Information and Software Technology 145 (2022) 106824

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Ambiguity in user stories: A systematic literature review

Anis R. Amna [a,b,*], Geert Poels [a,c]

[a] Department of Business Informatics and Operations Management, Ghent University, Belgium
[b] Department of Informatics Engineering, University of 17 Agustus 1945 Surabaya, Indonesia
[c] FlandersMake@UGent – corelab CVAMO, Ghent, Belgium

ARTICLE INFO

Keywords:
Requirements engineering
Agile software development
User story
Ambiguity
Systematic Literature Review

ABSTRACT

Context: Ambiguity in user stories is a problem that has received little research attention. Due to the absence of review studies, it is not known how and to what extent this problem, which impacts the effectiveness of user stories in supporting systems development, has been solved.
Objectives: We review the studies that investigate or develop solutions for problems related to ambiguity in user stories. We investigate how these problems manifest themselves, what their causes and consequences are, what solutions have been proposed and what evidence of their effectiveness has been presented. Based on the insights we obtain from this review, we identify research gaps and suggest opportunities for future research.
Methods: We followed Systematic Literature Review guidelines to review problems investigated, solutions proposed, and validation/evaluation methods used. We classified the reviewed studies according to the four linguistic levels of ambiguity (i.e., lexical, syntactic, semantic, pragmatic) proposed by Berry and Kamsties to obtain insights from patterns that we observe in the classification of problems and solutions.
Results: A total of 36 studies published in 2001–2020 investigated ambiguity in user stories. Based on four patterns we discern, we identify three research gaps. First, we need more research on human behaviors and cognitive factors causing ambiguity. Second, ambiguity is seldom studied as a problem of a set of related user stories, like a theme or epic in Scrum. Third, there is a lack of holistic solution approaches that consider ambiguity at multiple linguistic levels.
Conclusion: Ambiguity in user stories is a known problem. However, a comprehensive solution for addressing ambiguity in a set of related user stories as it manifests itself at different linguistic levels as a cognitive problem is lacking.

1. Introduction

Agile Software Development (ASD) has proven its success in delivering high-quality software [1–3], supporting changes in requirements and priorities [4,5], accelerating software delivery [6–9], and better aligning IT solutions to business needs [10,11]. A distinctive feature of ASD is the continuous attention paid to requirements in project teams' practices like iterative planning, daily stand-up meetings, retrospectives, and iterative reviews [11–13]. Iterative planning enables Product Owners (PO) and developer teams to gradually refine their understanding of the initial requirements and discuss how to deliver a software product that meets those requirements. Progress is constantly monitored through daily stand-up meetings to discuss prioritization of the requirements and to assess whether meeting those requirements can be accomplished in time and within budget. Iterative reviews obtain feedback from the client on the software product in development and look back at what has been accomplished. Retrospective processes investigate what can be done better in the next iteration.

Those practices are facilitated by a communication instrument that refers to the requirements. Preferably, requirements are described using a language that is familiar to both POs and developers. In ASD, the user story technique is a widely used artifact to express commitments between the development team and (a type of) user [14,15]. User stories are written in a semi-structured natural language from a user's perspective on the required software system's functionality [15–17]. Throughout the project lifecycle, user stories help to share the understanding of the expected system goals and functions [4,16,18,19]. Further, user stories are beneficial for monitoring progress towards developing the required system functionality, identifying persisting problems, and improving customer satisfaction [20–24].

* Corresponding author.
E-mail addresses: AnisRahmawati.Amna@UGent.be, anis.r.amna@untag-sby.ac.id (A.R. Amna), Geert.Poels@UGent.be (G. Poels).

https://doi.org/10.1016/j.infsof.2022.106824
Received 9 April 2021; Received in revised form 22 December 2021; Accepted 3 January 2022
Available online 10 January 2022

Despite empirical support for these benefits (e.g., [18,19,25,26]), several studies showed that user stories are prone to ambiguity [24, 26-28]. Based on these studies, we describe ambiguity as *problems regarding the articulation of requirements as user stories, which cause doubtful, imprecise, and multiple interpretations of these requirements*. These problems typically originate in different ways of articulating requirements and differences in application domain knowledge.

Although ambiguity of requirements impacts the system development process and its outcomes, we are not aware of any secondary study that reviewed and synthesized the state-of-the-art of the academic research that addresses this problem specifically for user stories. Therefore, in this paper, we investigate how research has dealt with user story ambiguity and what the outcome of this research is. Based on this review, we assess whether any research gaps remain and whether they provide interesting research opportunities.

To achieve these objectives, we perform a literature review of the published research on ambiguity in user stories. Following the Systematic Literature Review methodology [29], we investigate the following research questions:

RQ1: How are ambiguity problems defined?

RQ2: What solutions are proposed for ambiguity?

RQ3: What evidence of the effectiveness of these solutions is available?

Using a linguistic classification of ambiguity issues, we contribute novel insights into the current knowledge of understanding, avoiding, and resolving ambiguity in user stories. Further, based on the review, we identify research opportunities to enhance the effectiveness of the user story technique in supporting Requirements Engineering (RE) activities in ASD projects.

This paper is divided into seven sections. This first section presents and motivates the goal of our review study. Section 2 presents a simple, fictitious example that illustrates what ambiguity in user stories could be and how it could arise, as background for our study. Section 3 reviews related work on ambiguity in requirements, not specifically confined to user stories. Section 4 describes our research methodology. Subsequently, Section 5 presents the results of our systematic literature review. Section 6 discusses the results, exhibits the limitations of the research, and formulates recommendations for future research. Finally, Section 7 concludes the paper.

2. Background

User story formulation is standardized using templates. The Connextra template [15] was rated as the most used template for user stories in the survey reported in [25]. The template is:

"*As a* ⟨*role*⟩, *I want* ⟨ *goal*⟩, *so that* <*benefit*>"

The template contains three segments: ⟨role⟩ describes a (type of) system user that wants the system to achieve or do something, ⟨goal⟩ describes the action to be performed by the system in support of the user, and ⟨benefit⟩ provides the rationale for this action from the point of view of the user.

To illustrate how ambiguity can exist in user stories, we use the example of a software application for administering user rights to a company's systems. The company developing the application is using the Scrum ASD methodology. One required feature of the application is to reenable access to the systems for registered users that have lost their credentials. An example user story related to this system feature is:

"As a system administrator, I want to generate a new password for a registered user, so that the user can login into the systems"

Suppose now that within the same collection of user stories (e.g., an epic or theme) that describe the required features of the application, also the following user story is found:

"As a manager responsible for user authorization, I want to reconfirm the systems' access for a registered user, so that the continued access to the systems is assured."

To continue our example, assume that both user stories are taken out of the Product Backlog and are added to the Sprint Backlog, meaning that development of the features described by these user stories is planned for the upcoming iteration in the project. When analyzing the user stories, the project team is faced with several questions:

- Is the manager responsible for user authorization also the system administrator? Or is being responsible for user authorization one of the responsibilities of the system administrator? Or are we talking about two different organizational roles here?
- Is generating a new password the same action as reconfirming systems' access? Or is generating a new password just part of reconfirming the systems' access? Or are we talking about two different actions here?
- Is assuring the continuation of access to the systems realized when the user can login again to the system? Or is more required than just being able to login again?

Without clear answers to these questions, it is hard to figure out whether these two user stories just duplicate required functionality or require different functionality for different types of system users. If the action (i.e., goal segment) of the first user story is included in the action of the second user story but the role of the second user story is part of the role of the first user story, then an obvious inconsistency between both user stories exists. If the system administrator is indeed the manager responsible for user authorization, then it can be questioned whether the goal of the first user story is sufficient to achieve the goal of the second user story. If yes, then why need the second user story? If no, then what else is needed to assure continued access to the systems? The vagueness of concepts used in the formulation of these user stories (e.g., continued systems' access? manager responsible for user authorization?) results in ambiguity, meaning multiple interpretations of these user stories. In turn, this potentially leads to other problems like duplication of required system functionality, the inconsistency of requirements, and requirements being incompletely described.

While the use of templates reduces user story writing variety, a user story remains by nature a verbal communication instrument as there are many ways to express an intended meaning. Although the ⟨role⟩, ⟨goal⟩, and ⟨benefit⟩ segments impose a fixed structure to the user story format, there are different ways to express a particular system feature when instantiating these segments in words and grammatical structures. The different formulations in words and grammatical structures for the different user story segments might be intentional (e.g., providing access to a user who has lost access requires more than just generating a new password) or unintentional. Without further clarification or knowledge of context, it is not easy to unambiguously interpret these two example user stories, which are clearly related or even dependent, even if the precise nature of their relationship is vague.

While the emphasis on informal communication using verbal language (as opposed to, for instance, formal or semi-formal requirements models) certainly provides advantages with respect to user involvement [24,27], it can also lead to ambiguity in what exactly is expressed by the user story, as illustrated by our example. Causes of such ambiguity could be differences in domain knowledge, differences in terminology used to articulate that knowledge, or just being careless or sloppy when formulating user stories. We are interested in finding out how ambiguity in user stories manifests itself and to what extent researchers have developed and tested solutions for this problem.

3. Related work

According to the Merriam-Webster dictionary, ambiguity exists when a word or expression can be understood in two or more possible ways [30]. In Software Engineering, ambiguity has been described as requirements that do not have a clear single meaning [28,31-33], but also as referring to inconsistency in requirements [34] and missing requirements [35]. Ambiguity is inherent in natural language, yet, the issue is considered a serious problem in software development because ambiguity in requirements might not always be easily recognized [36–38]. Chantree et al. [37] classified ambiguity as nocuous (i.e., ambiguity that leads to different interpretations) and innocuous (i.e., although potentially ambiguous, there is one obvious interpretation). They further classify nocuous ambiguity into easily recognized (i.e., acknowledged) and undetected (i.e., unacknowledged). Undetected ambiguity might thus persist during software development and is considered particularly problematic.

Given that situation, numerous studies have been conducted to detect and reduce ambiguity in natural language requirements. Studies focusing on detecting ambiguity generally applied means of automatic interpretation of natural language requirements (e.g., [24,28,37,39-41]) or the transformation of natural language requirements into models (e.g., [36,42-47]). Studies that aimed to reduce ambiguity in natural language requirements proposed grammatical restrictions (e.g., [26, 48-50]), the standardization of sentence structures (e.g., [32,51,52]), and models or ontologies supporting requirements formulation (e.g., [53–57]).

Several secondary studies reviewed research on ambiguity in requirements. These studies generally reviewed specific solution approaches. Dermeval et al. [58] examined the use of ontologies for supporting RE activities. Their review indicated that 56.72% of the reviewed studies applied ontologies to reduce ambiguity when specifying requirements. Kocerka et al. [59] and Alzayed and Al-Hunaiyyan [60] performed literature reviews on the use of NLP to improve requirements quality and to automatically detect ambiguity present in requirements. An interesting finding is that most of the reviewed studies aim at detecting ambiguity in single sentences rather than in fragments of text covering multiple requirements. The literature review by Bano [61], who listed different NLP techniques to address RE ambiguity, showed that 57% of the reviewed studies focused on syntactic ambiguity issues, while only 18% of studies addressed semantic ambiguity.

None of these review studies focused on requirements articulated as user stories. To the best of our knowledge, there are no reviews of the research on ambiguity in requirements that are described as user stories. Our systematic literature review addresses this knowledge gap.

4. Methodology

Our research process followed the Kitchenham guidelines [29] for Systematic Literature Review. We first searched for published research on user stories. Next, we selected those studies that focus on ambiguity in user stories.

In sub-Section 4.1, we explain how we systematically searched for published research on user stories. In sub-Section 4.2, we describe the classification of the identified studies in problem classes to select the studies addressing ambiguity in user stories. In sub-Section 4.3, we explain how we extracted the data for answering the review research questions (see Section 1). We also present the classification schema to make sense of the data and provide structure to this research area.

4.1. Searching for published research on user stories

The search strategy was designed by first defining our search space consisting of the following digital libraries: Web of Science, Scopus, Science Direct, Google Scholar, IEEE Xplore, Association for Computing Machinery (ACM) digital library, and Association for Information Systems (AIS) e-Library. The reason for selecting these digital libraries was pragmatic – they are either freely accessible, or our research institute provides access to them. Especially the inclusion of Google Scholar ensures that we cover with near certainty the entirety of the academic literature. On the other hand, it necessitates care in the selection of documents as Google Scholar also includes unpublished reports and other forms of ‘gray literature’. That is why we found it useful to include also digital libraries that mainly or exclusively contain journals, proceedings, and books for which peer review is assured. These other libraries might compensate for flaws in the search engine of Google Scholar or can be used to verify if a certain document found with Google Scholar was likely to be peer-reviewed.

Relevant sources were then searched using the search string "user story OR user stories", which was applied by the digital libraries’ search engines to the title, abstract or keywords of indexed documents, or any combination of these, depending on the search engine’s functionality. We limited our search to documents published since 2001, which is the year of publication of the Agile Manifesto. To further limit the search to the appropriate ASD context, we concatenated another search string with names or abbreviations of Agile methodologies that prescribe or suggest the use of user stories (or artifacts like user stories). After some studying, we found out that user stories play a role in documenting requirements in several ASD methodologies that are well known and widely used: *Scrum, Extreme Programming (XP), Scaled Agile Framework (SAFe), Behavior-Driven Development (BDD)* and *Feature-Driven Development (FDD)*. We explicitly included the BDD and FDD methods because they extend user stories (i.e., test scenarios in BDD) or offer an alternative to user stories (i.e., features in FDD), so it is plausible that papers reporting on research related to these ASD methodologies, also investigate the user story artifact. We also found out that *hybrid* ASD methodologies (e.g., Kanban and Scrum, RUP and XP) employ the user story technique, although not always in a primary role. The full search string we eventually used was:

(“user story” OR “user stories”) AND (“agile” OR “Scrum” OR “Extreme Programming” OR “XP” OR “Rational Unified Process” OR “RUP” OR “Feature-Driven Development” OR “Feature Driven Development” OR “FDD” OR “Behavior-Driven Development” OR “Behavior Driven Development” OR “BDD” OR “Hybrid” OR “Scrum/XP”)

Prior to the selection of this search string, different combinations of search terms were constructed using the PICOC method [29]. Those combinations were then tested as queries in the search engines of the selected digital libraries. The results were compared to select the search string that resulted in the most comprehensive number of studies returned.

We also deliberately added “agile” to the second term in the concatenation, to cover for documents that do not explicitly mention a particular ASD methodology in their title, abstract or keywords. As a drawback, we noticed that the search engines returned several documents that accidently mention the word “agile” and consider some concept of user story in a different context than ASD. We addressed this drawback through our choice of inclusion criteria that select papers based on relevance (confer infra).

Further, we noted too late that we made a mistake by including RUP instead of SAFe. Papers on RUP that accidentally mention user stories will not have been selected due to our inclusion criteria (confer infra). However, we risk missing out on papers that deal with user stories in the context of SAFe. We believe this problem to be minor as SAFe in full is Scaled Agile Framework, so contains the word “agile”, which is included in our search string.

Next, inclusion and exclusion criteria were defined for deciding which documents returned by the search engines were relevant (i.e., inclusion) and of sufficient quality (i.e., exclusion) to be further considered. Given the high number of documents returned (confer infra), these criteria were manually applied by the corresponding author only. We first applied the following *exclusion* criteria:

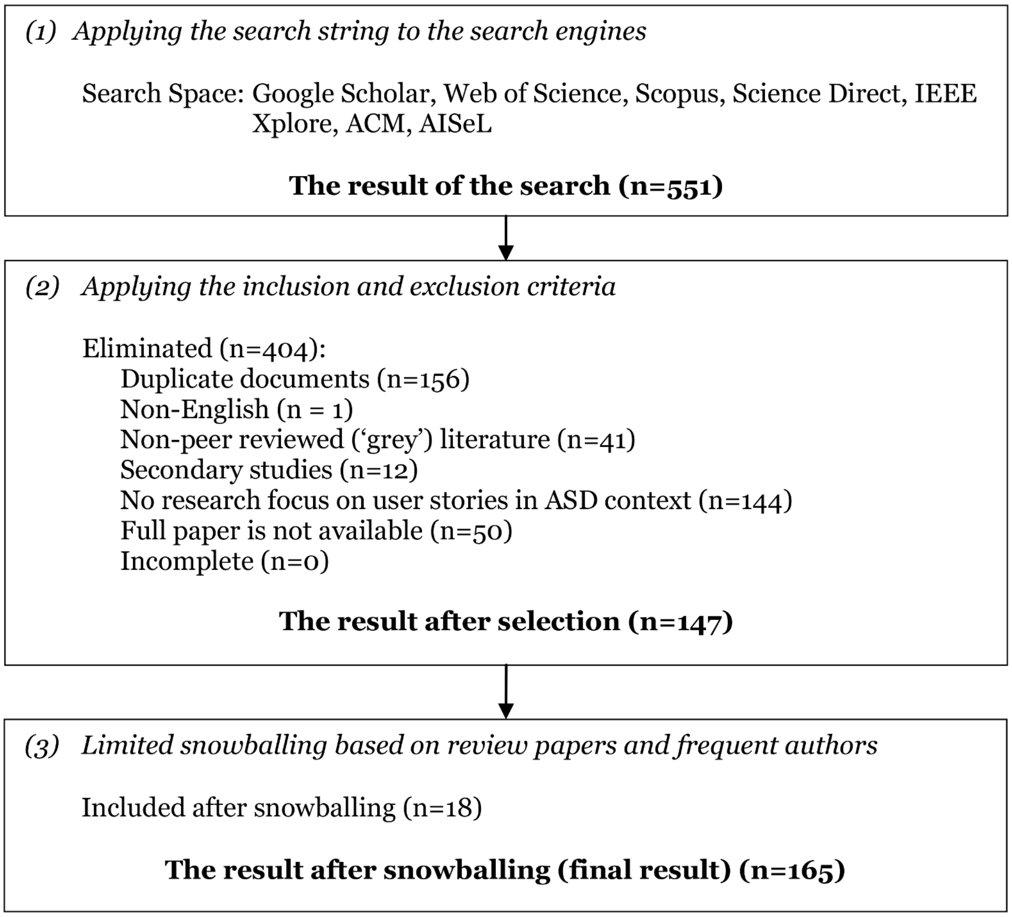
(1) Applying the search string to the search engines
Search Space: Google Scholar, Web of Science, Scopus, Science Direct, IEEE Xplore, ACM, AISeL
The result of the search (n=551)
(2) Applying the inclusion and exclusion criteria
Eliminated (n=404):
Duplicate documents (n=156)
Non-English (n = 1)
Non-peer reviewed ('grey') literature (n=41)
Secondary studies (n=12)
No research focus on user stories in ASD context (n=144)
Full paper is not available (n=50)
Incomplete (n=0)
The result after selection (n=147)
(3) Limited snowballing based on review papers and frequent authors
Included after snowballing (n=18)
The result after snowballing (final result) (n=165)

Fig. 1. The process of document selection.

- *Text in English:* Documents in other languages were immediately excluded.
- *Peer-reviewed publications:* We require that documents are published in journals, proceedings, or books for which we may reasonably assume or in case of doubt (e.g., retrieved via Google Scholar) can verify that they use peer-review. This excludes unpublished research reports and 'gray literature' (e.g., practitioner guidelines, opinion articles, company white papers) as we wished to be assured to a reasonable extent of their quality, as independently assessed through the peer review process. As we wished to select research studies only, editorials or introductions to special issues in academic outlets were also excluded.
- *Secondary studies:* Research review studies were excluded as we were looking for primary studies only. This criterion was easy to apply by inspecting the document's title and abstract.

For deciding on the *inclusion* of the documents that were not excluded by applying the former criteria, we then used the following criteria for deciding on a document's relevance, which were applied by inspecting each document's abstract:

- *Agile Software Development (ASD) context:* After some trials, we learned that the concept of user story is also used in other domains (e.g., healthcare, movies). The term "agile" may also appear in these other contexts, which is why such documents were returned. A document is only relevant if the research context is software development.
- *Focus on user stories:* Documents could have been returned by the search engines just because the term "user story" was mentioned. We consider a document as relevant if, based on the abstract, the document reports on research that has the user story technique (or its use) as an object of study.

Next, we applied again *exclusion* criteria, in the following order:

- *Full text is not available:* For documents that were, according to the inclusion criteria, relevant based on their abstract, but for which we could not access the full text, we contacted the authors to obtain the full text through academic social networks such as ResearchGate, which often succeeded, but not always. That is the reason why we applied this exclusion criterion in the latest phase of the selection as for investigating our research questions, we reckoned that the abstract alone might not be sufficient.
- *Completeness:* To be able to investigate our research questions (see Section 1), the documents must clearly state the research problem or issue investigated, the research question(s)/objective(s), the research methodology, and the research outcomes. If this information could not be retrieved from the abstract, we searched for it in the rest of the document.

Fig. 1 summarizes our search process for published research on user stories. The search was performed on the different databases iteratively with a last run in December 2020. Duplicates returned by more than one search engine were eliminated. In total, 551 documents were automatically extracted by running the search query. This set contained 156 duplicates. The other 395 documents were submitted to our selection criteria. In total, 147 papers (i.e., unique research studies on user stories) were selected. The other documents were eliminated on the grounds of 'gray literature' (41 documents), non-English papers (1 document), full-text not available to us (50 documents), no focus on user stories as object of study in an ASD context (144 documents), and secondary studies (12 documents). No documents were rejected for reasons of incompleteness, which might be explained by the prior exclusion of non-peer reviewed publications.

Next, a limited snowballing process was applied to search for additional studies that were missed by our search strategy. To identify such papers, we analyzed the twelve research reviews. We also searched for other relevant papers, not yet in our set of documents, published by (the few) authors that had more than one publication in our document set. All candidate papers were submitted to our inclusion and exclusion criteria. This snowballing process yielded 18 additional papers. Ultimately, 165 studies were selected. In the next sub-section, we explain how we classified these 165 studies in problem classes to select the studies addressing ambiguity in user stories.

4.2. Selecting studies addressing ambiguity in user stories

A thematic analysis of the 165 selected papers revealed 22 unique research problems (or issues of interest). We classified these research problems as problems related to ambiguity, collaboration, and system design. For classifying the 165 studies that were selected, the corresponding author read through the entire paper and discussed any doubts with the second author.

As defined in Section 1, ambiguity problems are problems related to *the articulation of requirements as user stories, which cause doubtful, vague, and multiple interpretations of these requirements*. Studies addressing ambiguity problems focus on the user story as an artifact for documenting requirements and the interpretation of these requirements. The research thus treats user stories as an output of RE activities. In total, 36 papers were classified as being related to ambiguity in user stories. Their reference details can be found in Table A1 in the Appendix.

The other problem classes contain studies that address problems related to *the use of user stories during the project as a mechanism for facilitating communication and collaboration* (i.e., collaboration problem class) and *the impact that user stories have on the quality of the system and its development* (i.e., system design problem class). Although ambiguity of requirements affects or might cause these problems, all studies in these two classes have in common that they do not investigate the ambiguity in user stories nor propose solutions to mitigate this ambiguity. The research treats user stories as a given input to RE activities, and the focus is on other problems or goals (e.g., estimating effort, prioritizing requirements, and checking conformance of the developed software, based on a given set of user stories). These studies, 129 in total, were thus considered as outside the scope of our literature review.

4.3. Data extraction and classification

As discussed in Section 3, ambiguity in requirements is not a problem that is unique to the user story technique (e.g., [34,37,60,62,63]). To get more insight into how this problem manifests itself in user stories and what possible causes and effects are, we first classified the 36 selected papers according to the linguistic level of ambiguity of the problem, as it is described in the papers, following the classification schema of Berry and Kamsties [64]:

- ***Lexical ambiguity*** is ambiguity at the level of individual words that might unconsciously exist due to words having several meanings because of etymological differences (i.e., homonymy) (e.g., bank as a financial institution or the bank of a river) or related but different meanings (i.e., polysemy) (e.g., bank as a financial institution or the building in which it resides). The role "system administrator" in the first example user story of Section 2, could lead to lexical ambiguity if it is not clear whether "system" refers to all systems of the company or only the system that administers user rights. It is plausible that the first interpretation is most likely, so this could be a case of innocuous ambiguity.
- ***Syntactic ambiguity*** is ambiguity at the sentence level that is present when a sentence can be interpreted using different grammatical structures. An example is "I want to go to the bank with cash money", which can be interpreted as me taking cash money to the bank or me going to a bank that has cash money available. Notice that the user story template does not prevent this kind of ambiguity as the example sentence could be the *I want* ⟨*goal*⟩ segment of the user story when using the Connextra template (see Section 2). The "to reconfirm the systems' access for a registered user" goal in the second example user story of Section 2, could result in syntactic ambiguity because it can be interpreted as the manager requiring this reconfirmation on behalf of a user (who requested continued access but is not a user of the user rights administration system) but also as a manager taking herself the initiative to request a reconfirmation for a specific user. This is probably also innocuous ambiguity.
- ***Semantic ambiguity*** is ambiguity at the phrase (i.e., part of a sentence) or sentence level that exists if there are multiple interpretations of a phrase or sentence, when taken out of context – note that semantic ambiguity at the level of individual words is lexical ambiguity in the classification of Berry and Kamsties [64]. For instance, "sitting on the bank" can be interpreted differently in the contexts of a fisher and a roof carpenter. The interpretation problems noted in Section 2 on the exact nature of the relationships between the role, goal, and benefit segments of the two example user stories, illustrate semantic ambiguity as these two user stories are taken out of their context. It is probable that many of these problems can be resolved by analyzing the other user stories in the epic or theme. For instance, other user stories might confirm or negate whether generating a new password is sufficient to reconfirm a registered user's access to the systems. If not, then there is nocuous ambiguity that, when undetected, might lead to the development of duplicated features (i.e., generating a new password is the same as reconfirming systems' access) or missing features (i.e., generating a new password is just one step in the reconfirmation of systems' access and other steps are not implemented).
- ***Pragmatic ambiguity*** is ambiguity at the phrase or sentence level that is present if the context does not clarify the intended meaning. For instance, in the user story "As a frequent traveler, I want to be able to search tickets so that I can obtain information about rates and times of the flights" [41], the phrase "frequent traveler" could be interpreted as registered frequent flyers according to some airline customer program or as travelers that fly frequently. If the context (e.g., the other user stories in the epic or theme) does not clarify what is meant by "frequent", then both interpretations might persist, and this means pragmatic ambiguity exists. Similarly, in the example elaborated in Section 2, it must be clear what is meant by "registered user". If the context (e.g., other user stories in the epic or theme) does not clarify that a registered user is a user that had credentials (e.g., login, password, user rights to certain systems) before, but lost them, then it is hard to make sense of a phrase like "to reconfirm the systems' access for a registered user". However, in this case, we believe the ambiguity is most probably also innocuous.

Considering that user stories are expressed in natural language, this classification is advantageous for identifying the different types of ambiguity in user stories. From the definitions, context plays an important role in attempting to resolve lexical, syntactic, and semantic ambiguity, whereas it does, by definition, not help in removing pragmatic ambiguity. Although these four linguistic levels of ambiguity apply to words, phrases, and sentences, meaning individual user stories or parts of these, it logically follows that solutions for resolving linguistic ambiguity, other than pragmatic ambiguity, need to consider user stories in the context of other related user stories (e.g., user stories in the same epic or theme) or other information regarding the system to be developed, its application domain, etc. Therefore, this linguistic classification of ambiguity is also useful in understanding the solutions that are proposed in

Table 1
Research assessment following Wieringa et al. [65].

#	Questions	Possible answers
Evaluation research		
1	Is the problem well defined?	$Y = 1, N = 0, P = 0.5$
2	Are the investigated features of the problem clearly described?	$Y = 1, N = 0, P = 0.5$
3	Is the research method support the knowledge claim?	$Y = 1, N = 0, P = 0.5$
4	Is the knowledge claim validated empirically?	$Y = 1, N = 0$
5	Does the result contribute to knowledge base improvements?	$Y = 1, N = 0, P = 0.5$
6	Is there adequate discussion of related studies?	$Y = 1, N = 0, P = 0.5$
Validation research		
1	Is the proposed technique well defined?	$Y = 1, N = 0, P = 0.5$
2	Are the investigated features of the technique clearly described?	$Y = 1, N = 0, P = 0.5$
3	Is the research method support the knowledge claim?	$Y = 1, N = 0, P = 0.5$
4	Is the knowledge claim validated empirically?	$Y = 1, N = 0$
5	Is it clear under which condition the technique possess the features?	$Y = 1, N = 0, P = 0.5$
6	Does the result contribute to knowledge base improvements?	$Y = 1, N = 0, P = 0.5$
7	Is there adequate discussion of related studies?	$Y = 1, N = 0, P = 0.5$
Proposed-of-solution research		
1	Is the proposed problem clearly described?	$Y = 1, N = 0, P = 0.5$
2	Is the proposed technique or the implementation of the techniques for moderately problem novel?	$Y = 1, N = 0, P = 0.5$
3	Is the technique adequately described to allow later validation?	$Y = 1, N = 0, P = 0.5$
4	Is the technique reliable and relevant to this kind of problem?	$Y = 1, N = 0, P = 0.5$
5	Is there adequate discussion of related studies or technical benchmarks?	$Y = 1, N = 0, P = 0.5$

Table 2
Classification of user story ambiguity problems.

Ambiguity problem	Linguistic level of ambiguity	Reference(s)	#studies
Vagueness	Lexical	[66]	1
	Syntactic	[31,33,67]	3
	Semantic	[28,44,54]	3
	Pragmatic	[27,32,68]	3
Inconsistency	Lexical	[49]	1
	Syntactic	[42,47]	2
	Semantic	[41,43,45,53,55,57, 69-75]	13
	Pragmatic	[24,46,76,77]	4
Insufficiency	Semantic	[26,56,78]	3
	Pragmatic	[52]	1
Duplication	Lexical/Semantic	[79,80]	2

the reviewed studies, for obtaining insights into whether and how solutions build on contextual information.

Out of 36 papers, 32 papers present a solution for ambiguity-related problems (see Table A1 in the Appendix**)**. The other four papers describe ambiguity problems with user stories as they are observed in practice, or they empirically validate or evaluate solutions that have been proposed in other papers. For the papers that present solutions, we wished to gather evidence of the effectiveness of these solutions. We, therefore, classified these papers into three groups, following a classification schema for RE papers by Wieringa et al. [65], that distinguishes between studies that justify the proposed solution using an illustrative scenario that is presented as a proof of concept or that is used to compare the solution to other solutions (i.e., *proposed-of-solution research*), studies that demonstrate solution design validity through testing or by means of an experiment or simulation (i.e., *validation research*), and studies that evaluate the quality, effectiveness or efficiency of a solution through observing a real-world implementation presented as a case-study or field study (i.e., *evaluation research*).

Based on this classification, we then applied the research assessment instrument that has been proposed by Wieringa et al. [65]. This instrument, shown in Table 1, uses specific criteria for each research type to assess the quality of the papers. We used this instrument to obtain insights into the extent to which solutions for ambiguity have been validated or evaluated (see Table A2 in the Appendix).

The classification according to linguistic level of ambiguity and the quality assessment of the papers proposing solutions for ambiguity was performed by the corresponding author and reviewed by the second author until consensus was reached.

5. Results

5.1. *How are ambiguity problems defined (RQ1)?*

The thematic analysis (see sub-Section 4.2) aggregated four distinct problems that were identified in the initial set of 165 papers into the ambiguity class: *vagueness* of requirements articulated as user stories, *inconsistencies* between user stories, *insufficiency* of user stories in capturing requirements, and *duplication* of requirements in multiple user stories. Ambiguity problems arise due to user stories having doubtful, imprecise, or multiple interpretations. While studies focusing on the vagueness of requirements formulated as user stories have investigated different sources of ambiguity that (potentially) result in interpretation problems, other studies have investigated the consequences of uncertain or multiple interpretations that can be traced back to ambiguity. Here we found in the literature three distinct problems: user stories being understood as inconsistent; user stories being perceived as insufficiently describing requirements (regarding completeness and precision); and user stories being judged as duplicating functionality.

To answer how these ambiguity problems have been defined and thus obtain deeper insights into their manifestations, causes and effects, we link the problems discussed in the papers to linguistic levels of ambiguity (Table 2).

5.1.1. *Vagueness*

Studies related to vagueness have addressed issues of unclarity related to the intended meaning of a user story. This kind of ambiguity is found at all linguistic levels. From our review of the studies, we learn that vagueness as a *lexical ambiguity* problem manifests itself as unclear word choices in user stories [66].

Vagueness at the level of *syntactic ambiguity* has been observed in the further fragmentation of user story segments into linguistic structures (e.g., subject, verb, adjective) [31,33]. This problem not only impedes the precise interpretation of the expected system structure and behavior [31,67]. It may also result in failure to generate a single system feature from a user story. In addition, Gazzerro et al. [31] remarked that languages with a different structure than English could also engender confusion as a different disambiguation logic could apply. All three studies concur that due to vagueness (at the syntactic level), the same user story might be translated into different required system features [31,33,67].

Several studies have attributed vagueness as a *semantic ambiguity* problem of user stories to the absence of a particular scope or reference that helps interpret the user story expression. Therefore, user stories might be wrongly decoded or unconsciously skipped [28,44,54]. In the study performed by De Souza et al. [28], vagueness occurs due to the use of terminology from particular fields (e.g., healthcare, military, oil and gas). Araujo and Siqueira [54] attribute vagueness to the absence of a method to reason about high-level system goals and to decompose them into user stories. Lin et al. [44] believe that cognitive factors (e.g.,

Table 3
Overview of the proposed solutions for ambiguity in user stories.

Ambiguity problem	Linguistic level of ambiguity	Solution type	Proposed solution	Reference (s)
Vagueness	Lexical	Word glossary (Framework)	User preferences	[66]
	Syntactic	Controlled natural language (Framework)	Human-assisted analysis	[31]
			Sentence patterns based on Standford POS tags and EBNF grammar	[33]
			Use of the Tomita-parser	[67]
	Semantic	Ontology (Framework)	Domain ontology	[28]
		Model	Goal model (i* model)	[44,54]
	Pragmatic	Personas (Framework)	Personas	[32]
		Model	Process models (BPMN)	[27,68]
Inconsistency	Lexical	Word glossary (Framework)	LEL (lexicon)	[49]
	Syntactic	Algorithm	Transformation into UML use case diagrams	[42]
			Transformation into robustness diagrams	[47]
	Semantic	Ontology (Framework)	OWL ontology	[41]
		User story template (Framework)	OODARE	[69]
		Model	Goal model (i* model)	[53,72]
			Goal model (Rationale Trees/ Diagrams)	[73,74]
			Goal model (Agent-oriented goal model)	[55,57]
			UML model (Use case diagram, Sequence diagram, class diagram, object model)	[43,45, 71,75]
			ER diagram	[70]
Insufficiency	Semantic	Taxonomy (Framework)	Taxonomy of existing user stories	[26]
		Ontology (Framework)	Business Process Ontology	[56]
		Algorithm	Case-based Reasoning (CBR)	[78]
	Pragmatic	User story template (Framework)	QUESt user story	[52]
Duplication	Lexical	Algorithm	Similarity metrics	[79,80]

knowledge background, experience) are crucial to avoid vagueness in user story formulation.

Finally, studies related to *pragmatic ambiguity* of user stories have explored the role of human cognition in identifying and grasping the meaning of user stories [68] and investigated the effectiveness of methods (i.e., BPMN diagrams, Personas) to improve the unambiguous understanding of requirements documented as user stories [27,32,68].

5.1.2. *Inconsistency*

Inconsistency is a problem that occurs within a set of related user stories if interpretations conflict, hence it is a problem of ambiguity. The reviewed studies showed that inconsistency may result in requirements being incomplete [41,43,46,47,49,53,73,74,76] and systems being non-compliant [24,43,45,46,49,55,57,69-76].

Regarding *lexical ambiguity*, the non-standard use of vocabulary has been identified as a source of user story inconsistency [49]. This problem is comparable to the problem of vagueness studied in [28] that has user story clarity and conciseness as main research objectives. However, study [49] is classified as a lexical ambiguity problem associated with inconsistencies of user stories because the study highlighted inconsistency problems frequently found during requirements validation and exploited the use of vocabularies for expressing user stories.

As a *syntactic ambiguity* problem, inconsistency has been identified through user story fragmentation into Part-of-Speech tags (e.g., verb, noun) [42,47]. In [42], inconsistency was detected when a tool was used to comprehend complex sentences in user stories. Similarly, the problem was found in [47] when the robustness diagram wrongly interpreted information from user stories.

In terms of *semantic ambiguity*, imprecise scope determination [41, 45,53,72], inaccurate goals decomposition [43,55,57,73,74], and misinterpretations [69-71,75] have been investigated as consistency obstacles. For imprecise scope determination, inconsistency problems show up as discrepancies or missing user stories when user stories are derived from goal models [53,72] or as incompatible specifications of systems [41,45] caused by minimal visual documentation [41,45,53, 72]. Regarding inaccurate goal decomposition [43,55,57,73,74] and misinterpretations [69-71,75], Wautelet et al. [43,73,74] showed that inconsistency issues at a syntactic ambiguity level might evolve into semantic ambiguity when user story elements are not correctly classified. This finding was supported by other studies [55,57], which also remarked missing stakeholder input and invalid requirements as the main causes of non-compliant software [55,57,69-71].

Finally, user story inconsistency as a *pragmatic ambiguity* issue has been investigated by considering human cognition as an influential factor in achieving a consistent understanding of the meaning of a set of related user stories [24,46,76,77]. These studies investigated cognitive factors such as representation style and interaction patterns [77] and modeling experience [24,76]. Some of these studies (i.e., [46,76]) also validate solutions proposed in other papers.

5.1.3. *Insufficiency*

Ambiguity can also lead to a user story being incomplete regarding the description of a feature that a user expects from a system. Several studies have mentioned neglected requirements [52,56], while team member turnover has been identified as a cause [26,78]. Insufficiency as a *semantic ambiguity* problem has been demonstrated by reviewing a user story in the context of previously written user stories [26,56,78]. In [78], high turnover impacted the inability of development teams to relate clients' emails with specific user stories. Limitations of development teams were also highlighted in [26,56] as a motivation to reuse previous user stories.

Finally, studies focusing more on insufficiency as *pragmatic ambiguity* have investigated how technical users can be supported in understanding user stories more sufficiently [52].

5.1.4. *Duplication*

Finally, user story duplication has been considered initially as a problem of *lexical ambiguity*. The scarce research on the topic has investigated the similarity of user stories using similarity metrics, particularly the Cosine Similarity [79,80], and Jaccard Index [79] metrics. However, as explained further in sub-Section 5.2.1, those studies concluded that to resolve duplication, it needs to be considered as a *semantic ambiguity* problem.

5.2. *What solutions are proposed for ambiguity (RQ2)?*

As already noted in Section 4, out of the set of 36 papers, 32 papers present a solution for problems of ambiguity in user stories. Table 3 provides an overview of these solutions. The four papers that did not

Table 4
Papers including a validation or evaluation of the proposed solution.

Ambiguity problem	Linguistic level of ambiguity	Type of outcome	Validation/ Evaluation method	Reference (s)
Vagueness	Syntactic	Controlled natural language (Framework)	Controlled experiment	[31]
	Semantic	Ontology (Framework)	Testing	[28]
		Model	Quasi-experiment	[44]
	Pragmatic	Personas (Framework)	Quasi-experiment	[32]
		Model	Field study	[27]
			Controlled experiment	[68]
Inconsistency	Lexical	Word glossary (Framework)	Controlled experiment	[49]
	Syntactic	Algorithm	Controlled experiment	[42]
	Semantic	Ontology (Framework)	Testing	[41]
		Model	Case study	[45,57, 71]
Insufficiency	Semantic	Taxonomy (Framework)	Case study	[26]
		Algorithm	Quasi-experiment	[78]
	Pragmatic	User story template (Framework)	Quasi-experiment	[52]
Duplication	Lexical	Algorithm	Controlled experiment	[79,80]

Table A1
Reviewed Studies.

ID	Author(s)	Ref.	RQ1	RQ2	RQ3
S8	T. R. Silva et al.	[24]	2	1	3
S10	H. Ordóñez et al.	[27]	1	7	7
S13	M. Trkman et al.	[68]	1	7	5
S14	A. Jaqueira et al.	[72]	2	7	2
S16	P.L. de Souza et al.	[28]	1	2	6
S18	S. M. Sohan et al.	[78]	3	6	4
S21	Y. Wautelet et al.	[43]	2	7	2
S22	Y. Wautelet et al.	[73]	2	7	2
S23	Y. Wautelet et al.	[74]	2	7	2
S27	L. Müter et al.	[33]	1	9	2
S28	Y. Wautelet et al.	[76]	2	1	4
S34	B. De Brock	[75]	2	7	2
S35	A. Zeaaraoui et al.	[69]	2	4	2
S74	T. Avdeenko et al.	[67]	1	9	2
S75	G. Stella et al.	[31]	1	9	5
S76	T. R. Silva et al.	[41]	2	2	6
S77	C. Agra et al.	[53]	2	7	2
S78	L. B. De Araujo and F. L. Siqueira	[54]	1	7	2
S79	J. Lin et al.	[44]	1	7	4
S80	T. Tenso and K. Taveter	[55]	2	7	2
S81	P. Kamthan and N. Shahmir	[66]	1	8	2
S84	L. A. Lopes et al.	[32]	1	5	4
S85	J. Melegati and X. Wang	[52]	3	4	4
S86	N. Prakash and D. Prakash	[70]	2	7	2
S89	J. Jia et al.	[77]	2	1	5
S93	C. Thamrongchote and W. Vatanawood	[56]	3	2	2
S98	T. Tenso et al.	[57]	2	7	3
S99	F. Wanderley et al.	[45]	2	7	3
S125	E. Dilorenzo et al.	[26]	3	3	3
S126	M. Urbieta et al.	[49]	2	8	5
S144	N. Costa et al.	[71]	2	7	3
S145	R. Barbosa et al.	[79]	4	6	5
S146	M. Elallaoui et al.	[42]	2	6	5
S147	F. Gilson and C. Irwin	[47]	2	6	2
S150	B. Jiang et al.	[80]	4	6	5
S151	Y. Wautelet et al.	[46]	2	1	4

present a new solution, are not included in Table 3 as they are not relevant for answering RQ2.

We distinguished three broad classes of solutions in the 32 papers. First, 15 papers propose the use of conceptual models or software models for understanding and analyzing the interaction among and dependencies between user stories. Second, 5 papers propose algorithmic solutions (i.e., a procedure of well-defined and sequenced actions to be performed on user stories) to solve various types of problems. Third, 12 papers propose solutions that aim at improving user story writing or user story templates for such writing. Any artifact that is proposed to help with user story writing was classified as a *framework* as it operationalizes conceptual structures or ideas for improving the formulation of requirements as user stories. However, if the research proposes an algorithm to use such artifacts, then it was classified as algorithm. So, the framework class includes solutions as artifacts (other than models) for which no algorithm has been proposed. In total, six artifact types were identified in the selected papers: *word glossaries, controlled natural language, taxonomies, ontologies, user story templates*, and *Personas*.

Our review of the proposed solutions is organized by the linguistic ambiguity level at which problems manifest themselves, as we found that the chosen types of solutions are related to these linguistic ambiguity levels.

5.2.1. *Lexical ambiguity*

As mentioned in sub-Section 5.1.4, two studies focused on detecting *duplication* by measuring similarity in the used words. However, it was found that these techniques were not sufficient, and that the problem needed to be addressed at the semantic level of ambiguity. Consequently, it was recommended to calculate similarity taking into account user story dependencies [80]. Also, semantic similarity measures (i.e., the WuP, Lin, and Lesk-A similarity metrics) were recommended for user story duplication detection [79]. As similarity measurements involve a number of steps, we consider these solutions as **algorithms**.

For tackling *vagueness* at the level of lexical ambiguity, word glossaries based on user preferences that are observed in previously written user stories are proposed as a solution in [66]. The unstandardized expression of user stories was indicated as the primary cause of lexical ambiguity leading to user story *inconsistency*, as noted in [49]. This problem was resolved by advocating the use of a word glossary that employed the Language Extended Lexicon (LEL) for user story writing. We classified word glossaries as a **framework** type of solution.

5.2.2. *Syntactic ambiguity*

Studies proposing solutions for syntactic ambiguity address problems of *vagueness* [31,33,67] and *inconsistency* [42,47]. For increasing the precision of user story articulation, sentence patterns were introduced in [33]. For the same purpose, a parsing technique was proposed in [67], while [31] suggested a human-assisted analysis technique. Those solutions were classified as **frameworks**. Other studies proposed **algorithms** that resolve syntactic ambiguity by automatically transforming user stories into models – Use Case diagrams [42] and Robustness diagrams [47].

All those solutions have been supported by various NLP techniques, such as the TreeTagger parser [42,47], the spaCy library parser [47], the Tomita-parser [67], Standford POS tags [33], and The Extended Backus-Naur form (EBNF) grammar [33].

5.2.3. *Semantic ambiguity*

Solutions to semantic ambiguity problems have been researched most. Different types of **models** have been proposed to address problems related to issues of *inconsistency* and their consequences. The general idea is to consider user stories in relation to each other or in the context of the application domain, the goals of system stakeholders, or the system architecture that would result from implementing the user stories.

Proposed models are often goal models. I* goal models have been

Table A2
Quality assessment following Wieringa et al. [65].

● Proposed of solution

ID	Author(s)	1	2	3	4	5	Total score	Qual. (%)
S14	A. Jaqueira et al.	1	0	1	1	0	3	60
S21	Y. Wautelet et al.	1	1	0,5	1	1	4,5	90
S22	Y. Wautelet et al.	1	1	1	1	1	5	100
S23	Y. Wautelet et al.	1	1	1	0,5	1	4,5	90
S27	L. Müter et al.	1	0,5	1	1	0	3,5	70
S34	B. De Brock	1	0,5	1	0,5	0	3	60
S35	A. Zeaaraoui et al.	0,5	0,5	0,5	0,5	0	2	40
S74	T. Avdeenko et al.	1	0,5	1	1	1	4,5	90
S77	C. Agra et al.	1	0,5	1	0,5	1	4	80
S78	L. B. De Araujo and F. L. Siqueira	1	0	1	1	0	3	60
S80	T. Tenso and K. Taveter	1	0,5	1	1	1	4,5	90
S81	P. Kamthan and N. Shahmir	1	0,5	1	0,5	1	4	80
S86	N. Prakash and D. Prakash	1	0,5	1	0,5	1	4	80
S93	C. Thamrongchote and W. Vatanawood	1	0,5	1	1	1	4,5	90
S147	F. Gilson and C. Irwin	1	0,5	1	1	1	4,5	90

● Validation research

ID	*Author(s)*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*Total score*	*Qual. (%)*
S13	M. Trkman et al.	1	1	1	1	1	1	1	7	100
S16	P.L. de Souza et al.	1	1	1	1	1	1	1	7	100
S18	S. M. Sohan et al.	1	1	1	1	1	1	1	7	100
S28	Y. Wautelet et al.	1	1	1	1	1	1	1	7	100
S75	G. Stella et al.	1	0,5	1	1	1	1	1	6,5	93
S76	T. R. Silva et al.	1	1	1	1	1	1	1	7	100
S79	J. Lin et al.	1	0,5	0,5	1	1	1	0	5	71
S84	L. A. Lopes et al.	1	1	1	1	1	1	1	7	100
S85	J. Melegati and X. Wang	1	1	0,5	1	1	1	1	6,5	93
S89	J. Jia et al.	1	1	1	1	1	1	1	7	100
S126	M. Urbieta et al.	1	1	1	1	1	1	1	7	100
S145	R. Barbosa et al.	1	0,5	0,5	1	0,5	1	0	4,5	64
S146	M. Elallaoui et al.	1	1	1	1	1	1	1	7	100
S150	B. Jiang et al.	1	1	1	1	1	1	1	7	100
S151	Y. Wautelet et al.	1	1	1	1	1	1	1	7	100

● Evaluation research

ID	*Author(s)*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*Total score*	*Qual. (%)*
S8	T. R. Silva et al.	1	1	1	1	1	1	1	7	100
S10	H. Ordóñez et al.	1	1	1	1	1	1	1	7	100
S98	T. Tenso et al.	1	1	1	1	1	1	1	7	100
S99	F. Wanderley et al.	1	1	0,5	1	1	1	1	6,5	93
S125	E. Dilorenzo et al.	1	1	1	1	1	1	1	7	100
S144	N. Costa et al.	1	1	1	1	1	1	1	7	100

proposed for user story functionality mapping [53,72], Product Backlog specification [44], and Sprint Backlog specification [54]. Also, extensions of the i* formalism have been proposed to solve semantic ambiguity issues with user stories. For instance, Rationale Trees (also called Rationale Diagrams) [73,74] have been proposed to prevent missing requirements, provide for a more consistent user story implementation, and improve system performance [81]. The use of Agile Agent-Oriented Goal Models was proposed in [55,57] to elaborate user stories into system features and to specify the Product Backlog.

Other proposed models are those generally associated with software development methodologies other than ASD methodologies. UML use case diagrams have been proposed to map user story functionality [43], improve user story clarity [75], and compose user story scenarios [71]. UML object models [69], class diagrams [45], and sequence diagrams [71,75] have been used to visualize system structure and behavior. ER diagrams have been used to resolve conflicts during the specification of user stories [70].

Apart from models, **frameworks** such as domain ontologies [28], business process ontologies [56], and OWL ontologies [41] have been proposed to address issues of *vagueness* [28], *inconsistency* [41], and *insufficiency* [56]. Apart from ontologies, also other proposed frameworks exploit contextual information to make existing knowledge related to user stories explicit. In [26], taxonomies of existing user stories are created to recommend potentially relevant user stories. In [69], the structure of an object-based model is proposed as a new user story template. Finally, in [78], a sentence glossary for writing user stories was presented – we classified the latter study as **algorithm** because it also presented a Case-Based Reasoning algorithm for using the sentence glossary.

5.2.4. *Pragmatic ambiguity*

Solutions proposed for addressing ambiguity at the pragmatic level are scarce. The use of Personas as a technique to improve the understandability of user stories was proposed in [32]. As a result of an experimental study, a writing guideline for user stories was presented in [52]. We classify both solutions as **frameworks**. The use of process **models** to enhance the understanding of user stories was investigated in [27], while [68] also investigated how such models help in eliciting user stories.

5.3. *RQ3: What evidence of the effectiveness of these solutions is available?*

The 32 papers on ambiguity in user stories that present solutions include 17 papers that include validation or evaluation of these solutions. This means that 15 papers demonstrated or illustrated their solution but lack in giving empirical evidence of solution design validity or the effectiveness of the solution when implemented.

The 17 papers that include a validation or evaluation of the presented solution are classified into the Framework (i.e., [26,28,31,32,41,49,52]), Algorithm (i.e., [42,78-80]), and Model (i.e., [27,44,45,57,68,71]) solution types (see Table 4). In what follows, we review the evidence that is presented of the effectiveness of these solutions. We also discuss the studies [46,76] that validated solutions presented in [73,74], to get a more complete picture for answering RQ3.

Starting with solutions for *lexical ambiguity*, the papers proposing the use of similarity metrics for detecting duplication of requirements in user stories confirmed by means of experiments that similarity analysis is an adequate solution [79,80]. The experiments showed that semantics-based metrics (i.e., WUP, Lin, and Lesk-A similarity metrics) outperform lexical-based metrics (i.e., Cosine Similarity, Jaccard Index), which indicates that focusing solutions on removing lexical ambiguity alone, is not effective for avoiding user story duplication. However, the use of a lexicon when writing user stories was shown to be effective in ensuring consistency in a set of user stories by means of a controlled experiment [49].

Regarding *syntactic ambiguity*, the use of an automatic transformation of user stories into use case diagrams was tested experimentally in [42]. The transformation was supported by the TreeTagger parser to chunk the user story elements into the expected grammatical structure and to re-arrange the elements into the graphical model. The results were promising as a high number of user stories could be processed accurately (i.e., precision = 87%; recall = 85%).

Next, for *semantic ambiguity*, several papers have presented evidence of the effectiveness of proposed solutions. Starting with framework type of solutions, the usefulness of a domain ontology for adding clear meaning to user stories was demonstrated in [28] by means of an ontology verification process performed using scenarios and semantic reasoners (i.e., FACT++, HermiT, and Pellet). Similarly, the automatic checking of user story consistency by means of OWL ontologies was tested in [41]. The creation of a sentence glossary for writing user stories using a Case-Based Reasoning algorithm has been shown effective using a simulation study in [78]. In [26], it was shown in a case study of two running ASD projects, after prior testing with user stories of ended projects, that contextual information captured in a taxonomy format could be used to ensure user story consistency.

Continuing with the use of models as solution approach, it was shown in [44], using different validation approaches, that the use of goal models helps non-technical users to better grasp the system value and better estimate the project duration. The benefits of Rationale Tree/-Diagrams, presented in [73,74], for identifying the functional elements in a set of user stories were shown using an experiment with novice modellers in [46] (not included in Table 4), while the ability to build such models was experimentally assessed for both novice and experienced modellers in [76] (not included in Table 4). In [71], the proposed method for using UML use case diagrams to compose user story scenarios was evaluated in a case study.

Apart from these validation experiments, real-world case studies have been used for evaluating proposed model-based solutions. It was shown in [57] that the AAOM goal model helps express user expectations of system features into the user story format. Similarly, it was shown in [45] that UML class diagrams improve requirements visualization, enhance user story consistency, help shorten test case time, help identify missing requirements, and reduce the time to elicit, document, and analyze user stories and disseminate the understanding to all team members.

To end with solutions proposed for *pragmatic ambiguity* problems, it was shown by means of a field experiment involving 11 ASD projects [27] that the use of process models is beneficial for a more unambiguous understanding of requirements captured in user stories. Similarly, in [68], a student experiment indicated that the use of BPMN models improves the understanding of user stories. Further, the solutions employing Personas [32] and an alternative user story template [52] were also validated experimentally. Finally, although not presenting a solution (hence not listed in Table 4), paper [24] was classified as evaluation research (see also table A2 in the Appendix) as it investigated the consistency of user stories written by POs using a case-study.

6. Discussion

6.1. *How has academic research addressed ambiguity in user stories?*

This study has presented the results of a systematic literature review of 36 unique studies that investigate problems and describe solutions related to ambiguity in user stories. Our review showed that there are four distinct ambiguity-related problems (i.e., vagueness, inconsistency, duplication, and insufficiency) and that research has defined these problems considering four linguistic levels at which ambiguity manifests itself in user stories (i.e., lexical, syntactic, semantic, pragmatic). Vagueness and inconsistency have received the most research attention with respectively 10 and 20 studies. These are also the only two problems that have been defined at all four linguistic ambiguity levels. User story vagueness is a problem that occurs during requirements elicitation and documentation activities as a direct consequence of user stories being a (structured) natural language RE artifact. On the other hand, user story inconsistency is a problem that mostly emerges during requirements analysis and negotiation activities when user stories are used as a basis for system specification, project estimation, and work prioritization activities. We learned from our review that vagueness might lead to other problems, like imprecision of requirements and inconsistencies between requirements. Inconsistency itself has been shown to result in incomplete requirements and systems being non-compliant. Another insight we gained from the review was that vagueness has mostly been investigated considering individual user stories, while problems of inconsistency, duplication and insufficiency manifest themselves when comparing user stories.

As to the question of what causes ambiguity, the answer depends on the linguistic level of ambiguity considered. This linguistic level also determines the solution approach taken, more than the actual type of ambiguity problem solved. By analyzing these relationships, four clear patterns emerged from our review.

First, lexical ambiguity is caused by idiosyncratic word choices or non-standard use of vocabulary. For the detection of lexical ambiguity, the use of similarity metrics is proposed for the problem of duplication of system features in user stories. For solving lexical ambiguity, the use of word glossaries is advised. Most of these solutions (3 out of 4) have been experimentally validated.

Second, syntactic ambiguity has been identified by fragmenting user stories into linguistic structures (e.g., parts-of-speech). Solutions consist of controlled natural language mechanisms (for vagueness) or the automatic transformation of user stories into models (for inconsistency). All these solutions heavily rely on the use of NLP techniques. Validation of these solutions is, however, scarce (2 out of 5).

Semantic ambiguity has been investigated most (19 studies). The most identified causes for semantic ambiguity are lack of precise system scope determination, inadequate support for decomposing system goals into system features, and cognitive factors like domain knowledge and experience. Regarding solutions, a third pattern we observe is the dominance of approaches (13 studies) that propose the use of models, typically goal models or UML diagrams, for tackling problems of vagueness or inconsistency. Closely related to models, also ontologies (3 studies) and taxonomies (1 study) have been proposed as a solution for semantic ambiguity problems, also including insufficiency. Models, ontologies and taxonomies address causes of semantic ambiguity by clarifying system scope and goals. Further, these solutions provide visualization support for detecting inconsistencies and analyzing dependencies between user stories. They also offer contextual information that is useful for improving the clarity of the requirements documented as user stories.

Related to the evidence of the effectiveness of proposed solutions, a fourth pattern that we observe is that solutions based on models have been less validated or evaluated than other types of solutions. Of the 13 papers proposing the use of models, three papers present a field or case study that evaluates the solution. Only one paper presents an experimental validation, although some nuance is needed here as the use of Rationale Diagrams/Trees (proposed in [73,74]) has been experimentally validated in other papers (i.e. [46,76],). Still, more than half of the solutions based on the use of models have not been validated or evaluated, so evidence of their effectiveness is lacking.

Regarding pragmatic ambiguity, no clear patterns in causes or types of solutions could be discerned. This type of ambiguity, which is present for vagueness, inconsistency, and insufficiency, is different as it cannot be avoided or removed by just considering contextual information. We found that this part of the research has no clear identity as also the few studies that do not present solutions but investigate cognitive factors that lead to ambiguity belong to this class. The two studies investigating the use of process models to clarify the understanding or elicitation of user stories also assume the linguistic level of pragmatic ambiguity as research context.

A final insight obtained from the review is that validation studies employing testing or experimentation methods (15 papers) outnumber evaluation studies (6 papers). The evaluation of proposed solutions through implementing them in practice and observing how well they work or how their usefulness is perceived by users, is rare.

6.2. Research gaps and opportunities

Based on our systematic literature review of this research area, we can now identify opportunities for research, considering different research gaps. First, studies that investigate cognitive factors as causes of ambiguity are scarce. Some research on human cognition and its impact on the formulation and interpretation of user stories has been done. Factors investigated include domain knowledge, experience, representation style, and interaction patterns amongst team members. Also, team turnover has been shown to have a negative effect on the clarity of user stories. However, most studies that we reviewed propose solutions like glossaries, taxonomies, ontologies, and models that can be used to mitigate ambiguity caused by cognitive factors but without directly addressing these factors. These solutions are usually not developed from a deep understanding of these cognitive factors and the specific context in which these factors are at play. Therefore, an avenue for future research is to investigate more deeply the human-related causes of ambiguity and seek for solutions aimed specifically at factors related to cognition and human behavior. Study [32] that introduced the use of Personas is a nice example of such research.

Second, we notice that all proposed solutions, whether validated/evaluated or not, aim at solving specific problems related to ambiguity in user stories. The proposed solutions have generally been addressing these problems as they are defined at specific linguistic levels of ambiguity. These proposals can thus be considered partial solutions to the multi-faceted problem of ambiguity in writing user stories and using them in the ASD process. In our literature review, we found no studies that tried to combine different solutions into a more comprehensive approach to tackle ambiguity at different linguistic levels, for different consequences regarding the quality of the user stories, and in different phases of the ASD process. Our study shows that a considerable amount of knowledge is available in the literature, yet there is no evidence of this knowledge being synthesized and integrated into a more comprehensive solution approach. We did find indications that narrowing down solutions to specific linguistic levels of ambiguity is not optimal. For instance, the two studies addressing the problem of duplicated functionality in user stories found that solutions approaching the problem as a lexical ambiguity problem were outperformed by solutions that consider the same problem as a semantic ambiguity problem. Also, studies by Wautelet and collaborators indicated that inconsistency problems from syntactic ambiguity could evolve into semantic ambiguity problems if they are not properly addressed. This search for more holistic approaches in detecting and solving ambiguity in user stories is a clear suggestion for future research.

Third, we noted that, apart from some solutions based on the use of models (e.g. [44,45,68,71,73,74],), most studies investigate ambiguity within the scope of individual user stories or between a pair of user stories. This is in line with previous literature reviews [59,60] that observed that most of the reviewed studies aim at detecting ambiguity in single sentences rather than in fragments of text covering multiple requirements. As we demonstrated with our example in Section 2, ambiguity at all linguistic levels can depend on the scope considered. Only considering one of the two example user stories even takes away most of the ambiguity. On the other hand, considering more user stories might reduce ambiguity, but perhaps also introduce new ambiguity. What is important in our opinion, and underemphasized in the reviewed studies, hence a research gap, is that considering ambiguity as a problem of a set of related user stories (e.g., an epic or theme in Scrum) is more relevant than considering it is a problem of the user stories themselves. Further research can be done on solutions for ambiguity in a set of user stories that collectively describe a software system or major functional component of such systems.

Finally, our review showed that for many problems, alternative solutions have been proposed in the literature. Apart from [68], we found no studies that compared alternative solutions. Also, real-life experiences with solutions proposed by researchers are scarce. A practitioner survey on how ambiguity problems related to requirements in ASD projects are being experienced and handled would be a nice complement to the academic literature review we present in this paper.

6.3. Study limitations

Despite rigorously following systematic review guidelines, our research has some limitations. First, because of access limitations of the digital libraries of our institute, we excluded papers for which we could not inspect a full-text version. After removing all duplicates from the initial search result, applying our exclusion and inclusion criteria, and adding the papers found through snowballing, a total of 215 relevant studies were identified, of which 50 (i.e., 23%) could not be obtained in full-text version. This is a relatively large proportion, even if we tried to obtain as many papers as possible by contacting the authors. On the other hand, the assumption that researchers have access to all academic literature is a limitation of the Systematic Literature Review methodology, even if it is not always acknowledged. Consequently, due to lack of reference, it is hard to evaluate the severeness of this limitation.

Second, we only selected papers that have been written in English. Given that we intended to review only the 'academic' literature, we don't believe this limitation is severe as the academic literature in Software Engineering is generally written in English. We don't consider not having reviewed the 'gray literature' as a validity threat because this was a deliberate research design choice.

Third, we made a mistake by including Rational Unified Process (RUP) and not Scaled Agile Framework (SAFe) in our search string. As the term 'agile' was part of our search string, we don't believe in having missed relevant papers unless they only mentioned the acronym SAFe (and not 'agile' in full) in title, keywords, and abstract.

Fourth, the selection and initial classification of papers was performed by one (junior) researcher, while doubts and possible interpretation problems were discussed with a senior researcher. A larger team of researchers might reduce some of the inherent subjectivity in these processes, however, at the expense of an increased effort and time investment.

Fifth, the classification of the 36 finally selected studies using the linguistic levels of ambiguity proposed by Berry and Kamsties [64] and the quality assessment of these studies based on the research assessment instrument of Wieringa et al. [65], were performed by the corresponding author, while interpretation problems or doubts were discussed with the second author. The final classification and quality assessment results are based on a consensus of both authors who discussed and resolved conflicts amongst them. Also, the investigation of the research questions and the identification of research gaps and suggestion of research opportunities was the joint effort of the authors consisting of a junior and senior researcher. We acknowledge that a larger team of researchers might reduce some of the inherent subjectivity in this process, however, at the expense of an increased effort and time investment.

7. Conclusion

This paper presented the first systematic literature review of the research that focused on investigating or solving problems related to the ambiguity of requirements that are documented as user stories. Ambiguous user stories are a problem that, if undetected, propagates throughout the Agile software development process and ultimately affects the quality of the software itself. Our review, up to date till December 2020, shows that different solutions have been developed to identify and mitigate problems that manifest themselves at different linguistic levels of ambiguity.

Research addressing ambiguity in user stories is, however, with 36 papers in total and the oldest paper in the set dating from 2010, limited when compared to the initial set of 165 papers reporting on research on user stories over the period 2001–2020. It is therefore not surprising that several research gaps were identified. There is a lack of research investigating human behavior-related and cognitive factors causing ambiguity and solutions that specifically address those factors. Further, ambiguity is seldom investigated as a problem of a set of user stories (e.g., a theme or epic in Scrum) as most studies approach it as a problem residing in a user story or between user stories. We also noted a lack of holistic solutions to ambiguity-related problems as most studies focus on solving specific problems (e.g., inconsistency, vagueness) defined at specific linguistic levels of ambiguity (e.g., semantic, pragmatic). The area also lacks studies that compare different solutions and evaluate solutions when implemented in the field. Given that ambiguity in user stories is the mother of many other problems related to the quality of requirements in ASD projects, we believe this to be a fruitful field for future research.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was partially funded by the Indonesia Endowment Fund for Education (LPDP) (Contract S-1003/LPDP.4/2019) and Universitas 17 Agustus 1945 Surabaya.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.infsof.2022.106824.

Appendix

Index code used in Table A1:

RQ1	Vagueness	1
	Inconsistency	2
	Insufficiency	3
	Duplication	4
RQ2	Not applicable	1
	Ontology	2
	Taxonomy	3
	User story template	4
	Personas	5
	Algorithm	6
	Model	7
	Word glossary	8
	Controlled natural language	9
RQ3	Not applicable	1
	Illustrative scenario	2
	Case study	3
	Quasi-experiment	4
	Controlled experiment	5
	Testing	6
	Field study	7

Table A2

References

[1] R. Hoda, N. Salleh, J. Grundy, The rise and evolution of agile software development, IEEE Softw 35 (5) (2018) 58–63, https://doi.org/10.1109/MS.2018.290111318.
[2] K. Beck, J. Grenning, R. Martin, M. Beedle, J. Highsmith, S. Mellor, Manifesto for agile software development, Agil. Alliance (2001). http://agilemanifesto.org/.
[3] H. Svensson and H. Martin, "Introducing an Agile Process in a Software Maintenance and evolution organization," 2005, doi: 10.1109/CSMR.2005.33.
[4] B. Ramesh, L. Cao, R. Baskerville, Agile requirements engineering practices and challenges: an empirical study, Inf. Syst. J. 20 (5) (2010) 449–480, https://doi.org/10.1111/j.1365-2575.2007.00259.x.
[5] B. Meyer, Agile!: The good, the Hype and the Ugly, 9783319051, Springer International Publishing, Cham, 2014.
[6] F. Paetsch, A. Eberlein, and F. Maurer, "Requirements Engineering and Agile Software Development," in *Twelfth IEEE International Workshop on Enabling Technologies: infrastructure for Collaborative Enterprises*, 2003, pp. 1–6.
[7] O. Liskin, K. Schneider, F. Fagerholm, J. Münch, Understanding the role of requirements artifacts in kanban, in: Proceedings of the 7th International

Workshop on Cooperative and Human Aspects of Software Engineering - CHASE 2014, 2014, pp. 56–63, https://doi.org/10.1145/2593702.2593707.
[8] V.T. Heikkila, D. Damian, C. Lassenius, M. Paasivaara, A mapping study on requirements engineering in agile software development, in: Proceedings - 41st Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2015, 2015, pp. 199–207, https://doi.org/10.1109/SEAA.2015.70.
[9] A. Cockburn, J. Highsmith, Agile software development, the people factor, Computer (Long. Beach. Calif). 34 (11) (2001) 131–133, https://doi.org/10.1109/2.963450.
[10] I. Gartner, "Results summary: agile in the enterprise," 2019. [Online]. Available: https://circle.gartner.com/Portals/2/Resources/pdf/Agile in the Enterprise 2019 - Results Summary (updated).pdf.
[11] Digital.ai, "15th state of agile report," 2021. [Online]. Available: https://stateofagile.com/#ufh-i-661275008-15th-state-of-agile-report/7027494.
[12] Versionone.com. and Collab.net., "the 12th Annual State of Agile Report," 2019.
[13] K. Schwaber and J. Sutherland, "The scrum guide the definitive guide to scrum: the rules of the game," 2020. [Online]. Available: https://creativecommons.org/licenses/by-sa/4.0/legalcode.
[14] M. Cohn, Succeeeding with Agile Software Development Using Scrum, Pearson Education, Ann Arbor, Michigan, 2010.
[15] M. Cohn, User Stories Applied For Agile Software Development, 13th ed., Pearson Education, Inc., Boston, 2009.
[16] J. Choma, L.A.M. Zaina, D. Beraldo, UserX story: incorporating UX aspects into user stories elaboration,", in: International Conference on Human-Computer Interaction, 2016, pp. 131–140, https://doi.org/10.1007/978-3-319-39510-4_13, vol. 9731, no. July.
[17] A.M. Moreno, A. Yagüe, Agile user stories enriched with usability, in: International Conference on Agile Software Development, 2012, pp. 168–176, https://doi.org/10.1007/978-3-642-30350-0_12, vol. 111 LNBIP.
[18] V. Kannan, et al., User stories as lightweight requirements for agile clinical decision support development, J. Am. Med. Informatics Assoc. 26 (11) (2019) 1344–1354, https://doi.org/10.1093/jamia/ocz123. Nov.
[19] S. Gossage, J.M. Brown, R. Biddle, Understanding Digital Cardwall Usage,", in: 2015 Agile Conference, 2015, pp. 21–30, https://doi.org/10.1109/Agile.2015.16. Aug.
[20] J. Savolainen, J. Kuusela, A. Vilavaara, Transition to agile development rediscovery of important requirements engineering practices, in: Proc. 2010 18th IEEE Int. Requir. Eng. Conf. RE2010, 2010, pp. 289–294, https://doi.org/10.1109/RE.2010.41.
[21] J.M. Rivero, J. Grigera, G. Rossi, E. Robles Luna, F. Montero, M. Gaedke, Mockup-driven development: providing agile support for model-driven web engineering, Inf. Softw. Technol. 56 (6) (2014) 670–687, https://doi.org/10.1016/j.infsof.2014.01.011.
[22] J. Barlow, J.S. Giboney, M.J. Keith, D.W. Wilson, R.M. Schutzler, Overview and guidance on agile development in large organizations. Communications of the Association for Information Systems, Commun. Assoc. Inf. Syst. 29 (1) (2011) 25–44.
[23] S. Jalali, C. Wohlin, Global software engineering and agile practices: a systematic review, J. Softw. Evol. Process 24 (6) (2012) 643–659, https://doi.org/10.1002/smr.
[24] T.Rocha Silva, M. Winckler, C. Bach, Evaluating the usage of predefined interactive behaviors for writing user stories: an empirical study with potential product owners, Cogn. Technol. Work (2019) 1–21, https://doi.org/10.1007/s10111-019-00566-3.
[25] G. Lucassen, F. Dalpiaz, J.M.E.M. van der Werf, S. Brinkkemper, The use and effectiveness of user stories in practice, in: Requirements engineering: Foundation for software quality: 22nd international working conference, REFSQ 2016, 2016, pp. 205–222, https://doi.org/10.1007/978-3-319-30282-9.
[26] E. Dilorenzo, et al., Enabling the reuse of software development assets through a taxonomy for user stories, IEEE Access 8 (2020) 107285–107300, https://doi.org/10.1109/ACCESS.2020.2996951.
[27] H. Ordóñez, A.F.E. Villada, D.L.V. Vanegas, C. Cobos, A. Ordóñez, R. Segovia, An impact study of business process models for requirements elicitation in XP, in: International Conference on Computational Science and Its Applications 9155, Springer Verlag, 2015, pp. 298–312.
[28] P.L. de Souza, A.F. do Prado, W.L. de Souza, S.M. dos Santos Forghieri Pereira, L. F. Pires, Improving agile software development with domain ontologies, Inf. Technol. Gener. 738 (2018) 267–274, https://doi.org/10.1007/978-3-319-77028-4_37.
[29] B. Kitchenham, O.P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering–a systematic literature review, Inf. Softw. Technol. 51 (1) (2009) 7–15.
[30] Merriam-Webster Inc., "Definition of ambiguity," *Merriam-Webster Inc.*, 1828. https://www.merriam-webster.com/dictionary/ambiguity (accessed Sep. 13, 2021).
[31] G. Stella, R. Marsura, A. Messina, S. Rizzo, Capturing user needs for agile software development, in: Proceedings of 4th International Conference in Software Engineering for Defence Applications 422, 2016, pp. 191–202, https://doi.org/10.1007/978-3-319-27896-4.
[32] L.A. Lopes, E.G. Pinheiro, T.S. Da Silva, L.A.M. Zaina, Using UxD artefacts to support the writing of user stories: findings of an empirical study with agile developers,", in: Proc. 19th Int. Conf. Agil. Softw. Dev. Companion*, vol. Part F1477*, 2018, pp. 1–4, https://doi.org/10.1145/3234152.3234158.
[33] L. Müter, T. Deoskar, M. Mathijssen, S. Brinkkemper, F. Dalpiaz, Refinement of user stories into backlog items: linguistic structure and action verbs, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, 2019, pp. 109–116, https://doi.org/10.1007/978-3-030-15538-4_7, vol. 11412 LNCS.
[34] V. Gervasi, A. Ferrari, D. Zowghi, P. Spoletini, Ambiguity in requirements engineering: towards a unifying framework, Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 11865 (2019) 191–210, https://doi.org/10.1007/978-3-030-30985-5_12. LNCS.
[35] D.C. Gause, M. Weinberg, Gerald, Exploring Requirements: Quality before Design, Dorset House Publishing Co., Inc., 353 West 12th Street, New York, NY 10014, 1989.
[36] E. Kamsties, D.M. Berry, and B. Paech, "Detecting ambiguities in requirements documents using inspections," 2001.
[37] F. Chantree, B. Nuseibeh, A. De Roeck, A. Willis, Identifying nocuous ambiguities in natural language requirements, in: 14th IEEE International Requirements Engineering Conference (RE'06, 2006, pp. 59–68, https://doi.org/10.1109/RE.2006.31.
[38] E. Kamsties, B. Peach, Taming ambiguity in natural language requirements, in: Proceedings of the International Conference on System and Software Engineering and their Applications 2, 2000, pp. 1–8.
[39] A. Willis, F. Chantree, A. De Roeck, Automatic identification of nocuous ambiguity, Res. Lang. Comput. 6 (3–4) (2008) 355–374, https://doi.org/10.1007/s11168-008-9058-2.
[40] H. Yang, A. De Roeck, A. Willis, B. Nuseibeh, A methodology for automatic identification of nocuous ambiguity, Coling 2010 - 23rd Int. Conf. Comput. Linguist. Proc. Conf 2 (August) (2010) 1218–1226.
[41] T.R. Silva, J.L. Hak, M. Winckler, A formal ontology for describing interactive behaviors and supporting automated testing on user interfaces, Int. J. Semant. Comput. 11 (4) (2017) 513–539, https://doi.org/10.1142/S1793351X17400219.
[42] M. Elallaoui, K. Nafil, R. Touahni, Automatic transformation of user stories into UML use case diagrams using NLP techniques, Procedia Comput. Sci. 130 (2018) 42–49, https://doi.org/10.1016/j.procs.2018.04.010.
[43] Y. Wautelet, S. Heng, D. Hintea, M. Kolp, S. Poelmans, Bridging user story sets with the use case model,", in: International Conference on Conceptual Modeling, 2016, pp. 127–138, https://doi.org/10.1007/978-3-319-47717-6.
[44] J. Lin, H. Yu, Z. Shen, C. Miao, Using goal net to model user stories in agile software development,", in: 2014 IEEE/ACIS 15th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2014, 2014, pp. 1–6.
[45] F. Wanderley, A. Silva, J. Araujo, D.S. Silveira, SnapMind: a framework to support consistency and validation of model-based requirements in agile development,", in: 2014 IEEE 4th International Model-Driven Requirements Engineering Workshop, MoDRE 2014 - Proceedings, 2014, pp. 47–56, https://doi.org/10.1109/MoDRE.2014.6890825.
[46] Y. Wautelet, D. Gielis, S. Poelmans, S. Heng, Evaluating the impact of user stories quality on the ability to understand and structure requirements,", in: IFIP Working Conference on The Practice of Enterprise Modeling*, vol. 369, Springer International Publishing*, 2019, pp. 3–19.
[47] F. Gilson, C. Irwin, From user stories to use case scenarios towards a generative approach,", in: Proceedings - 25th Australasian Software Engineering Conference, ASWEC 2018, Dec. 2018, pp. 61–65, https://doi.org/10.1109/ASWEC.2018.00016.
[48] R. Giganto, Generating class models through controlled requirements,", in: New Zeal. Comput. Sci. Res. Student Conf. NZCSRSC 2008 - Proc.*, no. April*, 2008, pp. 208–211.
[49] M. Urbieta, L. Antonelli, G. Rossi, J.C.S. do Prado Leite, The impact of using a domain language for an agile requirement management, Inf. Softw. Technol. 127 (November 2019) (2020), 106375, https://doi.org/10.1016/j.infsof.2020.106375.
[50] A. Umber, I.S. Bajwa, Minimizing ambiguity in natural language software requirements specification,", in: 2011 6th Int. Conf. Digit. Inf. Manag. ICDIM 2011, 2011, pp. 102–107, https://doi.org/10.1109/ICDIM.2011.6093363.
[51] J. Polpinij, An ontology-based text processing approach for simplifying ambiguity of requirement specifications,", in: 2009 IEEE Asia-Pacific Serv. Comput. Conf. APSCC 2009, 2009, pp. 219–226, https://doi.org/10.1109/APSCC.2009.5394119.
[52] J. Melegati, X. Wang, QUESt: new practices to represent hypotheses in experiment-driven software development,", in: Proceedings of the 2nd ACM SIGSOFT International Workshop on Software-Intensive Business: start-ups, Platforms, and Ecosystems - IWSiB 2019, 2019, pp. 13–18, https://doi.org/10.1145/3340481.3342732.
[53] C. Agra, A. Sousa, J. Melo, M. Lucena, F. Alencar, Specifying Guidelines to Transform i* Model into User Stories: an Overview,", in: Proceedings of the Eighth International i* Workshop (istar 2015), 2015, pp. 109–114, no. istar.
[54] L.B. De Araujo, F.L. Siqueira, Using i* with scrum: an initial proposal, CEUR Workshop Proc 1674 (2016) 19–24.
[55] T. Tenso, K. Taveter, Requirements engineering with agent-oriented models, in: ENASE 2013 - Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering, 2013, pp. 254–259, https://doi.org/10.5220/0004569302540259.
[56] C. Thamrongchote, W. Vatanawood, Business process ontology for defining user story, in: 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), 2016, pp. 1–4.
[57] T. Tenso, A.H. Norta, H. Rootsi, K. Taveter, I. Vorontsova, Enhancing requirements engineering in agile methodologies by agent-oriented goal models: two empirical case studies,", in: Proceedings - 2017 IEEE 25th International Requirements Engineering Conference Workshops, REW 2017, 2017, pp. 268–275, https://doi.org/10.1109/REW.2017.24.

[58] D. Dermeval, et al., Applications of ontologies in requirements engineering: a systematic review of the literature, Requir. Eng. 21 (4) (2016) 405–437, https://doi.org/10.1007/s00766-015-0222-6.

[59] J. Kocerka, M. Krzeslak, A. Galuszka, Analysing quality of textual requirements using natural language processing: a literature review,", in: 2018 23rd Int. Conf. Methods Model. Autom. Robot. MMAR 2018, 2018, pp. 876–880, https://doi.org/10.1109/MMAR.2018.8486143.

[60] A. Alzayed, A. Al-Hunaiyyan, A Bird's eye view of natural language processing and requirements engineering, Int. J. Adv. Comput. Sci. Appl. 12 (5) (2021) 81–90, https://doi.org/10.14569/IJACSA.2021.0120512.

[61] M. Bano, Addressing the challenges of requirements ambiguity: a review of empirical literature, in: 2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE), Aug. 2015, pp. 21–24, https://doi.org/10.1109/EmpiRE.2015.7431303.

[62] H. Yang, A. de Roeck, V. Gervasi, A. Willis, B. Nuseibeh, Analysing anaphoric ambiguity in natural language requirements, Requir. Eng. 16 (3) (2011) 163–169, https://doi.org/10.1007/s00766-011-0119-y.

[63] H. Yang, A. Willis, A. De Roeck, B. Nuseibeh, Ambiguities in natural language requirements,", in: ASE'10 - Proc. IEEE/ACM Int. Conf. Autom. Softw. Eng.*, no. December 2013*, 2010, pp. 53–62, https://doi.org/10.1145/1858996.1859007.

[64] D.M. Berry, E. Kamsties, Ambiguity in Requirements Specification, Perspect. Softw. Requir. (2004) 7–44, https://doi.org/10.1007/978-1-4615-0465-8_2.

[65] R. Wieringa, N.A.M. Maiden, N.R. Mead, C. Rolland, Requirements engineering paper classification and evaluation criteria: a proposal and a discussion, Requir. Eng. 11 (1) (2006) 102–107, https://doi.org/10.1007/s00766-005-0021-6.

[66] P. Kamthan, N. Shahmir, Effective User Stories are Affective, in: 11th International Conference on Ubiquitous Computing and Ambient Intelligence 2, 2017, pp. 605–611.

[67] T. Avdeenko, M. Murtazina, M. Avdeenko, T. Murtazina, Intelligent Support of Requirements Management in Agile Environment,", in: International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing, 2018, pp. 97–108, https://doi.org/10.1007/978-3-030-03003-2, no. June.

[68] M. Trkman, J. Mendling, P. Trkman, M. Krisper, Impact of the conceptual model's representation format on identifying and understanding user stories, Inf. Softw. Technol. 116 (October 2018) (Dec. 2019), 106169, https://doi.org/10.1016/j.infsof.2019.08.001.

[69] A. Zeaaraoui, Z. Bougroun, M.G. Belkasmi, T. Bouchentouf, M.G. Belkasmi, T. Bouchentouf, User stories template for object-oriented applications,", in: Third International Conference on Innovative Computing Technology (INTECH 2013), Aug. 2013, pp. 407–410, https://doi.org/10.1109/INTECH.2013.6653681.

[70] N. Prakash, D. Prakash, Model-driven user stories for agile data warehouse development,", in: Proceedings - 2017 IEEE 19th Conference on Business Informatics, CBI 2017 1, Aug. 2017, pp. 424–433, https://doi.org/10.1109/CBI.2017.67.

[71] N. Costa, N. Santos, N. Ferreira, R.J. Machado, Delivering user stories for implementing logical software architectures by multiple scrum teams,", in: International Conference on Computational Science and Its Applications*, vol. 8581 LNCS, no. PART 3*, 2014, pp. 747–762.

[72] A. Jaqueira, M. Lucena, E. Aranha, F.M.R. Alencar, J. Castro, E. Aranha, Using i* models to enrich user stories, CEUR Workshop Proc 978 (iStar) (2013) 55–60.

[73] Y. Wautelet, S. Heng, M. Kolp, Perspectives on user story based visual transformations, in: CEUR Workshop Proceedings, 2017, p. 1796.

[74] Y. Wautelet, S. Heng, M. Kolp, I. Mirbel, S. Poelmans, Building a rationale diagram for evaluating user story sets,", in: 2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS), 2016, pp. 1–12.

[75] B. De Brock, "Towards pattern-driven requirements engineering: development patterns for functional requirements," in *Proceedings - 2018 8th International Model-Driven Requirements Engineering Workshop, MoDRE 2018*, 2018, pp. 73–78, doi: 10.1109/MoDRE.2018.00016.

[76] Y. Wautelet, et al., On modelers ability to build a visual diagram from a user story set: a goal- oriented approach,", in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10753 LNCS*, 2018, pp. 209–226.

[77] J. Jia, X. Yang, R. Zhang, X. Liu, Understanding software developers' cognition in agile requirements engineering, Sci. Comput. Program. 178 (Jun. 2019) 1–19, https://doi.org/10.1016/j.scico.2019.03.005.

[78] S.M. Sohan, M.M. Richter, F. Maurer, Auto-tagging Emails with User Stories Using Project Context,", in: International Conference on Agile Software Development*, vol. 48 LNBIP*, 2010, pp. 103–116.

[79] R. Barbosa, A.E.A. Silva, R. Moraes, Use of Similarity Measure to Suggest the Existence of Duplicate User Stories in the Srum Process,", in: Proceedings - 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN-W 2016, Sep. 2016, pp. 2–5, https://doi.org/10.1109/DSN-W.2016.27.

[80] B. Jiang, P. Liu, Y. Wang, Y. Chen, HyOASAM: a Hybrid open API selection approach for mashup development, Math. Probl. Eng. 2020 (1) (2020), https://doi.org/10.1155/2020/4984375.

[81] Y. Wautelet, S. Heng, S. Kiv, M. Kolp, User-story driven development of multi-agent systems: a process fragment for agile methods, Comput. Lang. Syst. Struct. 50 (Dec. 2017) 159–176, https://doi.org/10.1016/j.cl.2017.06.007.