

# Controlled Experiments on User Stories' Modeling: Past, Present, and Future

Konstantinos Tsilionis<sup>1</sup>, Anis Rahmawati Amna<sup>2</sup>, Samedi Heng<sup>3</sup>,  
Stephan Poelmans<sup>1</sup> and Geert Poels<sup>2,4</sup>

<sup>1</sup>KU Leuven, Leuven, Belgium

<sup>2</sup>Ghent University, Ghent, Belgium

<sup>3</sup>HEC Liège, Université de Liège, Liège, Belgium

<sup>4</sup>FlandersMake@UGent core lab CVAMO, Ghent, Belgium

## Abstract

User stories (US) are short sentences written in natural language, structured around 3 dimensions (*WHO/WHAT/WHY*). They describe functionalities of the to-be software system and constitute the primary requirements artifacts used in agile methods. Originally, a few US templates have been suggested but these were written in a completely informal manner without guidance. In time some research has been made to furnish better guidelines when creating US and develop frameworks to increase their overall quality. Some of these frameworks are based on conceptual modeling, some others on linguistic approaches. The application of these frameworks in real life contexts nevertheless remains an open issue. To evaluate their applicability several experiments have been conducted in various settings. The present paper aims to summarize these experiments and suggest some others for future work.

## Keywords

User Stories, User Story, Agile Requirement Modeling, Modeling Experiment.

## 1. Introduction

User stories (US) are artifacts often used in agile methods (i.e., Extreme Programming and Scrum) and tools (i.e., Jira) to describe user desiderata during the requirements' elicitation stage preceding the implementation of the system. These artifacts are purposed to document major functionalities and/or indispensable features that the to-be system needs to satisfy; they are commonly built around the *WHO*-dimension (i.e., explicating the *role* asking for the functionality), the *WHAT*-dimension (i.e., describing the *functionality*), and the *WHY*-dimension (i.e., justifying the *reasoning* for implementing the functionality). The benefits of utilizing these artifacts are mostly associated to the understandability and readability that they yield to the requirements' gathering process especially when perceived individually or in small sets.


Nevertheless, the utilization of such a simple requirements' specification format does come at a cost. A lot of freedom is given in their creation which is in a sense very positive for users non-


---

*Agil-ISE 2022: Intl. Workshop on Agile Methods for Information Systems Engineering, June 06, 2022, Leuven, Belgium*

✉ konstantinos.tsilionis@kuleuven.be (K. Tsilionis); anisrahmawati.amna@ugent.be (A. R. Amna); samedi.heng@uliege.be (S. Heng); stephan.poelmans@kuleuven.be (S. Poelmans); geert.poels@ugent.be (G. Poels)

ORCID 0000-0001-9702-6941 (K. Tsilionis); 0000-0002-6037-0914 (S. Heng); 0000-0001-9247-6150 (G. Poels)

 © 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

familiar with technical aspects but could lead to poor consistency in the way they are expressed and the overview and consistency of the requirements they express. Also, US do not provide visually-aided requirements representations that would help with the sorting of multiple (and extended) sets of US and would provide alternative modeling configurations for the impending system. The following section provides a discretionary collection of experiments purposed to delineate some of the factors that may be influencing a more qualitative appropriation of US artifacts on behalf of novice modelers.

## 2. Past Controlled Experiments

To begin with, the studies of Dapiaz et al. [1, 2] describe two modes of an experiment investigating the impact of two requirements' specification notations (namely Use Cases and US) on the derivation of static conceptual models (i.e., UML class diagrams). The entire experiment was meant to explore the factors affecting the quality of these models so as to assist agile practitioners and modelers in making informed decisions concerning the choice of requirements engineering notations. First, the authors describe a controlled mode for the experiment where they realize that the US accounted for the participants' preferred notation, contributing to higher quality static conceptual models. However, this mode included a requirements' specification process that was pre-determined by the researchers and uniform to all the subjects. The second mode of the experiment was more pragmatic in the conditions of its execution in the sense that test subjects had to go through the entire process of requirement elicitation and specification to derive their conceptual models. This second mode revealed that it is not much the notation that affects the quality of the derived models but other factors inherent to the derivation process itself. Nonetheless, the findings support an overall degree of optimality attributed to the US notation as measured by the correctness and completeness of a manually derived UML class diagram by novice modelers.

The problem of poor representability concerning the nature, granularity, and interdependence of the constituted elements within a US set has been acknowledged in the study of Wautelet et al. [3]. To deal with these issues, the authors furnish a unified US model (see Fig. 1) for tagging the elements of the *WHO*-, *WHAT*-, and *WHY*-dimensions of a US; each tag represents a concept with an inherent nature and defined granularity. The constructs of this model (i.e., *Role*, *Task*, *Hard-Goal*, *Soft-Goal*, etc.) become the basis for the creation for the visual representation of a US structuring method called the *Rational Tree* [4, 5]. The latter uses the notation of the i\* Strategic Rationale (SR) diagram [6] to build various trees of relating US elements in a single project (Fig. 2.a) with the purpose of identifying depending US, identifying Epic ones and group them around common Themes (Fig. 2.b). In this way, the utilization of the Rational Tree can recognize and reduce even further any occurring modeling redundancies during the stages of requirements' analysis and design.

The study performed by Wautelet et al. [7] was meant to uncover the difficulties that software modelers face in the process of building visually-aided representations (i.e., conceptual models) when the software problem is structured in the form of US sets. For this, the authors have conducted an experiment purposed to analyze, describe, and compare the feasibility of novice modelers (master-level students) and experienced modelers (PhD candidates and Post-docs) to

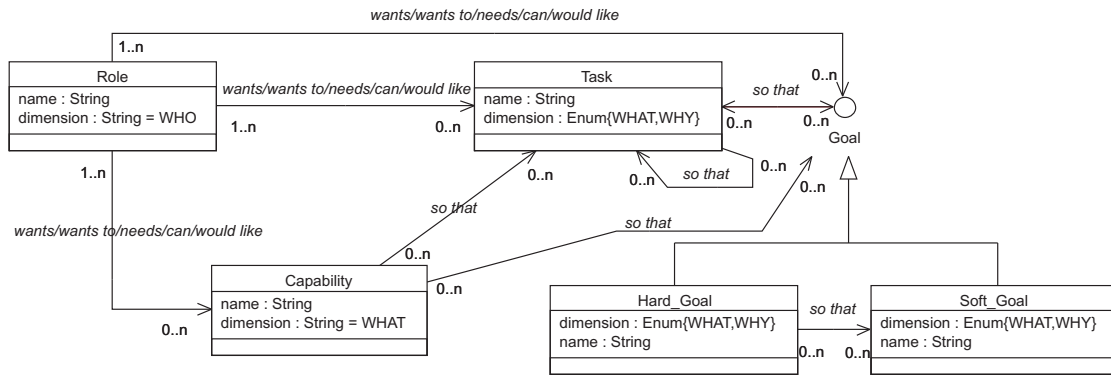


Figure 1: Unified Model for User Stories.

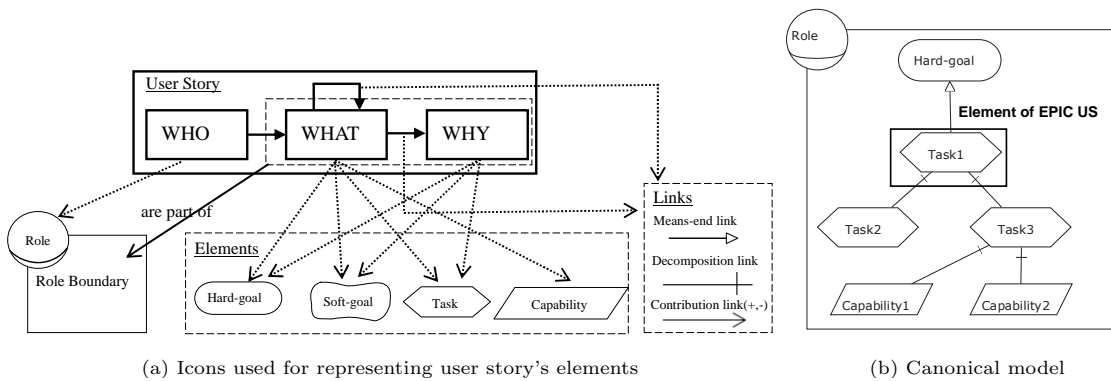


Figure 2: Using Rationale Tree to Structure User Stories' Sets.

build and use a *Rationale Tree diagram*<sup>1</sup> out of an existing US set. Both groups had to start by recognizing and tagging elements of the *WHO*-, *WHAT*-, *WHY*-dimensions of a given US set to the constructs of the unified model of [3], with each tag representing a concept with an inherent nature and defined granularity. Once tagged, the US elements had to be graphically represented by building one or more Rational Tree diagrams. Overall, the participants had some difficulties in making the association between the *WHO*-, *WHAT*-, *WHY*-affiliated concepts within the US sets and the modeling constructs of the unified US model. Nevertheless, there was no significant difference in the ability to build a Rational Tree between novice modelers and more experienced researchers since all test subjects produced artifacts that would describe rather well the software problem.

The experiment of Wautelet et al. [8] was meant to identify whether improving US quality through the Quality User Story (QUS) framework proposed by Lucassen et al. [9] would lead to a better identification of different concepts of US sets and a better development of Rationale

<sup>1</sup>The *Rational Tree diagram* refers to the artifact produced when employing the Rational Tree as a user story structuring method.

Tree diagrams. The experiment was performed on 34 master-level students composed of 2 groups, one using the raw US set while the other using the QUS-compliant one. The exact goals were i) to identify the concepts of non-functional requirements, missing requirements, epics and themes; ii) to be able to build a Rationale Tree diagram; iii) to evaluate the impact of the Rationale Tree on the identification of these concepts; iv) to evaluate the impact of using an improved in quality (through QUS) US set on the ability of to identify the aforementioned concepts. Overall, the authors could not conclude with certainty whether the improvement of the quality on a US set could improve the ability of novice modelers to identify crucial constructs of a specific modeling case (i.e., missing requirements etc.). Nonetheless, the experiment was able to portray that improvements in the quality of US is correlated with an enhanced ability on behalf of the modelers to identify and represent elements like tasks, capabilities and links.

The studies of Tsilionis et al. [10, 11] perform a primary theoretical analysis of two distinct techniques that can be used for structuring sets of US but have different complexities in their application and different abilities to represent dependencies and decompositions. In essence, the authors try to investigate which user story structuring method (i.e., the Rational Tree or the User Story Mapping [12]) produces artifacts that allow for a better comprehension of the software problem. The answer to this question represents the basis for the conduct of a controlled experiment; indeed, a first group of novice modelers (master-level students) has been required to employ the Rational Tree method and build artifacts out of a given set of US; a second group of novice modelers was asked to do the same with the User Story Mapping approach. The results suggest that whilst the Rational Tree method seems not as easy to understand as the User Story Mapping, when the modelers receive practical, step-by-step guidance on how to implement the Rational Tree, they managed to produce qualitative representations of the software problem. Additionally, the authors notice that there are no significant differences between the participating groups, regardless of the US-structuring method they utilize, in terms of recognizing basic constructs within their given sets of US (i.e., missing requirements, Epics, etc.).

### **3. Ambiguity in User Stories - A Planned Experiment**

While the experiments described in the previous section focus on conceptual modeling as an approach to help writing better US or making better use of US, a different stream of research has investigated the quality of US from a linguistic perspective. Even when highly structured because of the use of predefined templates, US are essentially a textual format for describing requirements, and are thus prone to ambiguity just as any other type of requirements described in natural language. Recently, Amna and Poels performed a systematic literature review of the research on ambiguity in US [13]. This review indicated that vagueness in the formulation of US results in problems like inconsistent requirements, incomplete requirements, and duplicated requirements. These problems have been attributed to the intrinsic quality of US [8, 14], the cognitive capability of developers [7, 15], and US misrepresentation [16, 17]. The ambiguity that results from problematic US can be manifested at different linguistic levels (i.e., lexical, syntactic, semantic, pragmatic).

To help identifying issues with US that possibly result in ambiguity, we are developing a

framework that classifies these issues according to the linguistic level at which ambiguity manifests itself and the consequence of this ambiguity in terms of requirements quality (i.e., vagueness, inconsistency, insufficiency, duplication). The framework illustrates the issues with examples (as kind of anti-patterns) and also provides examples of how the issues can be resolved. The issues themselves are based on quality criteria taken from the QUS framework proposed by Lucassen et al. [9] and the Agile Requirements Verification framework proposed by Heck and Zaidman [18]. The idea is that developers can use the framework when writing US or reviewing US written by others.

To test this idea (and the framework itself), a first 'laboratory' experiment is planned. The experiment will involve advanced students, including both IT students and business students with a major in IT. These students have preliminary knowledge in Requirements Engineering (RE), US writing and reviewing, and Agile Software Development (ASD), and thus act as proxies for real developers. Participating students will be randomly assigned to one of two groups. Each group is required to review a set of US in which we have injected problems related to the issues covered by the framework. The treatment group will be instructed to make use of the framework to detect these problems, while the control group will be asked to do the same, without the help of the framework. Participants are required to tag the US dimension WHO/WHAT/WHY that is problematic, and will also be asked to provide a short explanation for the potential ambiguity issues they identify (i.e., why is it problematic?).

Both groups will first receive some training (e.g., a tutorial) to make them aware of requirements ambiguity and its consequences for software development. Prior to the experiment, a short questionnaire will be distributed to collect information regarding the participants' profile. After the experiment, participants in the treatment group will be asked feedback regarding the usefulness of the framework. Apart from that we will compare the number of problems identified to the total number of problems injected (i.e., an absolute measure of how effective the framework is) and also make a comparison between both groups (i.e., a relative measure of effectiveness). More detailed analysis will be performed for the different linguistic levels of ambiguity and the requirements quality problems that might be caused by ambiguity in US.

## 4. Conclusion and Future Work

This paper presented a review of experiments that have been conducted in US modeling. Those are typically designed to investigate the effectiveness of artifacts to comprehend the correct meaning of user stories. While conceptual models have been commonly used as artifacts to make better sense and use of US, linguistic approaches can help in identifying and ultimately avoiding badly written US. We are currently developing a framework based on such linguistic approach, and plan an experiment to evaluate its effectiveness and perceived usefulness.

## References

- [1] F. Dalpiaz, A. Sturm, Conceptualizing requirements using user stories and use cases: A controlled experiment, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2020, pp. 221–238.

- [2] F. Dalpiaz, P. Gieske, A. Sturm, On deriving conceptual models from user requirements: An empirical study, *Information and Software Technology* 131 (2021) 106484.
- [3] Y. Wautelet, S. Heng, M. Kolp, I. Mirbel, Unifying and extending user story models, in: *International conference on advanced information systems engineering*, Springer, 2014, pp. 211–225.
- [4] 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)*, IEEE, 2016, pp. 1–12.
- [5] Y. Wautelet, S. Heng, S. Kiv, M. Kolp, User-story driven development of multi-agent systems: A process fragment for agile methods, *COMLAN* 50 (2017) 159–176.
- [6] E. Yu, Modeling strategic relationships for process reengineering., *Social Modeling for Requirements Engineering* 11 (2011) 66–87.
- [7] Y. Wautelet, M. Velghe, S. Heng, S. Poelmans, M. Kolp, On modelers ability to build a visual diagram from a user story set: a goal-oriented approach, in: *International Working Conference on Requirements Engineering: Foundation for Software Quality*, Springer, 2018, pp. 209–226.
- [8] 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*, Springer, 2019, pp. 3–19.
- [9] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, S. Brinkkemper, Improving agile requirements: the quality user story framework and tool, *Req. Eng.* 21 (2016) 383–403.
- [10] K. Tsilionis, J. Maene, S. Heng, Y. Wautelet, S. Poelmans, Evaluating the software problem representation on the basis of rationale trees and user story maps: premises of an experiment, in: *International Conference on Software Business*, Springer, 2020, pp. 219–227.
- [11] K. Tsilionis, J. Maene, S. Heng, Y. Wautelet, S. Poelmans, Conceptual modeling versus user story mapping: Which is the best approach to agile requirements engineering?, in: *RCIS2021*, Springer, 2021, pp. 356–373.
- [12] J. Patton, P. Economy, *User story mapping: discover the whole story, build the right product*, O’Reilly Media, Inc., 2014.
- [13] A. R. Amna, G. Poels, Ambiguity in user stories: A systematic literature review, *Information and Software Technology* 145 (2022) 106824.
- [14] M. Urbietta, L. Antonelli, G. Rossi, J. C. S. do Prado Leite, The impact of using a domain language for an agile requirements management, *Information and Software Technology* 127 (2020) 106375.
- [15] J. Jia, X. Yang, R. Zhang, X. Liu, Understanding software developers’ cognition in agile requirements engineering, *Sci. Comput. Program.* 178 (2019) 1–19.
- [16] M. Elallaoui, K. Nafil, R. Touahni, Automatic transformation of user stories into uml use case diagrams using nlp techniques, *Procedia Computer Science* 130 (2018) 42–49.
- [17] 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, *Cognition, Technology, and Work* 22 (2020) 437–457.
- [18] P. Heck, A. Zaidman, A quality framework for agile requirements: A practitioner’s perspective (2014).