April 2024

The AmbiTRUS Ambiguity Analysis Framework

THIS PAGE LEFT BLANK INTENTIONALLY

KEYWORDS

Agile Software Engineering

Ambiguity

Requirements Engineering

Requirements Quality

Software Engineering

User Stories

CONTENTS

THIS PAGE LEFT BLANK INTENTIONALLY

1. INTRODUCTION

This report is designed to outline the technical details involved in the development of AmbiTRUS (Ambiguity Tracking and Resolution for User Stories) framework. Its objective is to offer comprehensive technical insights, enabling a clear understanding of what AmbiTRUS entails and how it operates.

AmbiTRUS is an ambiguity analysis framework constructed based on the identification and extraction of various manifestations of ambiguity problems in user stories across multiple linguistic levels. Based on *the ambiguity classification framework* (Amna and Poels 2022), AmbiTRUS contrast different manifestations of ambiguity to create a systematic approach for identifying potential ambiguity in user stories.

AmbiTRUS identifies ambiguity by investigating 13 types of ambiguity problems that occur at different linguistic levels. If left unresolved, these problems can lead to a range of problems in requirements quality, such as inconsistency, insufficiency, and duplication. To make the classification framework actionable, AmbiTRUS assigns a user story quality criterion to each type of ambiguity problem. Utilizing AmbiTRUS involves assessing each quality criterion for a set of related user stories.

This report will discuss the methodology used in the development of the AmbiTRUS, the refinement of its quality criteria, and the practical implementation of the AmbiTRUS in analyzing ambiguity in user stories.

2. AmbiTRUS DEVELOPMENT METHODOLOGY

The development process of AmbiTRUS started by searching for examples of ambiguity issues for each type of ambiguity problem identified in the ambiguity classification framework that was proposed in (Amna and Poels 2022). Next, we conceptualized what was needed to avoid or resolve each type of ambiguity problem. We then changed the examples such that they no longer feature the ambiguity issues. By reflecting on these changes, we came up with suggestions of a quality criterion for each type of ambiguity problem.

To define the suggested quality criteria, we reviewed three established user story quality frameworks: the QUS framework (Lucassen et al. 2016), the Agile Requirements Verification framework (Heck and Zaidman 2014), and the INVEST criteria (Wake 2003). The QUS (Quality User Story) framework is designed to identify and remedy defects in user stories to prevent quality issues. This framework consists of 13 quality criteria categorized across syntactic, semantic, and pragmatic levels, where potential defects may arise (Lucassen et al. 2016). Adherence to this framework is essential to ensure the quality of user stories.

The Agile Requirements Verification framework is developed to assess the quality of Agile requirements. The structure of the framework follows the Software Product Certification

Model (SPCM), hence, it incorporates three Verification Criteria (VC), each containing several Specific Criteria (SC) ranging from three to ten (Heck and Zaidman 2014). This framework is employed for assessing the requested features and effectively managing them to deliver higher-quality software products.

The INVEST criteria are a set of quality criteria that are proposed to define and assess the quality of Agile requirements. While this framework does not explicitly mention user stories like the Agile Requirements Verification framework does, these criteria are widely adopted in Agile methodologies, particularly Scrum (Wake 2003).

These three frameworks primarily address ambiguity as a single criterion that simply denotes instances where a user story could be interpreted in various ways. In contrast, the ambiguity classification framework of Amna and Poels (2022) defines ambiguity as being manifested in diverse situations that are characterized by different interpretations and unclear intentions, which differs from the singular understanding of ambiguity in the reviewed frameworks. Nevertheless, we observed that several other user story quality criteria from those established frameworks are related to user story ambiguity as conceptualized in the ambiguity classification framework. These criteria involve user story requirements such as being atomic, conceptually sound, and unique, which we could relate to specific types of ambiguity problems in the classification framework.

Table 1. The AmbiTRUS ambiguity analysis framework

	Linguistic level			
Req. Quality	Lexical	Syntactic	Semantic	Pragmatic
Clarity	Complete term. Precise term.	Atomic	Representative	Explicit statement
Consistency	Consistent terminology	No-overlapping	Conflict-free*	Combined understanding
Adequacy	N/A**	N/A**	Directive	Contextual relevancy
Uniqueness	No synonyms	N/A**	Duplication-free (well-modularized)	N/A**

* Occurs during user story transformation into conceptual models

** Existing research has not found

In the next sub-sections, we present for each linguistic level of ambiguity the quality criteria that we defined for AmbiTRUS (see Table 1). We motivate the definition of the user story quality criteria, explain how they relate to the quality criteria of the reviewed user story quality frameworks, and provide examples and anti-examples of user stories that respectively satisfy or violate the criteria. An overview of the full set of examples used will be presented in section 3.

2.1 Quality Criteria for Lexical Ambiguity

At the lexical level, quality criteria are intended to ensure the completeness, consistency, and preciseness of the terms used to describe actors, actions, and objects in the *WHO*,

WHAT, and *WHY* segments of the user stories. Fulfilling these criteria ensures that each user story features one commonly used, verifiable, and consistently denoted actor in the *WHO* segment, and one clearly expressed action executed or supported on or for an explicitly defined and uniquely referred-to object that is maintained or controlled by the system in the *WHAT* and *WHY* segments (Gupta et al. 2023).

2.1.1 Complete Terminology

Complete terminology is a criterion that recommends that a user story be written using terminology that accurately represents the concept. This means using domain-specific terminology to denote the actor in the ***WHO*** segment and the intended action in the *WHAT* segment without spelling or grammar errors, abbreviations or jargon to avoid different interpretations (Urbieta et al. 2020). Based on the Agile Requirements Verification Framework, this criterion refers to the Verification Criteria (VC3), Consistency and Correctness, in particular the Specific Criteria (SC3.3) titled *correct language* (Heck and Zaidman 2014). The objective of this criterion is to avoid homonymy and polysemy due to inherent ambiguity in the contextual etymology of a user story (Berry and Kamsties 2004).

This criterion is violated when partial names or terms are used to describe the actor in the *WHO* segment of the user story and the action or the object in the *WHAT* and *WHY* segments. Table 2 illustrates an example of a user story that breaches this criterion (see $US1_a$) and a related example of how the user story can be changed to remove the ambiguity issues (see $US1_b$).

Table 2. An illustration of how incomplete terminology can trigger lexical ambiguity in a user story

$US1_a$	*"As an administrator, I want to change the profile, so that the page will appear."*
$US1_b$	*"As a site administrator, I want to change the company profile, so that the profile page will appear."*

The user story $US1_a$ exhibits potential ambiguity due to several reasons. Firstly, the actor labeled *"administrator"* lacks specificity about the particular role or responsibilities. Secondly, the action described as *"the profile"* lacks specificity, leaving room for interpretations of the intended profile. Lastly, the desired outcome of displaying *"the page"* lacks precise information about what kind of page should be presented, making the objective unclear.

The issues observed in user story $US1_a$ have been addressed in the revised user story $US1_b$. Notably, it refines the specification of the actor as *"site administrator"* instead of the more general *"administrator,"* offering a more explicit understanding of the assigned responsibilities. In addition, the description of the intended action in $US1_b$ has been improved to facilitate a better understanding of the intended goal. Furthermore, enhancing the benefit in $US1_b$ contributes to a clearer motivation for executing the user story.

2.1.2 Precise Terminology

Precise terminology emphasizes the use of commonly used and verifiable terminology that accurately portrays the actor, action, and motivation within the user story (Müter et al. 2019; de Souza et al. 2018; Urbieta et al. 2020). This criterion aligns with SC3.5 in the Agile Requirements Quality Framework, which specifies that requirements artifacts express verifiable feature requests and should be ***as precise as possible*** (Heck and Zaidman 2014). To minimize wide-ranging interpretations that could lead to differing interpretations, standard glossaries can be constructed as a unique source of the terms to be used in user stories (Müter et al. 2019).

Violations of this criterion manifest differently based on where in the user story too general terms are employed. In the *WHO* segment, a breach occurs when general terms describe the actor without recognizing the unique domain context, hence assuming similarity across different domains. In the *WHAT* and *WHY* segments, violations may arise from the use of broad terminology, resulting in different interpretations of action execution.

To illustrate how a user story can violate the precise terminology quality criterion, we use user story $US1_b$, in which the verb *"change"* in the phrase *"change the company profile"* lacks clarity regarding the necessary modifications. Additionally, the benefit in the *WHY* segment remains unclear. The improved $US1_c$ version emphasizes the need for precise terminology to clarify *the required modifications for the company profile* and outline the expected feature or behavior.

Table 3. An illustration of how imprecise terminology can trigger lexical ambiguity in a user story

$US1_b$	*"As a site administrator, I want to change the company profile, so that the profile page will appear."*
$US1_c$	*"As a site administrator, I want to modify the content of the company profile, so that I can view the page with new company activities and achievements."*

2.1.3 Consistent Terminology

Consistent terminology is a quality criterion demanding the use of identical terms to represent the same actor, action, or object across related user stories. This criterion, SC3.8 titled "Glossary", is part of the Agile Requirements Verification Framework (Heck and Zaidman 2014). They emphasize the importance of maintaining project-specific glossaries and abbreviations to ensure the consistency and correctness of agile requirements artifacts. Table 4 shows a comprehensive standard glossary to express actions in user stories, taken from (Müter et al. 2019). Similarly, (Urbieta et al. 2020) promoted the use of domain lexicons to express actors and behavioral responses. Violations of this criterion include the use of different terminology in related user stories to express the same actor in the *WHO* segment or similar behavior in the *WHAT* and *WHY* segments. Table 5 illustrates a breach of the consistent terminology quality criterion.

Table 4. The list of the standard terminology representing the class of actions, later referred to as the "standard glossary," is used in this paper (Müter et al. 2019)

Keyword Class	Keyword Item
Create	add, insert, create, make, build, develop, establish, generate, construct, invite
Read	view, read, display, show, retrieve, get, access, examine, browse
Update	modify, edit, change, update, revise, alter, adjust, adapt, refine, fix, improve, renew, replace
Delete	remove, delete, erase, clear, eliminate, exclude, discard, purge, drop
Merge	bind, export, integrate, link, list, offer
Validate	check, evaluate, test, verify
Search	investigate, inquire, research, search

Table 5. An illustration of how inconsistent terminology can trigger lexical ambiguity in a user story

$US2_a$	*"As a website administrator, I want to request my colleagues to administer the company profile, so that I can share responsibilities with them."*
$US2_b$	*"As a site administrator, I want to create access to my colleagues, so that they can update the content of the company profile as well."*

User story $US2_a$ displays inconsistent terminology by using *"website administrator"* to refer to an individual who has the privilege of a *"site administrator"* (see user story $US1_b$). Moreover, the term *request* in the *WHAT* segment is not included in the standard glossary of Table 5, causing potential variations in action execution. To mitigate these concerns, *US2b* improves user story *US2a* by removing the ambiguity issues related to the use of inconsistent terminology. However, the replacement of problematic terms may not be sufficient to avoid the ambiguity. Therefore, it is recommended to rephrase the user stories by including the keyword class of the standard glossary in the improved version of a user story.

2.1.4 No Synonyms

No synonyms is a quality criterion inspired by Barbosa et al. (Barbosa et al. 2016), who compared several algorithms to find similarities between user stories. The violation of this criterion manifests in the use of different terms having a similar meaning in other user stories from the same set of user stories.

Table 6 displays an example of a violation of this criterion and the improved version of the user story. The actor in user story $US3_a$ is referred to as *"webmaster"* while it is "site administrator" in $US1_b$, $US1_c$, and $US2_b$. Moreover, to maintain consistency and prevent duplications, replacing the term *"grant"* with the standard glossary term "create" is recommended (see Table 6). To comply with the criterion, the user story $US1_a$ is improved in user story $US1_b$.

Table 6. An illustration of how synonyms can trigger lexical ambiguity in a user story

$US3_a$	*"As a webmaster, I want to grant access to the administrator page to my colleagues, so that they can manage the content of the company profile as well."*
$US3_b$	*"As a site administrator, I want to create access to the administrator page for my colleagues, so that they can manage the content of the company profile as well."*

2.2 Quality Criteria for Syntactic Ambiguity

These criteria specify rules that should be followed to avoid syntactic ambiguity in user stories. Accordingly, a user story should be divided into segments (i.e., *WHO, WHAT, WHY*) and checked for well-formedness and atomicity. These criteria are adopted from the QUS framework (Lucassen et al. 2016).

2.2.1 Atomic

Atomic is a quality criterion that requires a user story to be a simple sentence, comprising one actor and one feature in terms of a single action imposed on a particular object. To prevent user stories from generating multiple parse-trees which would result in different interpretations, it is recommended to avoid using compound and complex sentences in user stories (Elallaoui et al. 2018; Gervasi et al. 2019). This practice helps to ensure the fulfillment of the *"Small"* criterion in INVEST (Wake 2003) and the *"Minimal"* criterion in QUS (Lucassen et al. 2016). In the Agile Requirements Verification Framework, this criterion refers to the SC3.7 titled *Atomic* (Heck and Zaidman 2014). According to this criterion, one user story must describe only one action request.

Violations linked to the atomic criterion arise from conjunctions that introduce different interpretations of a user story. Breaches in compliance rarely occur within the *WHO* segment but are prevalent in the *WHAT* and *WHY* segments. To determine whether these conjunctions might cause ambiguity requires examining both the action and its underlying motivation. A breach of the atomic criterion occurs when multiple actions and objects are described in the *WHAT* segment, each intended to achieve distinct benefits. Table 7 provides an example illustrating a breach of the atomic criterion and demonstrates how the improved version should be structured.

Table 7 highlights a potential violation of the atomic criterion in user story $US4_a$ due to the conjunction used in the *WHAT* segment. The use of the conjunction *"and"* to separate *"news"* and *"certificates"* in the phrase *"update the news and delete expired certificates"* introduces multiple scenarios as depicted in user stories $US4_b$ and $US4_c$. Further, the conjunction *"and"* in the phrase *"delete expired certificate and achievements"* introduces uncertainty regarding how *"achievement"* should be treated, whether it should be *"updated"* or *"deleted"*, as demonstrated by user stories $US4_d$ and $US4_e$.

Table 7. An illustration of how non-atomicity can trigger syntactic ambiguity in a user story

$US4_a$	*"As a site administrator, I want to update the news and delete expired certificates and achievements on the company profile website, so that people can see the news, certificates, and achievements."*
$US4_b$	*"As a site administrator, I want to update the news on the company profile website, so that people can see the updated news."*
$US4_c$	*"As a site administrator, I want to delete expired certificates from the company profile website, so that people can see still valid certificates."*
$US4_d$	*"As a site administrator, I want to update achievements on the company profile website, so that people can see the updated achievements."*
$US4_e$	*"As a site administrator, I want to delete achievements from the company profile website, so that people can see the current achievements."*

2.2.2 No Overlapping

No overlapping is a quality criterion that advocates that the actor of the WHO segment and the action described in the WHAT segment should avoid any explicit condition that is required to be met by a user story to be executed. When the user story needs a specific condition to perform the user story, it is recommended to explicitly describe this condition in the WHY segment as a part of user story motivation. While there is no particular reason on why this structure should be implemented, it is revealed that the use of a subordinate conjunction to conceive the action described in the WHAT segment might result in syntactic ambiguity, therefore it is recommended to avoid it (Berry and Kamsties 2004).

Within *the Agile Requirements Verification Framework*, this criterion corresponds to SMART/INVEST, which is specified under SC3.5 (Heck and Zaidman 2014). This criterion aligns with verifiable quality expected in Software Requirements Specification (SRS) (Davis et al. 2011). Although there is debate about the feasibility of simultaneously checking all *SMART (Specific, Measurable, Acceptable, Realistic, Time-bound)* criteria, the essence of this criterion remains relevant. In addition, despite potential skepticism about the *"no-overlapping"* criterion, the intention of this criterion is to ensure conciseness to verify the consistency of a user story.

Table 8. An illustration of how poor structure (complicated) user story can trigger syntactic ambiguity

$US5_a$	*"As a site administrator, I want to delete certificates after a certain period, so that people can see the still valid certificates."*
$US5_b$	*"As a site administrator, I want to delete expired certificates, so that people can see the still valid certificates."*

Table 8 highlights potential ambiguity in user story $US5_a$, particularly due to the phrase *"after a certain period"* in the *WHAT* segment, which has the potential to be interpreted differently among individuals. The *WHY* segment talks about "seeing still valid certificates" which is not automatically consistent with "deleting certificates after a certain period". To enhance the consistency of the user story, it was changed to $US5_b$.

2.3 Quality Criteria for Semantic Ambiguity

These quality criteria require that user stories convey the intended meaning of the required feature concisely, provide a conceptual understanding of what is required (so do not strictly describe the technical implementation), be free from conflict with other user stories, and have a unique understanding of one another (Lucassen et al. 2016). The analysis requires a set of user stories from the same domain treated as a whole unit. A user story should be regarded as a complete sentence, even though it needs to be chunked (i.e., split the user story into *WHO, WHAT, WHY* segments).

2.3.1 Representative

Representative is a quality criterion that suggests that user stories avoid including extra information and hints that can be expressed through conjunctions and punctuation marks (e.g., commas, colons, semi-colons, and parentheses). Even though this problem is syntactic, Berry and Kamsties (Berry and Kamsties 2004) classified this additional information and hints as semantic problems due to their potential to introduce multiple interpretations of a user story within its context. Ensuring representativeness of user stories involves identifying punctuation marks and parentheses that often serve as unnecessary information and hints.

In the QUS framework, it is also recommended for user stories to avoid extra information specified in the sentence. Besides breaking the *minimal* criterion, additional information could add complexity to user story specifications (Lucassen et al. 2016). Therefore, it is recommended to express a user story as concisely as possible.

Table 9. An illustration of how a verbose user story can trigger semantic ambiguity

$US6_a$	*"As a site administrator, I want to delete certificates that are obtained from suspicious vendors, indicated by "suspicious vendors" list in our database, so that people can only see legitimate certificates."*
$US6_b$	*"As a site administrator, I want to delete suspicious certificates, so that people can only see legitimate certificates."*

In Table 9, user story $US6_a$ is potentially ambiguous due to discrepancies in the *WHAT* and *WHY* segments. The sentence in the *WHAT* segment, *"certificate that are obtained from suspicious vendors, indicated by "suspicious vendors" list in our database"* contains excessive details that might restrict flexibility in the transformation of the user story into a

system feature. To address this problem, we adjust the user story $US6_a$ by removing the phrase *"indicated by "suspicious vendors" list in our database"* from the *WHAT* segment, refining it to *"suspicious certificates"* (see user story $US6_b$).

2.3.2 Conflict-free

Conflict-free is a quality criterion that requires that user stories avoid conflict with other user stories within the same domain. This criterion is accommodated in the Agile Requirements Verification Framework under the *"No contradiction"* criterion (SC3.2) (Heck and Zaidman 2014), while the QUS framework addresses this criterion as *"Conflict-free"* (Lucassen et al. 2016). While the QUS framework (Lucassen et al. 2016) refers to a conflict as a situation when two or more user stories aim to achieve the same benefits using different actions, further confirmation is required to determine whether this situation refers to a variation that is requested in user stories or a conflicting situation.

Table 10. An illustration of conflicted user stories can trigger semantic ambiguity in user stories

$US4_c$	*"As a site administrator, I want to delete expired certificates from the company profile website, so that people can see still valid certificates."*
US6	*"As a public relation manager, I want to update expired certificates on the company profile website, so that people can see still valid certificates."*

Table 10 highlights two distinct user stories, $US4_c$ and *US6* which could potentially conflict due to different actors aiming to achieve identical benefits in the *WHY* segment. However, these potentially conflicting situations need to be manually confirmed via discussion before implementation starts.

2.3.3 Directive

Directive is a quality criterion adopted from the QUS framework under the name of *conceptually sound* (Lucassen et al. 2016). This criterion requires user stories to describe a concrete action applied to a specific object in the *WHAT* segment and provide a clear motivation for this in the *WHY* segment. Violations of this criterion manifest themselves as underspecified action or motivation in the *WHAT* and *WHY* segments (De Brock 2018; Gupta et al. 2023; Urbieta et al. 2022). An example of the violation and how this problem should be avoided is presented in Table 11.

Table 11. An illustration of solution-oriented user stories can trigger semantic ambiguity in user stories

$US7_a$	*"As a site administrator, I want to be able to change advertisements via setup menu, so that I can target my ads to the most relevant visitors and maximize my ad effectiveness."*
$US7_b$	*"As a site administrator, I want to update a particular advertisement, so that I can set up the time period for frequent advertisement for a particular event in social media."*

In Table 11, user story *US7$_a$* indicates ambiguity due to an unclear action as there is too much detail of the implementation in the *WHAT* segment but insufficient reasoning in the *WHY* segment. To enhance clarity and consistency, it is essential to specify the intended action in the *WHAT* segment while removing the solution implementation details to maintain the flexibility on how the action should be implemented. This involves substituting *"change advertisements"* with *"update a particular advertisement"* using standard glossary (see Table 11). Further, we clarify that the *particular advertisement* that can be updated in the *WHAT* segment is specifically applied to the advertisement related to *a particular event* by explicitly mentioning this condition in the *WHY* segment. To give flexibility to the user story implementation, we also remove the solution *"via setup menu"* from the *WHAT* segment. The revised version, *US7$_b$*, provides a more directive and clarified representation.

2.3.4 Duplication-free (well-modularized)

Duplication-free (well modularized) is a quality criterion that suggests that user stories have a unique interpretation concerning one another. The QUS framework defines this criterion as unique and conflict-free (Lucassen et al. 2016), while the Agile Requirements Verification Framework defines this criterion as a distinct quality criterion, titled *no duplication*, which is specified in SC3.9 (Heck and Zaidman 2014). Violations of this criterion arise when user stories use different wordings but imply the same actions and aim for similar benefits. An example of this violation can be found in *US8$_a$* in Table 12 as follows.

Table 12. An illustration of redundant user stories can trigger semantic ambiguity

US7$_b$	*"As a site administrator, I want to update viewing options on the company profile website, so that I can set up the time period for frequent advertisement on a particular event in social media."*
US8$_a$	*"As a site administrator, I want to customize viewing preferences on the company profile website, so that I can set up the schedule of my advertisement in social media."*

US7$_b$ and *US8$_a$* in Table 12 display duplication. This duplication arises because both user stories potentially refer to identical actions and benefits, despite being distinctly described.

2.4 Quality Criteria for Pragmatic Ambiguity

These quality criteria require user stories to be formulated such that people from different backgrounds have a consistent understanding of user stories (Lindland et al. 1994; Lucassen et al. 2016). The criteria that we suggest comprise explicit statement, combined understanding and contextual relevancy, which we match to respectively the implicit dependency, interpretation diversity, and insufficient context types of ambiguity problems in the ambiguity classification framework of Amna and Poels (Amna and Poels 2022). Upon the comparison with the QUS framework (Lucassen et al. 2016) and the Agile

Requirements Verification Framework (Heck and Zaidman 2014), it is found that similar concepts are present in both frameworks.

2.4.1 Explicit Statement

Explicit statement is a criterion that is constructed based on the studies of Ordóñez et al. (Ordóñez et al. 2015) and Trkman et al. (Trkman et al. 2019), that compared user story templates and conceptual models. Their studies found that although user stories are good at representing high-level requirements, they can only implicitly describe the dependency among user stories.

This definition aligns with the *"specify problem"* criterion in the Agile Requirements Verification Framework (SC3.4) (Heck and Zaidman 2014). A similar concept is also present in the *"independent"* criterion described in the QUS framework (Lucassen et al. 2016). An example of a violation of the explicit statement criterion and how the user story should be improved can be found in Table 13.

Table 13. An illustration of an implicit dependency between user stories that can trigger pragmatic ambiguity

$US9_a$	*"As a site administrator, I want to optimize the company profile website, so that I can coordinate promotional posts during high-traffic periods on social media."*
$US9_b$	*"As a site administrator, I want to create a scheduling feature on the company profile website, so that I can modify the schedule of my advertisements during high-traffic periods in social media."*
$US9_c$	*"As a site administrator, I want to merge analytic tools into the company profile website, so that I can track traffic patterns and identify peak periods to promote in social media."*

In user story $US9_a$, the phrase *"optimize the company profile website"* in the *WHAT* segment might imply various actions, including implementing a scheduling tool or embedding analytical tools. These actions might suggest different desired benefits outlined in the *WHY* segment. Although the end goal is similar, to enhance social media promotion during peak sessions, these varied interpretations could lead to confusion. Thus, it is advisable to revise the user story $US9_a$ to $US9_b$ or $US9_c$ to explicitly describe the intended objective.

2.4.2 Combined Understanding

Combined understanding is a criterion that requires user stories to follow a consistent template to prevent different interpretations when they are implemented. In the Agile Requirements Verification framework, this criterion shares some similarity with the *navigable links* criterion in SC3.10, in an attempt to maintain a consistent understanding between individuals (Heck and Zaidman 2014). Meanwhile, the QUS framework define

this concept as *uniform* focusing on maintaining a consistent template to describe user stories (Lucassen et al. 2016).

Violations of this criterion arise when a set of user stories are described in different templates which could potentially interpreted differently by different team members (Rocha Silva et al. 2019), or when the action in the *WHAT* segment is not linked to a particular motivation that is referred to in the *WHY* segment in another user story. An example of these violations can be found in $US10_a$ in Table 14 as follows.

Table 14. An illustration of interpretation diversity in user stories that can trigger pragmatic ambiguity

$US10_a$	*"As a site administrator, I can add certificates to the company profile website in order to delete another certificate."*
$US10_b$	*"As a site administrator, I want to add new certificates to the company profile website, so that I can delete another certificate."*
$US10_c$	*"As a site administrator, I want to add new certificates to the company profile website, so that people can see our industry certifications on the website."*

Not adhering to the Connextra template in user story $US10_a$ causes complications between the expected action in the *WHAT* segment and the corresponding benefit in the *WHY* segment. The attempt to improve user story $US10_a$ by rewriting the user story into the Connextra template structure, as in user story $US10_b$, does not remove the inconsistency between the action in the *WHAT* segment and the associated benefit in the *WHY* segment. Therefore, to avoid potential conflict between the expected action in the *WHAT* segment and the corresponding benefit in the *WHY* segment, the user story $US10_b$ might be improved into $US10_c$. However, some people might perceive that this change is not sufficient as the expected action in the *WHAT* segment "*add new certificates*" could potentially resemble to the action of "*update*" in the *WHAT* segment of user story $US4_c$, while another person might believe that the actions in both user stories are different. Therefore, to avoid different interpretations, the formulation of user story $US10_c$ should consider both the adoption of a consistent template and the alignment of the expected benefits across a set of related user stories.

2.4.3 Contextual Relevancy

Contextual relevancy is a criterion that requires user stories to adequately represent clear features when they are implemented. This criterion is derived from Melegati and Wang (Melegati and Wang 2019), sharing similarities with the *complete* criterion in the QUS framework (Lucassen et al. 2016). Nevertheless, no alignment was found with the Agile Requirements Verification framework, resulting in limited understanding of the concept. Consequently, we referenced an example of Melegati and Wang (Melegati and Wang 2019) to illustrate how the problematic user story could be improved to meet the criterion.

Table 15. An illustration of insufficient context in a user story that can trigger pragmatic ambiguity

$US12_a$	*"As a youngster, I want to search for activities nearby related to my interests, so that I can enjoy them."*
$US12_b$	*"As an app user, I want to search relevant activities near my location based on my interests, so that I can get map directions to navigate me to the location."*

In Table 15, $US12_a$ has clearly described the actions in the *WHAT* segment. Yet, the *WHO* segment lacks contextual relevance in describing the system role, along with vague benefits mentioned in the *WHY* segment. The user story $US12_a$ can be improved by specifying roles, such as *app user* or *site user,* and elaborating on the expected benefits. While unclear benefits, such as *enjoy* are stated, it is crucial to explicitly define the expected features that a user story should deliver. This clarity is evident in $US12_b$, which describes *"get map directions to navigate me to the location"*.

3. REFINING THE AmbiTRUS CRITERIA TO ANALYZE AMBIGUITY IN USER STORIES

AmbiTRUS was carefully designed by defining a quality criterion for each ambiguity problem in the ambiguity classification framework, mirroring the structure of that framework, and taking most of the quality criteria from existing user story quality frameworks. However, the evaluation of the framework in a laboratory experiment did not yield conclusive evidence of its effectiveness. While the study indicated weak evidence of perceived usefulness from the experiment's participants, the presentation of the framework was deemed less usable. As a result, it becomes imperative to revise the framework criteria, with a focus on refining terminologies and descriptions to enhance comprehension, particularly within limited exposure times.

It is evident that the treatment group performed better for some criteria but fell behind the control group for others. Some studies suggest that this discrepancy might be due to the manual review employed in the experiment, which is known to be error-prone and time-consuming (Ferrari et al. 2018; Riaz et al. 2019). This might explain why for some criteria the intuitive explanations provided by the control group participants were better than those provided by the treatment group participants.

Given this situation, after the experiment, we collaborated with linguistic experts to ensure that our criteria and their corresponding definitions accurately represent the fundamental requirements for unambiguous user stories. We revisited the definition of each criterion and discussed how the criteria names should be formulated, resulting in changes in almost all criteria names. We decided to remove the 'No synonym' criterion as this type of ambiguity problem is mainly occurring at the semantic level of ambiguity (Amna and Poels 2022; Barbosa et al. 2016; Fantechi et al. 2023).

An overview of the changes is shown in Table 15. The redesigned AmbiTRUS ambiguity analysis framework is presented in Table 16. The criteria definitions are presented in Table 17.

Table 16. The modification of the AmbiTRUS criteria

Req. Quality	Linguistic level	Initial criterion	Redefined criterion
Clarity	Lexical	Complete terminology Precise terminology	Precise terminology
	Syntactic	Atomic	Atomicity
	Semantic	Representative	Conciseness
	Pragmatic	Explicit statement	Well-understood
Consistency	Lexical	Consistent terminology	Consistent terminology
	Syntactic	No-overlapping	Well-formedness
	Semantic	Conflict-free*	Conceptually sound*
	Pragmatic	Combined understanding	Objective
Adequacy	Lexical	N/A**	N/A**
	Syntactic	N/A**	N/A**
	Semantic	Directive	Conceptually sound
	Pragmatic	Contextual relevancy	Supportive
Uniqueness	Lexical	No synonym	N/A***
	Syntactic	N/A**	N/A**
	Semantic	Duplication-free (well-modularized)	Uniqueness
	Pragmatic	N/A**	N/A**

* Occurs during user story transformation to conceptual models
** Existing research does not found
*** Moved to the semantic level and merged with 'Duplication-free' into 'Uniqueness'

Table 17. The final version of the ambiguity analysis framework (later referred to as QUAS framework)

	Linguistic level			
Req. Quality	Lexical	Syntactic	Semantic	Pragmatic
Clarity	Precise terminology	Atomicity	Conciseness	Well-understood
Consistency	Consistent terminology	Well-formedness	Conceptually sound*	Objective
Adequacy	N/A**	N/A**	Conceptually sound	Supportive
Uniqueness	Extend into semantic***	N/A**	Uniqueness	N/A**

* Occurs during user story transformation into conceptual models
** Existing research has not found
*** Moved to the semantic level and merged with 'Duplication-free' into 'Uniqueness'

Table 18. Definition of the AmbiTRUS criteria

Lexical	
Precise terminology	User stories should use the unique domain context to describe the actor in the *WHO* segment, or standard terms that are provided in the glossary to represent action in the *WHAT* and *WHY* segments to avoid different interpretations of action execution.
Consistent terminology	User stories should use the same actor in the *WHO* segment or similar behavior in the *WHAT* and *WHY* segments.
Syntactic	

Atomicity	User stories should describe one action and object in the *WHAT* segment to avoid multiple benefits that are intended to achieve.
Well-formedness	A user story should use a consistent template or contain a complete segment of *WHO*, *WHAT*, and *WHY*.
Semantic	
Conciseness	User stories should contain concise and clear information to express what the system should deliver.
Conceptually sound*	User stories should express clear information in order to be able to deliver into system architecture.
Conceptually sound	User stories should express clear information about the expected action which can be interpreted as a single operational activity or system functionality.
Uniqueness	A user story should describe a unique functionality that cannot be found in any other user stories.
Pragmatic	
Well-understood	User stories should not describe implicit dependencies which may go unnoticed.
Objective	User stories should be described in a uniform template to prevent different interpretation by different team members, or if the action in the *WHAT* segment is linked to a specific motivation that is referred to in the *WHY* segment of another user story, it should be described using standard terminology.
Supportive	User stories should describe sufficient information related to the context and intended system.

4. CONCLUSIONS

The design of AmbiTRUS mirrors the structure of the ambiguity classification framework of Amna and Poels (2022). For each type of ambiguity problem described in the framework, a user story quality criterion was defined that when violated signifies an ambiguity issue. These quality criteria were carefully drawn from existing requirements and user story quality frameworks such as the QUS framework (Lucassen et al. 2016), the Agile Requirements Verification framework (Heck and Zaidman 2014), and the INVEST criteria (Wake 2003). To evaluate the effectiveness and usability of the AmbiTRUS framework, a controlled experiment was conducted.

5. FUTURE WORK

We believe that the paper-based presentation and use of the framework draw attention to the complexity of its operationalization. We recognize the drawbacks of a manual review of the quality criteria, which could introduce errors and may not accurately reflect real-world scenarios. To address these challenges and enhance the practicality of the framework, we propose the development of a tool-based method for operationalizing the framework.

Several studies suggest the application of NLP techniques to enhance the performance of identifying and correcting ambiguous requirements (Koh and Chua 2023; Riaz et al. 2019). Based on this insight, we intend to develop a semi-automatic tool that employs NLP

techniques to more effectively operationalize AmbiTRUS. As an extension of our research, the tool could further benefit users by providing recommendations aimed at enhancing the writing of user stories. This feature would contribute to more effective user story analysis and positive impacts on the outcome of requirements elicitations and analysis activities. Further, we propose the implementation of an automatic calculation to evaluate the alignment of recommendations with user opinions.

6. REFERENCES

Amna, A. R., and Poels, G. 2022. “Ambiguity in User Stories : A Systematic Literature Review,” *Information and Software Technology* (145:January), Elsevier B.V., p. 106824. (https://doi.org/10.1016/j.infsof.2022.106824).

Barbosa, R., Silva, A. E. A., and Moraes, R. 2016. “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*, Institute of Electrical and Electronics Engineers Inc., September 22, pp. 2–5. (https://doi.org/10.1109/DSN-W.2016.27).

Berry, D. M., and Kamsties, E. 2004. “Ambiguity in Requirements Specification,” *Perspectives on Software Requirements*, pp. 7–44. (https://doi.org/10.1007/978-1-4615-0465-8_2).

De Brock, B. 2018. “Towards Pattern-Driven Requirements Engineering: Development Patterns for Functional Requirements,” in *Proceedings - 2018 8th International Model-Driven Requirements Engineering Workshop, MoDRE 2018*, IEEE, pp. 73–78. (https://doi.org/10.1109/MoDRE.2018.00016).

Davis, A., Overmyer, S., Jordan, K., Caruso, J., Dandashi, F., Dinh, A., Kincaid, G., Ledeboer, G., Reynolds, P., Sitaram, P., Ta, A., and Theofanos, M. 2011. “Identifying and Measuring Quality in a Software Requirements Specification,” *Software Requirements Engineering*, pp. 194–205. (https://doi.org/10.1109/9781118156674.ch3).

Elallaoui, M., Nafil, K., and Touahni, R. 2018. “Automatic Transformation of User Stories into UML Use Case Diagrams Using NLP Techniques,” in *Procedia Computer Science* (Vol. 130), Elsevier B.V., pp. 42–49. (https://doi.org/10.1016/j.procs.2018.04.010).

Fantechi, A., Gnesi, S., and Semini, L. 2023. “VIBE: Looking for Variability In Ambiguous Requirements,” *Journal of Systems and Software* (195), Elsevier Inc., p. 111540. (https://doi.org/10.1016/j.jss.2022.111540).

Ferrari, A., Gori, G., Rosadini, B., Trotta, I., Bacherini, S., Fantechi, A., and Gnesi, S. 2018. “Detecting Requirements Defects with NLP Patterns: An Industrial Experience in the Railway Domain,” *Empirical Software Engineering* (23:6), pp. 3684–3733. (https://doi.org/10.1007/s10664-018-9596-7).

Gervasi, V., Ferrari, A., Zowghi, D., and Spoletini, P. 2019. “Ambiguity in Requirements Engineering: Towards a Unifying Framework,” *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (11865 LNCS), pp. 191–210. (https://doi.org/10.1007/978-3-030-30985-5_12).

Gupta, A., Poels, G., and Bera, P. 2023. “Generating Multiple Conceptual Models from Behavior-Driven Development Scenarios,” *Data & Knowledge Engineering* (145), p. 102141. (https://doi.org/https://doi.org/10.1016/j.datak.2023.102141).

Heck, P., and Zaidman, A. 2014. *A Quality Framework for Agile Requirements: A Practitioner’s Perspective*. (http://arxiv.org/abs/1406.4692).

Koh, S. J., and Chua, F. F. 2023. “ReqGo: A Semi-Automated Requirements Management Tool,” *International Journal of Technology* (14:4), pp. 713–723. (https://doi.org/10.14716/ijtech.v14i4.5631).

Lindland, O. I., Sindre, G., and Sølvberg, A. 1994. “Understanding Quality in Conceptual Modeling,” *IEEE Software* (11:2), pp. 42–49.

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. M., and Brinkkemper, S. 2016. “Improving Agile Requirements: The Quality User Story Framework and Tool,” *Requirements Engineering* (21:3), Springer London, pp. 383–403. (https://doi.org/10.1007/s00766-016-0250-x).

Melegati, J., and Wang, X. 2019. “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*, New York, New York, USA: ACM Press, pp. 13–18. (https://doi.org/10.1145/3340481.3342732).

Müter, L., Deoskar, T., Mathijssen, M., Brinkkemper, S., and Dalpiaz, F. 2019. “Refinement of User Stories into Backlog Items: Linguistic Structure and Action Verbs,” in *International Working Conference on Requirements Engineering: Foundation for Software Quality* (Vol. 11412 LNCS), pp. 109–116. (https://doi.org/10.1007/978-3-030-15538-4_7).

Ordóñez, H., Villada, A. F. E., Vanegas, D. L. V., Cobos, C., Ordóñez, A., and Segovia, R. 2015. “An Impact Study of Business Process Models for Requirements Elicitation in XP,” in *International Conference on Computational Science and Its Applications* (Vol. 9155), Springer Verlag, pp. 298–312. (https://doi.org/10.1007/978-3-319-21404-7_22).

Riaz, M. Q., Butt, W. H., and Rehman, S. 2019. “Automatic Detection of Ambiguous Software Requirements: An Insight,” *5th International Conference on Information Management, ICIM 2019*, IEEE, pp. 1–6. (https://doi.org/10.1109/INFOMAN.2019.8714682).

Rocha Silva, T., Winckler, M., and Bach, C. 2019. “Evaluating the Usage of Predefined Interactive Behaviors for Writing User Stories: An Empirical Study with Potential Product Owners,” *Cognition, Technology and Work,* Springer London, pp. 1–21. (https://doi.org/10.1007/s10111-019-00566-3).

de Souza, P. L., do Prado, A. F., de Souza, W. L., dos Santos Forghieri Pereira, S. M., and Pires, L. F. 2018. “Improving Agile Software Development with Domain Ontologies,” *Information Technology-New Generations* (738), pp. 267–274. (https://doi.org/10.1007/978-3-319-77028-4_37).

Trkman, M., Mendling, J., Trkman, P., and Krisper, M. 2019. “Impact of the Conceptual Model’s Representation Format on Identifying and Understanding User Stories,” *Information and Software Technology* (116:October 2018), Elsevier B.V., p. 106169. (https://doi.org/10.1016/j.infsof.2019.08.001).

Urbieta, M., Antonelli, L., Guerra, J., and Rossi, G. 2022. “Tracing User Stories and Source Code Using the Language Extended Lexicon,” in *Information Systems and Technologies*, A. Rocha, H. Adeli, G. Dzemyda, and F. Moreira (eds.), Cham: Springer International Publishing, pp. 413–429.

Urbieta, M., Antonelli, L., Rossi, G., and do Prado Leite, J. C. S. 2020. “The Impact of Using a Domain Language for an Agile Requirement Management,” *Information and Software Technology* (127:November 2019), Elsevier B.V., p. 106375. (https://doi.org/10.1016/j.infsof.2020.106375).

Wake, B. 2003. “INVEST in Good Stories, and SMART Tasks,” , August 17. (https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/, accessed March 16, 2023).

APPENDIX A SAMPLE OF USER STORIES WITH THE AMBITRUS VIOLATED CRITERIA (PRACTICAL IMPLEMENTATION OF THE AmbiTRUS IN ANALYZING AMBIGUITY IN USER STORIES)

ID	User Story	Violated quality
US1a	*"As an administrator, I want to change the profile, so that the page will appear."*	User story *US1a* violates **complete terminology** criteria. The actor labeled *"administrator"* in the *WHO* segment lacks specificity about the particular role or responsibilities. Secondly, the action described as *"the profile"* lacks specificity, leaving room for interpretations of the intended profile. Lastly, the desired outcome of displaying *"the page"* lacks precise information about what kind of page should be presented, making the objective unclear.
US1b	*"As a site administrator, I want to change the company profile, so that the profile page will appear."*	User story *US1a* violates **precise terminology** due to lacks clarity regarding the necessary modifications of the phrase *"change the company profile"* in the *WHAT* segment and unclear benefit of the *WHY* segment.
US1c	*"As a site administrator, I want to modify the content of the company profile, so that I can view the page with new company activities and achievements."*	No violated criterion.
US2a	*"As a website administrator, I want to request my colleagues to administer the company profile, so that I can share responsibilities with them."*	User story *US2a* violates ***consistent terminology*** criterion by using *"website administrator"* to refer to an individual who has the privilege of a *"site administrator"* (see user story *US1b*). Moreover, the term *request* in the *WHAT* segment is not included in the standard glossary of Table 4, causing potential variations in action execution.
US2b	*"As a site administrator, I want to create access to my colleagues, so that they can update the content of the company profile as well."*	No violated criterion.
US3a	*"As a webmaster, I want to grant access to the administrator page to my colleagues, so that they can manage the content of the company profile as well."*	The *US3a* violates ***no synonym across user stories*** criterion. The actor in user story *US3a* is referred to as *"webmaster"* while it is "site administrator" in *US1b*, *US1c*, and *US2b*. Moreover, to maintain consistency and prevent duplications, replacing the term *"grant"* with the standard glossary term *"create"* is recommended (see Table 4).
US3b	*"As a site administrator, I want to create access to the administrator page for my colleagues, so that they can manage the content of the company profile as well."*	No violated criterion.
US4a	*"As a site administrator, I want to update the news and delete expired certificates and achievements on the company profile website, so that people can see the news, certificates, and achievements."*	User story *US4a* violates **atomic** criterion due to the conjunction used in the *WHAT* segment. The use of conjunction *"and"* to separate *"certificate"* and *"achievement"* in the phrase *"update the news and delete expired certificate and achievement"* introduces uncertainty about how *"achievement"* should be treated, whether it should be *"updated"* or *"deleted".*

ID	User Story	Violated quality
$US5_a$	*"As a site administrator, I want to delete certificates after a certain period, so that people can see the still valid certificates."*	User story $US5_a$ violates **overlapping** criterion particularly due to the phrase *"after certain period"* in the *WHAT* segment, which has the potential to be interpreted differently among individuals. The *WHY* segment talks about "seeing still valid certificates" which is not automatically consistent with "deleting certificates after a certain period".
$US6_a$	*"As a site administrator, I want to delete certificates that are obtained from suspicious vendors, indicated by "suspicious vendors" list in our database, so that people can only see legitimate certificates."*	User story $US6_a$ violates **representative** criterion due to discrepancies in the *WHAT* and *WHY* segments. The sentence in the *WHAT* segment, *"certificate that are obtained from suspicious vendors, indicated by "suspicious vendors" list in our database"* contains excessive details that might restrict flexibility in the user story transformation to a system feature. Moreover, the *WHY* segment lacks conciseness, contributing to the complexity of the user story $US6_a$.
$US4_c$	*"As a site administrator, I want to delete expired certificates from the company profile website, so that people can see still valid certificates."*	$US4_c$ and US_6 are potentially violate **conflict-free criterion** due to different actors aimed to achieve the identical benefits in the *WHY* segment. However, these **potentially conflicting situations need** to be manually confirmed via discussion before implementation starts.
US6	*"As a public relation manager, I want to update expired certificates on the company profile website, so that people can see still valid certificates."*	
$US7_a$	*"As a site administrator, I want to be able to change advertisements via setup menu, so that I can target my ads to the most relevant visitors and maximize my ad effectiveness."*	User story $US7_a$ violates **directive criterion** due to unclear action as there is too much detail of the implementation in the *WHAT* segment but insufficient reasoning in the *WHY* segment.
$US7_b$	*"As a site administrator, I want to update viewing options on the company profile website, so that I can set up the time period for frequent advertisement on a particular event in social media."*	$US7_b$ and $US8_a$ violate **duplication-free (well modularized) criterion**. This duplication arises because both user stories potentially refer to identical actions and benefits, despite being distinctly described in different sentences.
$US8_a$	*"As a site administrator, I want to customize viewing preferences on the company profile website, so that I can set up the schedule of my advertisement in social media."*	
$US9_a$	*"As a site administrator, I want to optimize the company profile website, so that I can coordinate promotional posts during high-traffic periods on social media."*	User story $US9_a$ violates **explicit statement criterion.** The phrase *"optimize the company profile website"* in the *WHAT* segment might imply various actions, including implementing a scheduling tool or embedding analytical tools. These actions might suggest different desired benefits outlined in the *WHY* segment. Although the end goal is similar, to enhance social media promotion during peak sessions, these varied interpretations could lead to confusion.

ID	User Story	Violated quality
US10$_a$	*"As a site administrator, I can add certificates to the company profile website in order to delete another certificate."*	*US10$_a$* violates **contextual relevancy criterion.** The expected action in the *WHAT* segment and the corresponding benefit in the *WHY* segment. The attempt to improve user story *US10$_a$* by rewriting the user story into the Connextra template structure, as in user story *US10$_b$*, does not remove the inconsistency between the action in the *WHAT* segment and the associated benefit in the *WHY* segment.
US10$_b$	*"As a site administrator, I want to add new certificates to the company profile website, so that I can delete another certificate."*	To avoid potential conflict between the expected action in the *WHAT* segment and the corresponding benefit in the *WHY* segment, the user story *US10$_b$* might be improved into *US10$_c$*. However, some people might perceive that this change is not sufficient as the expected action in the *WHAT* segment "*add new certificates*" could potentially resemble to the action of "*update*" in the *WHAT* segment of user story *US4$_c$,* while another person might believe that the actions in both user stories are different.
US11$_c$	*"As a site administrator, I want to add new certificates to the company profile website, so that people can see our industry certifications on the website."*	
US12$_a$	*"As a youngster, I want to search for activities nearby related to my interests, so that I can enjoy them."*	User story *US12$_a$* violates **contextual relevancy** criterion. The *WHO* segment lacks contextual relevance in describing the system role, along with vague benefits mentioned in the *WHY* segment.