https://doi.org/10.1109/ACCESS.2023.3302356, https://doi.org/10.1109/ACCESS.2023.3302356
15. Heikkil¨a, V.T., Damian, D.E., Lassenius, C., Paasivaara, M.: A mapping study on requirements engineering in agile software development. In: 41st Euromicro Conference on Software Engineering and Advanced Applications, EUROMICRO-SEAA 2015, Madeira, Portugal, August 26-28, 2015. pp. 199–207. IEEE Computer Society (2015). https://doi.org/10.1109/SEAA.2015.70, https://doi.org/10.1109/SEAA.2015.70
16. Kitchenham, B.: Procedures for performing systematic reviews. Keele, UK, Keele University **33**(2004), 1–26 (2004)
17. Kitchenham, B.A.: Systematic review in software engineering: where we are and where we should be going. In: Proceedings of the 2nd international workshop on Evidential assessment of software technologies. pp. 1–2 (2012)
18. Kitchenham, B.A., Pretorius, R., Budgen, D., Brereton, P., Turner, M., Niazi, M., Linkman, S.G.: Systematic literature reviews in software engineering - A tertiary study. Inf. Softw. Technol. **52**(8), 792–805 (2010), https://doi.org/10.1016/j.infsof.2010.03.006
19. Kopp, O., Binz, T., Breitenbücher, U., Leymann, F.: BPMN4TOSCA: A domain-specific language to model management plans for composite applications. In: Mendling, J., Weidlich, M. (eds.) Business Process Model and Notation - 4th International Workshop, BPMN 2012, Vienna, Austria, September 12-13, 2012. Proceedings. Lecture Notes in Business Information Processing, vol. 125, pp. 38–52. Springer (2012), https://doi.org/10.1007/978-3-642-33155-8_4
20. Koç, H., Erdoğan, A.M., Barjakly, Y., Peker, S.: Uml diagrams in software engineering research: A systematic literature review. Proceedings **74**(1) (2021), https://www.mdpi.com/2504-3900/74/1/13

21. Kuhar, S., Polancic, G.: Conceptualization, measurement, and application of semantic transparency in visual notations. Softw. Syst. Model. **20**(6), 2155–2197 (2021). https://doi.org/10.1007/S10270-021-00888-9, https://doi.org/10.1007/s10270-021-00888-9
22. Kustiawan, Y.A., Lim, T.Y.: User stories in requirements elicitation: A systematic literature review. In: 8th IEEE International Conference On Software Engineering and Computer Systems, ICSECS 2023, Penang, Malaysia, August 25-27, 2023. pp. 211–216. IEEE (2023). https://doi.org/10.1109/ICSECS58457.2023.10256364, https://doi.org/10.1109/ICSECS58457.2023.10256364
23. Lucia, A.D., Qusef, A.: Requirements engineering in agile software development. J. Emerg. Technol. Web Intell. **2**(3) (Aug 2010)
24. Mannio, M., Nikula, U.: Requirements elicitation using a combination of prototypes and scenarios. In: WER. pp. 283–296 (2001)
25. Medeiros, J., Goulão, M., de Vasconcelos, A.M.L., Silva, C.T.L.L.: Towards a model about quality of software requirements specification in agile projects. In: Paulk, M.C., Machado, R.J., Brito, M.A., Goulão, M., Amaral, V. (eds.) 10th International Conference on the Quality of Information and Communications Technology, QUATIC 2016, Lisbon, Portugal, September 6-9, 2016. pp. 236–241. IEEE Computer Society (2016). https://doi.org/10.1109/QUATIC.2016.058, https://doi.ieeecomputersociety.org/10.1109/QUATIC.2016.058
26. Medeiros, J., de Vasconcelos, A.M.L., Silva, C., Goulão, M.: Requirements specification for developers in agile projects: Evaluation by two industrial case studies. Inf. Softw. Technol. **117** (2020), https://doi.org/10.1016/j.infsof.2019.106194
27. Mornie, M.N., Jali, N., Junaini, S.N., Mit, E., Shiang, C.W., Saee, S.: Visualisation of user stories in UML models: A systematic literature review. Acta Inform. Pragensia **12**(2), 419–438 (Oct 2023)
28. Mustafa, A., M. N. Wan-Kadir, W., Ibrahim, N., Arif Shah, M., Younas, M., Khan, A., Zareei, M., Alanazi, F.: Automated test case generation from requirements: A systematic literature review. Comput. Mater. Contin. **67**(2), 1819–1833 (2021)
29. Nawrocki, J.R., Ochodek, M., Jurkiewicz, J., Kopczynska, S., Alchimowicz, B.: Agile requirements engineering: A research perspective. In: Geffert, V., Preneel, B., Rovan, B., Stuller, J., Tjoa, A.M. (eds.) SOFSEM 2014: Theory and Practice of Computer Science - 40th International Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 26-29, 2014, Proceedings. Lecture Notes in Computer Science, vol. 8327, pp. 40–51. Springer (2014), https://doi.org/10.1007/978-3-319-04298-5_5
30. Nguyen, P.H., Kramer, M.E., Klein, J., Traon, Y.L.: An extensive systematic review on the model-driven development of secure systems. Inf. Softw. Technol. **68**, 62–81 (2015). https://doi.org/10.1016/J.INFSOF.2015.08.006, https://doi.org/10.1016/j.infsof.2015.08.006
31. Nguyen, Q.L.: Non-functional requirements analysis modeling for software product lines. In: ICSE Workshop on Modeling in Software Engineering, MiSE 2009, Vancouver, BC, Canada, May 17-18, 2009. pp. 56–61. IEEE Computer Society (2009), https://doi.org/10.1109/MISE.2009.5069898
32. Obendorf, H., Finck, M.: Scenario-based usability engineering techniques in agile development processes. In: Czerwinski, M., Lund, A.M., Tan, D.S. (eds.) Extended Abstracts Proceedings of the 2008 Conference on Human Factors in Computing Systems, CHI 2008, Florence, Italy, April 5-10, 2008. pp. 2159–2166. ACM (2008), https://doi.org/10.1145/1358628.1358649
33. Ogata, S., Matsuura, S.: A review method for UML requirements analysis model employing system-side prototyping. Springerplus **2**(1), 134 (Dec 2013)

34. Ouhbi, S., Idri, A., Alemán, J.L.F., Toval, A.: Software quality requirements: A systematic mapping study. In: Muenchaisri, P., Rothermel, G. (eds.) 20th Asia-Pacific Software Engineering Conference, APSEC 2013, Ratchathewi, Bangkok, Thailand, December 2-5, 2013 - Volume 1. pp. 231–238. IEEE Computer Society (2013). https://doi.org/10.1109/APSEC.2013.40, https://doi.org/10.1109/APSEC.2013.40
35. Petticrew, M., Roberts, H.: Systematic Reviews in the Social Sciences. Blackwell Publishing Ltd, Oxford, UK (2006). https://doi.org/10.1002/9780470754887
36. Rashid, M., Anwar, M.W., Azam, F., Kashif, M.: Model-based requirements and properties specifications trends for early design verification of embedded systems. In: 11th System of Systems Engineering Conference, SoSE 2016, Kongsberg, Norway, June 12-16, 2016. pp. 1–7. IEEE (2016). https://doi.org/10.1109/SYSOSE.2016.7542917, https://doi.org/10.1109/SYSOSE.2016.7542917
37. dos Santos, P.S.M., Caetano, A., de Souza, B.P., Travassos, G.H.: On the benefits and challenges of using kanban in software engineering: a structured synthesis study. J. Softw. Eng. Res. Dev. **6**, 13 (2018). https://doi.org/10.1186/S40411-018-0057-1, https://doi.org/10.1186/s40411-018-0057-1
38. Schon, E., Thomaschewski, J., Escalona, M.J.: Agile requirements engineering: A systematic literature review. Comput. Stand. Interfaces **49**, 79–91 (2017). https://doi.org/10.1016/J.CSI.2016.08.011, https://doi.org/10.1016/j.csi.2016.08.011
39. de Sousa Santos, I., Andrade, R.M.C., de Alcântara dos Santos Neto, P.: Templates for textual use cases of software product lines: results from a systematic mapping study and a controlled experiment. J. Softw. Eng. Res. Dev. **3**, 5 (2015). https://doi.org/10.1186/S40411-015-0020-3, https://doi.org/10.1186/s40411-015-0020-3
40. Verbruggen, C., Snoeck, M.: Practitioners' experiences with model-driven engineering: a meta-review. Softw. Syst. Model. **22**(1), 111–129 (2023). https://doi.org/10.1007/S10270-022-01020-1, https://doi.org/10.1007/s10270-022-01020-1
41. Wagenaar, G., Overbeek, S., Lucassen, G., Brinkkemper, S., Schneider, K.: Working software over comprehensive documentation - rationales of agile teams for artefacts usage. J. Softw. Eng. Res. Dev. **6**, 7 (2018). https://doi.org/10.1186/S40411-018-0051-7, https://doi.org/10.1186/s40411-018-0051-7