Available online at www.sciencedirect.com

ScienceDirect

Information and Software Technology 00 (2022) 1–27

Journal Logo

The impact of using a Domain Language for an Agile Requirement Management

Matias Urbieta[a,b,*], Leandro Antonelli[a], Gustavo Rossi[a,b], Julio Cesar Sampaio do Prado Leite[c]

[a] *Lifia, Fac. de Informática,UNLP, Argentina*
[b] *CONICET, Bs As, Argentina*
[c] *Dep. Informática, PUC-Rio, RJ, Brazil*

Abstract

Context: The development of software systems is a complex activity because of its nature and the management of its construction. It is challenging to create and follow a plan. Moreover, budget overrun is a common consequence of this situation. Agile methods, like Scrum, help to mitigate this problem using incremental and iterative development. Agile methods jump start new developments, but it is difficult to be agile after several months when the software has to deal with many requirements that are scattered and tangled across several User Stories written in different Sprints. **Objective**: In this paper, we propose a traceability approach anchored on an index structure to access specific User Stories from a large set. Our proposed strategy has the goal to consolidate the information dispersed in different User Stories into a particular lexicon: The Language Extended Lexicon (LEL). **Method**: The proposed approach consists of a set of rules which extract the information dispersed in the User Stories and organize it in symbols of the Lexicon. Thus, the Lexicon supplies a consolidated and organized structure to mitigate the problem of tangled information that generates lack of traceability among different sprints. **Results**: We assessed how the Lexicon built by our approach improves everyday activities related to requirement management. The assessment is based on a quantitative evaluation with 36 subjects. **Conclusion**: The approach presents benefits for requirement tracing in agile methodologies supported by the preliminary results of the evaluation. We have developed an application (a prototype) that automates the LEL derivation rules from a set of User Stories.

Keywords: Domain Knowledge, Agile methods, User Stories, Language Extended Lexicon

1. Introduction

The development of a software system is a complex activity. The coordination of many people to produce the interconnected products is one of reason of this complexity [1]. Besides that, the nature of the software also makes its development a complicated task. Brooks [2] identifies four characteristics that make the software a different kind of artifact altogether contributing to the complexity of its construction: invisibility, modifiability, conformity, and complexity. The complexity is an intrinsic characteristic. Even if the software is implemented in a programming language that operates at the level of the problem domain, the software still needs to define relationships between real-world entities precisely, identify exception cases, and anticipate all possible state transitions. These characteristics are some of the reasons why there are so many budget overruns as well as failure to deliver what was expected [3].

*Corresponding author
Email addresses: `murbieta@lifia.info.unlp.edu.ar` (Matias Urbieta), `lanto@lifia.info.unlp.edu.ar` (Leandro Antonelli), `gustavo@lifia.info.unlp.edu.ar` (Gustavo Rossi), `http://www.inf.puc-rio.br/julio` (Julio Cesar Sampaio do Prado Leite)

Agile methods like Scrum rely on iterative and incremental development to reduce the gap between the expectations of the clients and the software built. In every development cycle (Sprint) a running application is released for its revision, and the client can provide valuable feedback to correct or enrich the software. In particular, this feedback is provided by a product owner, a mediator who represents stakeholders in general: clients and users. In this paper, we use both terms, although it is clear that users have an interest in the product as a tool, while the client has an economic interest. Thus, the software is periodically validated by the user instead of being delivered to the client as a completely unknown product after several months of development.

Even though there are reported benefits [4], it is not very easy to be agile. In this paper we focus on challenges related to requirement engineering [5]. User Stories play a vital role, but their definition requires ability. In large-scale agile adoption [6], the high-level requirements defined by business analysts must be broken down in small definitions to a size that can be estimated. During the process, the team must face the complex task to refine the requirement and estimate the effort. In order to do that, the team needs the context about the functionality and relies on other User Stories previously implemented. Nevertheless, there is very little information on what happens to the User Stories after they are turned into code [7]. On the other hand, face-to-face communication is promoted in Agile Software Development (ASD). In most cases, developers assume that the code is a more reliable source than other documents [8]. Thus, they need to trace it to the related User Stories.

In this context and after several sprints of developments, an issue about requirements traceability arises. Many User Stories are not related to each other. Sometimes the different User Stories describe requirements that complement others but sometimes modify previous definitions. Thus, it is tough to follow the evolution of requirements. Furthermore, the knowledge of the application becomes tacit. A team member who wants to understand some application features can ask his/her teammates for information. However, relying on team members may be misleading for different sets of reasons: like team turn over, or predominance of the code perspective, or different vocabulary used in different Sprints [9]. The team member can also perform reverse engineering from the source code, but it is harder and prone to errors as it was reported in the Global Software Development context [10]. In this case, when new members join a team, the low quality of software artifacts other than source code makes the team waste time to transfer the knowledge to new members dropping the overall team's performance [8]. When using User Stories for anchoring functional requirements, the functionality description is scattered and tangled across several User Stories written in different Sprints. The team member must review every User Story from every Sprint to collect the whole description about some functionality.

A glossary is a software requirement specification artifact used to describe the core elements in the context of the application, through the description of its vocabulary. It is useful towards introducing the domain to the team members that are not familiar with the business. There are several proposals to obtain a glossary from requirements artifacts [11] [12] [13] in order to capture the language. Nevertheless, the glossary can also be used to group modules of the application in order to relate requirements, design, and application components. There are many different kinds of glossaries. There are simple ones that define acronyms, and there are more complex ones that define words and expressions like the regular dictionary.

The Language Extended Lexicon (LEL) [14] supports the construction of a glossary and categorizes the symbols in: Subject, Objects, Verbs, and States. Every symbol (word or expression) is described through two attributes: notion (denotation) and behavioral responses (connotation). It is essential to mention that the name Language Extended Lexicon is used to describe the structure of the glossary, as well as the glossary constructed for a specific domain. Thus, in order to differentiate both things, we use the term LEL to refers to the structure, while the expression LEL lexicon to refer to a specific glossary. As such, the LEL lexicon differs from a regular dictionary, as it not only defines (denotation) a term but also provides its expanded meaning (connotation) in the given environment. The LEL was shown to be an effective technique for building lexicons since it conforms to the mechanism used by the human brain to organize expert knowledge [15]. It is an effective technique because it allows to describe and communicate the knowledge captured by grouping in chunks of information (the terms), which are described in a circular way. Because of that, the LEL lexicon is very convenient for stakeholders with no technical skills. Considering that the categorization is related to some essential elements in different modeling techniques (i.e., object orientation and relation entity) people with technical skills will also profit from its use [16].

In this paper, we propose a strategy to consolidate the information dispersed in different User Stories into an LEL lexicon. The contribution of this work is threefold:

- The approach consists of a set of rules that extract the information from User Stories to create a consolidated and organized LEL lexicon;
- An application that implements the rules in order to automatize the creation of the LEL lexicon from a set of User Stories;
- A preliminary evaluation based on an experiment that evaluates the use of the LEL lexicon created by the proposed rules. The quantitative analysis was an experiment based on a realistic domain supporting the benefits of our approach.

Thus, this work contributes with an approach combining agile methods with the LEL lexicon to improve requirements management, by providing traceability and helping knowledge sharing among team members.

The paper is organized in the following way. Section 2 explores some related works. Section 3 presents some background needed to understand the approach. Section 4 describes the proposed approach. Section 5 reports on quantitative results from an experiment with our approach. Section 6 describes a tool that implements the strategy. Finally, Section 7 concludes pointing out future work.

2. Related Work

Agile methods had successful results. However, they must deal with requirement traceability. Our approach fits the majority of the best practices described by Mäder et al. [17] for implementing effective traceability. Dimitrijevic et al. [18] performed a comparative study of tools to support User Stories management. They discovered that although there is a great variety of tools with a lot of functionality, professionals need simple tools easy to use. We propose a technique with a supporting tool that can be used with any other tool. Teams can use their tools to manage the User Stories, and after each sprint, they can process the User Stories with our tool.

Lucassen et al. [19] proposed a framework to obtain high-quality User Stories identifying inconsistencies, duplication, and dependencies. Although our approach does not deal with quality issues, it can contribute to detecting inconsistencies and duplication since it relates to different User Stories. Mavin [20] is interested in reorganizing requirements written in natural language to improve their quality as well as their access. We agree with the fact that requirements must be processed to build a new model derived from the original one to improve access and readability of the information the original model has. Franch et al. propose a template to specify requirements in the context of requirement patterns [21]. They aim to improve the quality and access to the requirements as we propose. Nevertheless, they provide a metamodel to guide the requirement description, while our approach considers that the User Stories are already described and we transform the knowledge they have captured to describe a glossary. Fu et al. [22] propose a similar approach providing templates for safety requirements

Alaa et al. [23] report an experiment that concluded that the inclusion of Use Cases in Agile Requirements benefit Agile teams. Our approach consists of building a glossary instead of Use Cases. Nevertheless, our approach integrates the knowledge scattered in the User Stories achieving the same goal than Use Cases. Our work is also related to the proposal of Trkman et al. [24]. They are also interested in providing a context to the User Stories building a business process model in order to understand the relationships and dependencies between the User Stories. Darimont et al. [25] are interested in reuse. They propose an approach to implementing a library of requirements. We build a glossary for a software system, but we can also use User Stories from different projects to generate a richer glossary.

Soares et al. [26] analyzed the problem of "documentation debt" (lack of supporting software artifacts). They discovered that it is one of the causes of technical debt. Sim et al. [27] performed a field study of the requirements practices of an agile software development team. Sometimes, agile projects are inside the umbrella of conventional projects. Since our strategy obtains a unique and consolidate piece of information describing the requirements, it helps to close the gap between both kinds of philosophies, and it also helps with technical debt. Barbosa et al. [28] also contribute to reducing the documentation debt, since they provide a strategy to identify duplicated User Stories measuring the distance between two User Stories assessing the lexical distance in the textual description. Our approach relates several User Stories with a different technique.

3. Background

This section introduces two main concepts developed in this work. First, it describes the Language Extended Lexicon, the artifact used to consolidate the domain knowledge. Then, it also describes some basic concepts of Scrum: User Stories and sprints.

3.1. Language Extended Lexicon

The LEL is a lexicon whose goal is to record the definition of terms that belong to a domain [14]. The main objective of the LEL is to record the language (words and expression) of the domain. The LEL is not a model of the solution. However, this language will provide the knowledge to produce more complex artifacts as models of the system to be built (i.e. requirements). It is important to mention, that the literature has used the term LEL as both the language to write the description of the lexicon (a specification) and the lexicon itself.

We believe that the LEL is a convenient tool because three characteristics that we found in our experience: it is easy to learn, it is easy to use, and it has good expressiveness. We have used the LEL in many domains, some of them very complex, and we had good results. Cysneiros et al. [29] report the use of LEL in a complex domain as the health domain.

Terms (called symbols within the LEL) are defined through two attributes: notion and behavioral responses. Notion describes the symbol denotation, i.e. the intrinsic and substantial characteristics of the symbol, while behavioral responses describe connotation, i.e. the relationship between the described term and other terms.

Each symbol of the LEL lexicon belongs to one of four categories: Subject, Object, Verb ,or State. This categorization guides and assists the requirements engineers during the description of terms. Table 1 shows each category with its characteristics and how to describe them.

Table 1: LEL categories

Category	**Description**	**Notion**	**Behavioral responses**
Subject	Active elements which perform actions	Characteristics or condition that subject satisfies	Actions that subject performs
Object	Passive elements on which subjects perform actions	Characteristics or attributes that object has	Actions that are performed on object
Verb	Actions that subjects perform on objects	Goal that verb pursues	Steps needed to complete the action
State	Situations in which subjects and objects can be located	Situation represented	Actions that must be performed to change into another state

The following example defines a verb from an e-commerce Web application. It is essential to mention that although the expression contains a verb and an object, it belongs to the category verb because it begins with a verb. The definition of "choose items" instead of "choose" relies on the possibility of defining another symbol like "choose payment method". Moreover, the object "item" could also be defined as another isolated symbol of Object category. Symbols are underlined in order to show that they are in the LEL lexicon, thus creating a natural hypertext.

Verb: Choose items
Notion: Select the items with the goal of buying them
Behavioral responses:
The client enters the category of desired items.
The web site shows all the products.
The client selects the items.

3.2. User stories and Sprints

A User Story is a description in natural language that captures what the user wants to achieve. There are many templates to describe User Stories, but the template generally used considers three attributes: a *role*, a *goal/desire* and a *reason* [30]. The *role* defines the user who interacts with the application to use the feature described by the *goal/desire*.

The *goal/desire* represents the requirement that the application must fulfill. Both these attributes refer to elements within the scope of the application. In contrast, the *reason* belongs to the context of the application, and it states why the user requires that the application provides the functionality described in *goal/desire*. This template describes the application to be built and the context in which the application will be used. These descriptions complement themselves providing different points of view of the functionality required:

As *role*
I want *goal/desire*
So that *reason*

We also give an example of a User Story for the same e-commerce web application:

As a client
I want to choose items
So that I can buy them

Scrum is an iterative and incremental software development process; thus, the construction of the application is usually organized in periods of two weeks, which are called Sprints. In each Sprint, some functionality is agreed to be developed and it is specified through User Stories. When the Sprint finishes, a software application that satisfies the User Stories is obtained, and a new sprint begins with new User Stories that enrich the functionality of the application.

4. Our Approach in a nutshell

The aim of the approach is to consolidate the knowledge scattered in User Stories into an LEL lexicon that can be considered as the system functionality index for easing the access to the requirements for traceability and knowledge transferring purposes. The approach needs as input the User Stories described in each Sprint to obtain an LEL lexicon as output. The construction of the LEL lexicon is an iterative process that involves creating and enlarging the LEL lexicon incrementally after each Sprint.

The approach is proposed to be applied in the context of an agile approach. First, the project should start as usual with the definition of a basic backlog. Then, a set of user stories are selected and refined during the sprint planning. After that, the development team works on developing software artifacts. Our approach should be executed during this part, where the analyst extracts the LEL lexicon from the set of user stories under development. Figure 1 shows an BPMN (Business Process Modeling Notation) diagram [31] depicting the context of our approach, which is highlighted in the Scrum Team lane.

The approach steps are:

1. User stories definition. The user stories are defined as part of the backlog and then detailed during the Sprint planning by the product owner, scrum master, developers, and stakeholders. The user stories may be enhanced with additional information like acceptance criteria or mockups. We focus on the base User story definition. The expected outcome of this step is a list of well-formed user stories elaborated by an analyst.
2. User story normalization and filtering. This step is twofold: User stories are rewritten in passive voice. On the other hand, only non-functional requirements are filtered out by the analyst.
3. LEL symbols extraction. The User stories gathered in the previous step are processed to extract LEL symbols (i.e., object, subjects and verbs.). LEL entry is identified and a new symbol is created and linked to the User Story that provided the information. Later on, a different User story may update the same symbol.To extract the LEL symbols from a User story we apply a set of rules that are described later on. The outcome of this step is a set of LEL symbols linked to user stories. This task can be performed manually but in this work we present a tool that automates the extraction of LEL symbols. Thus, the LEL lexicon provides essential traceability information for finding the information. It helps answering questions like: where is some object handled? Or in what way is a business entity used?

Figure 2 shows a Scrum process executing through several sprints. It exemplifies an how LEL lexicon is extracted in sprint 1 (at left-hand side) and how the LEL lexicon is enriched in sprint 2 (at right-hand side). The figure shows only the first and second sprint, nevertheless, the process of deriving the LEL lexicon from the User Stories continues

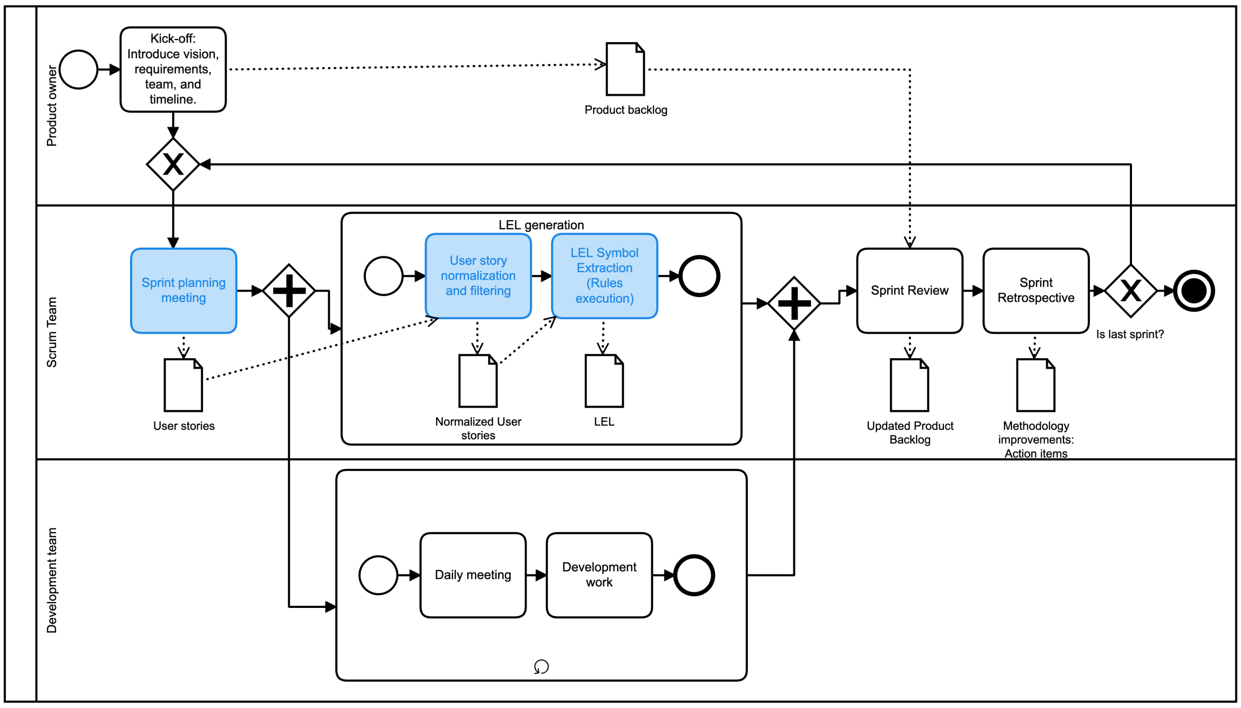


Figure 1: BPMN describing the main steps of the approach

during the whole process, sprint by sprint. It is worth mentioning that all sprints are treated equally from the first sprint to the last one. The product backlog is the input for the iterative development process. Before a sprint starts, a planning meeting is performed where some requirements are selected (and removed) from the product backlog and pulled into the spring backlog. After the first iteration (i.e. sprint) some symbols are identified (column "id"), and some descriptions are added (columns "notion" and "behavioural responses") from User Stories of the sprint backlog. In the following springs the product backlog evolves adjusting, removing or adding new requirements. Then, new symbols and definitions may be added in the sprint.

Let's consider the following User Stories related to the first Sprint:

User Story #1:

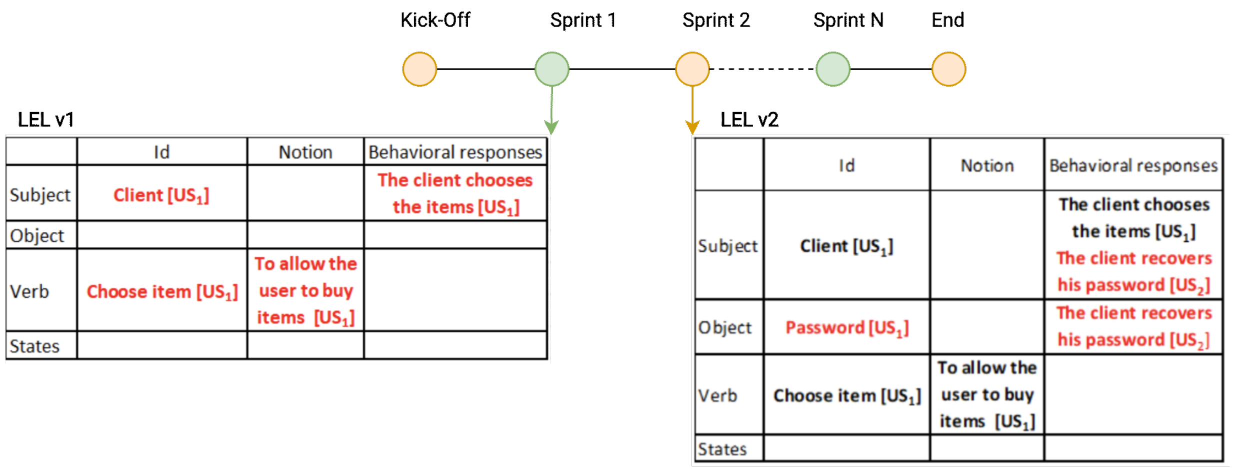
```
As a client
I want to choose items
So that I can buy them
```

According to the strategy proposed, the previous User Story leads to the definition of the following two symbols with their description. Some attributes are empty, because the previous User Story does not have information to describe completely all the attributes.

Subject #1: Client
Notion: [empty]
Behavioral responses: The client chooses the items

Verb #1: choose items
Notion: To allow the client to buy items
Behavioral responses: [empty]

Let's consider the following User Story related to the second sprint:



LEL v1

	Id	Notion	Behavioral responses
Subject	Client [US_1]		The client chooses the items [US_1]
Object			
Verb	Choose item [US_1]	To allow the user to buy items [US_1]	
States			

LEL v2

	Id	Notion	Behavioral responses
Subject	Client [US_1]		The client chooses the items [US_1] The client recovers his password [US_2]
Object	Password [US_1]		The client recovers his password [US_2]
Verb	Choose item [US_1]	To allow the user to buy items [US_1]	
States			

Figure 2: LEL evolution by sprints

User Story #2

```
As a client
I want to recover my password
So that I can login into the website
```

It gives origin to the symbol password with the following description. Moreover, the User Story refers to the client; so, the client symbol is enriched with a new description in the behavioral responses.

Object #1: password
Notion: [empty]
Behavioral responses: The client recovers his password

Subject #1: Client
Notion: [empty]
Behavioral responses:
The client chooses the items
The client recovers his password

It is important to mention that the objective of creating the LEL is describing the language inside the boundaries of the application in order to provide a traceability tool. With that goal, the rules proposed focus on describing roles (subjects), their actions (verbs) and the entities they use (objects). The rules deal with these three elements that can be found inside the boundaries of the application. Thus, the rules take this information from the first two attributes of the User Stories, the "As a" and the "I Want", since both attributes describe elements of the application. But the attribute "so that" describes a goal that some user pursues in the real world (outside of the application), that is why this attribute is not considered in the rules. Since the User Stories do not explicitly describe information about states (of subjects, verbs or objects), the proposed approach does not consider rules to identify and describe state symbols in the LEL. Although implicitly the User Stories could have information about states, further approaches with semantic support could help to obtain them, since it is not possible to obtain them with only a syntactical analysis. Because of this limited information captured in the User Stories, some attributes of the subjects, objects, and verbs are not captured with the rules proposed.

As long as different sprints occur, the lexicon is enhanced incrementally with new interconnected subjects, verbs, and objects that are linked to the user stories in which they are referenced. Let's refer an LEL symbol using a 3-tuple definition $s = \langle i, n, br \rangle$ where i denotes an identifier, n corresponds to the notion component and br refers to the behavioral response. The LEL of a project is defined a set of symbols $LEL = \{s_1, s_2, ..., s_n\}$. The LEL is incrementally built at each sprint (N interation) merging the previous LEL status ($N-1$ iteration) with new symbols from the user stories or enhancing already defined symbols with new notions (n) and behavioral responses (br). In equations 1 and 2, the symbols merging pseudo-code is depicted in Algorithm 1.

$$LEL_0 = \emptyset \tag{1}$$

$$LEL_{SprintN} = merge(LEL_{SprintN-1}, \{s_1^{SprintN}, s_2^{SprintN}, ..., s_n^{SprintN}\}) \tag{2}$$

Algorithm 1 Merge Algorithm

```
1:  function MERGE($LEL_1$,$LEL_2$)
2:      $processed = \emptyset$
3:      $result = \emptyset$
4:      for $s_1 \in LEL_1$ do
5:          for $s_2 \in LEL_2$ do
6:              if $i_{s_1} = i_{s_2}$ then                                ▷ Check if are the same object
7:                  $result \leftarrow result + \langle i, n_{s_1} \cup n_{s_2}, br_{s_1} \cup br_{s_2} \rangle$      ▷ Merge both symbols
8:                  $processed \leftarrow processed + i_{s_1}$
9:              else
10:                 if $i_{s1} \notin processed$ then
11:                     $result \leftarrow result + s_1$
12:     for $s_2 \in LEL_2$ do                          ▷ Check if the symbol's id has been processed
13:         if $i_{s_2} \notin processed$ then
14:             $result \leftarrow result + s_2$
        return $result$
```

It is worth to mention that any inconsistency or ambiguity present in one or more users stories will likely be directly reflected in the lexicon because of the nature of the derivation rules. For the sake of conciseness, the requirement consistency checking is out of the scope of this work.

The process to build an LEL lexicon from the User Stories consists of the following steps.

4.1. User stories description

The Product Owner specifies a set of User Stories that are stacked by priority in the backlog. At the beginning of every Sprint a planning meeting is celebrated, the User Stories are taken from the backlog, estimated by the team, and enriched with a better description and acceptance criteria.

4.2. Functional requirement filtering

User stories can define both functional and non-functional requirements. Since our approach only deals with functional requirements this step consists in isolating User Stories that only refers to functional requirements. This step must be done manually.

4.3. User stories normalization

User stories may be written in passive voice, using pronouns, and other variations. Their descriptions must be normalized to make possible the automatic processing. This normalization should not require too much effort since every sprint should have no more than 20 or 30 User Stories depending on the size of the team and the granularity of the User Stories. Moreover, the normalization needed for automatic processing demands the use of good practices to describe requirements. So, the team takes benefit from this normalization since it improves the readability of the User Stories.

4.4. LEL lexicon description from User Stories

Every User Story is processed, and its information is extracted using different rules to identify symbols and describe them. The rules consider the identification of three categories of symbols: Subjects, Objects, and Verbs. Then, our approach can obtain information from User Stories to describe the behavioral responses of the three categories of symbols identified, and we only consider the description of the notion of the verbs. During the definition of the LEL, links are defined in order to follow the circularity principle. Moreover, each notion and behavioral response is tagged with the User Story id in order to contextualize its definition.

The rest of the section describes each one of the rules to build an LEL lexicon from a set of User Stories. This set of rules is the main contribution of this paper. We have designed these rules from our experience working with the LEL and User Stories. We also present examples that show how to apply the approach to a small but real set of User Stories [32] . The original set of User Stories included non-functional requirements that were discarded in the previous step: User Stories normalization. We list the normalized User Stories:

US#1

```
As a moderator
I want to create a new game by entering a name and an optional
description
So that I can start inviting estimators
```

US#2

```
As a moderator
I want to invite estimators by given them a URL where they can
access the game
So that we can start the game
```

US#3

```
As an estimator
I want to join a game by entering my name on the page I received
the URL for
So that I can participate
```

US#4

```
As a moderator
I want to start a round by entering an item in a single multi-line
text field
So that we can estimate it
```

US#5

```
As an estimator
I want to see the item we are estimating
So that we can estimate it
```

US#6

```
As a moderator
I want to edit an item in the list of items to be estimated
So that I can make it better reflect the team's understanding
of the item
```

US#7

```
As a moderator
I want to export a transcript of a game as a CSV file
So that I can further process the stories and estimates
```

The processing of User Stories is based on eight rules. They are defined by two parts: *when* and *then*. The former defines the conditions that must be satisfied to apply the rule. The latter defines the element that must be recorded. That is, some rules determine that a symbol should be created, and other rules determine that some definitions should be added to a symbol. The rules only create symbols of the categories subject, object, and verbs. And they also define symbols adding an expression in the notion, or behavioural responses. In section 6 we present a tool that supports the derivation of using Natural Language Processing techniques. Next, we introduce the rules and illustrate how each rule applies to the set of User Stories.

Rule 1. Subjects identification. From each User Story, roles after "As a" attribute must be registered as a Subject.

When As a role
Then register role as a new subject symbol.

This rule determines that all the non-repeated roles in User Stories must be considered as subject symbols. The role in the User Story describes an external user of the application (a person or another system) that performs some operation with the application. This is the definition of the category Subject in the LEL lexicon. In the example, there are only two different roles: *moderator* and *estimator*. These symbols have no description because they will be completed with the application of other rules. Thus, the LEL lexicon obtained is the following:

Subject: Moderator [US#1] [US#2] [US#4] [US#6 US#7]
Subject: Estimator [US#3] [US#5]

It is noteworthy to mention that each symbol and definition that is recorded refers to the User Story from where the element came from. The reference is denoted with the pattern *US#* followed by the User Story Id embraced with brackets ("*[" and "]*").

Rule 2. Behavioral responses identification for subjects. Goals or desires described at the "I want" clause represent behavioral responses, since "I want" clause describes the actions that the application should perform and behavioral responses describe the action that the subject needs to execute with the application. The sentences are rewritten from first to third person. Moreover, the "I want" clause has the format *subject verb [object / subject] [other expression]*. Only *subject verb [object / subject]* must be considered as behavioral response.

When As a role I want to *verb [object / subject] [other expression]*
Then register role *verb [object / subject] as a new* behavioral response to the subject role.

There are five User Stories for *Moderator* and two User Stories for *Estimator* subject:

Subject: Moderator [US#1], [US#2], [US#4],[US#6],[US#7]
Behavioral responses:
The moderator creates a new game [US#1]
The moderator invites estimators [US#2]
The moderator starts a round [US#4]
The moderator edits an item [US#6]
The moderator exports a transcript [US#7]

Subject: Estimator [US#3],[US#5]
Behavioral responses:
The estimator joins a game [US#3]

The estimator sees the item [US#5]

Rule 3. Verbs identification. By considering "I want" clause (the same clause used in rule 2 to define the Behavioral responses of the subject), the Verb components are extracted, since this clause describes information about system functionality, and the functionality is described throughh actions (operations, services) provided by the software application.

When I want to verb *[object|subject] [other expression]*
Then register verb *[object|subject]* as a new Verb symbol.

After applying this rule, the following verbs are obtained:

Verb: create a new game [US#1]

Verb: invite estimators [US#2]

Verb: join a game [US#3]

Verb: start a round [US#4]

Verb: see the item [US#5]

Verb: edit an item [US#6]

Verb: export a transcript [US#7]

Rule 4. Behavioral identification for Verbs. The rest of the expression that is present in the "I want" sentence that gives origin to the verbs symbols of rule number 3 ([other expression]) must be used as behavioral response of the verb symbol. The rest of the expression describes how the actions should be performed. This description belongs to the behavioral response of the LEL glossary.

When As a role I want to *verb [object / subject] [other expression]*
Then register *[other expression] as a new* behavioral response to *verb*.

For example, the verb *creates a new game* appears in US#1, and the rest of the *I want* sentence states *by entering a name and an optional description.* This expression must be considered as behavioral responses of the verb *create a new game*.

Verb: create a new game [US#1]
Behavioral responses: by entering a name and an optional description [US#1]

Verb: invite estimators [US#2]
Behavioral responses: by given them a URL where they can access the game [US#2]

Verb: join a game [US#3]
Behavioral responses: by entering my name on the page I received the URL for [US#3]

Verb: start a round [US#4]
Behavioral responses: by entering an item in a single multi-line text field [US#4]

Verb: see the item [US#5]
Behavioral responses: we are estimating [US#5]

Verb: edit an item [US#6]

Behavioral responses: in the list of items to be estimated [US#6]

Verb: export a transcript [US#7]
Behavioral responses: of a game as a CSV file [US#7]

Rule 5. Notion identification for Verbs. The notion of the Verb symbol must be extracted from the "so that" clause. This clause describes the objective that the User Story pursues, this is also the aim of the notion attribute in the LEL glossary.

When I want to *verb [object / subject] [other expression]* so that *[expression]*
Then register *[expression] as a notion* to the verb.

The US#1 contains *I want to create a new game, and the* "so that" sentence states that *I can start inviting estimators*. This expression must be used as the Notion. The following examples show the Verbs with the full description (Notion and Behavioral Responses) obtained from the previous rules.

Verb: create a new game [US#1]
Notion: can start inviting estimators [US#1]
Behavioral responses: by entering a name and an optional description [US#1]

Verb: invite estimators [US#2]
Notion: can start the game [US#2]
Behavioral responses: by given them a URL where they can access the game [US#2]

Verb: join a game [US#3]
Notion: can participate [US#3]
Behavioral responses: by entering my name on the page I received the URL for [US#3]

Verb: start a round [US#4]
Notion: can estimate it [US#4]
Behavioral responses: by entering an item in a single multi-line text field [US#4]

Verb: see the item [US#5]
Notion: know what I am giving an estimate for [US#5]
Behavioral responses: we are estimating [US#5]

Verb: edit an item [US#6]
Notion: can make it better reflect the team's understanding of the team [US#6]
Behavioral responses: in the list of items to be estimated [US#6]

Verb: export a transcript [US#7]
Notion: can further process the stories and estimates [US#7]
Behavioral responses: of a game as a CSV file [US#7]

Rule 6. Object identification. Once Verbs has been defined, the object element that follows the verb is extracted and registered as a symbol. Some verbs are transitives, that means that the action relies on some object. The object mentioned in the User Story could be implemented as a database entity or a class, and it should be registered in the LEL glossary.

When I want to verb object
Then register object as a new Object symbol.

For example, the expression *creates a new game* contains the object *game*. The LEL symbols obtained are the following: *game, round, item* and *transcript*.

Object: game [US#1] [US#3]

Object: round [US#4]

Object: item [US#5] [US#6]

Object: transcript [US#7]

Rule 7. Behavioral identification of Object. From those expressions that give rise to Objects ("verb [object]"), a new Behavioral Response must be added. The expression describes the actions performed with the Object. This is the aim of the behavioral response attribute in the LEL glossary.

When As a role I want to verb object *[. . .]*
Then register role verb [object] as an object' behavioral response.

This information is also present in the subject who performs the action. Nevertheless, it must be used to describe the object too. For example, the action *creates a new game* is performed by the moderator and this information is described in US#1. This expression must also be included in the behavioral responses of the *object game*. Six of the User Stories has expression *subject verb [object]*. Thus, the four objects identified are described in the following way:

Object: game [US#1],[US#3]
Behavioral responses:
The moderator creates a new game [US#1]
The estimator joins a game [US#3]

Object: round [US#4]
Behavioral responses:
The moderator starts a round [US#4]

Object: item [US#5],[US#6]
Behavioral responses:
The moderator edits an item [US#5]
The estimator sees the item [US#6]

Object: transcript [US#7]
Behavioral responses: The moderator exports a transcript [US#7]

Rule 8. Additional behavioral identification of Subjects. When a Verb is followed by a predicate that also includes a subject, the sentence is used as a behavioral response of subject that is mentioned in the predicate.

When As a role I want to verb subject *[. . .]*
Then register role verb [subject] as a behavioral response of the subject.

This rule is similar to rule number seven but it deals with subjects instead of dealing with objects. The expressions *role verb [subject]* must be used to enrich the behavioral responses of *[subject]*. In general, User Stories follow the structure *As a role I want verb object* describing an action that a role wants to perform on an object. That is, a user of the application wants to execute some functionality on an element of the business. Nevertheless, it can be possible that a user of the application wants to execute some functionality that involves another role (and not an element). In this case, the second role involved in the functionality is a passive role. And this rule has the aim to record this passive functionality as a behavior of the second role. There is only one expression with this format *moderator invites* *estimators.* Thus, the subject *estimator* must be enriched with one more expression in its behavioral responses:

Subject: Estimator [US#3],[US#5]
Behavioral responses:
The estimator joins a game [US#3]
The estimator sees the item [US#5]
The moderator invites estimators [US#2]

The result of the rules' application is available at a public repository [1]. It describes all the LEL symbols and their descriptions. Some expressions are underlined in order to show the links between the symbols (circularity principle).

5. Quantitative evaluation

The experiment is motivated by the need to understand the performance of subjects who use User Stories and the LEL approach presented in this paper. Thus, we defined the following research questions:

- RQ1: Does LEL improve the correctness of requirement traceability tasks?
- RQ2: Does LEL help improving the performance of requirement traceability tasks?
- RQ3: Does LEL reduce the complexity of the requirement traceability tasks?

In the next subsection, we detail the most important elements of the evaluation and the preliminary results based on the Wohlin [33] and Juristo[34] guidelines for reporting empirical evaluations.

5.1. Goal definition

The main goal of this experiment is: *analyse User Stories enhanced with an LEL glossary for the purpose of evaluating the correctness, complexity, and performance of developers and analyst in requirements traceability tasks. The experiment was conducted from the point of view of researchers in the context of professional software engineers.*

5.2. Research Questions, Hypotheses, and Variables

This work aims at answering the following Research Questions (RQ).

RQ1: Does LEL improve the correctness of requirement traceability tasks?

For this RQ, we consider as the null hypothesis (H_0) that there is no difference in the correctness of requirement traceability tasks performed by subjects. In one case, we evaluated the use of User Stories and, on the other case, the use of User Stories with the LEL. The recall and precision metrics, which were borrowed from the information retrieval research field, are used to assess how many relevant data items are identified by the subjects, given a gold standard that was manually produced by the authors of this work. For the metrics computation, we classified the answers in: True Positive which is the correct identification of a relevant element (Eq. 4), the False Negative which is the wrongly omission of a relevant element (Eq. 7), the True Negative which is the correct omission of irrelevant elements (Eq. 5) and False Positive which the wrongly identification of irrelevant elements (Eq. 6). Some reference values like Response Domain (Eq. 3), Relevant Response and Subject Responses. The first one references all possible values of the subjects' responses. The *RelevantResponse(aQuestion)* function receives as argument a question and returns the set of valid items (gold standard) that can be part of an answer. And the *Response(aSubject, aQuestion)* receives one subject (Software Engineer) and a question, and returns the subject's answer to the question received as a parameter. The Recall and precision formulas are presented in Equation 8 and 9 respectively. To measure the overall performance, we considered the accuracy metric. The accuracy metric is one of the most obvious metrics; it weights the correct answers. The Equation 10 shows that the metric considers all correct responses (the identification of relevant elements and exclusion of irrelevant ones) over possible answers of all subjects.

[1]Experiment material - `https://www.dropbox.com/sh/4upl1gn644zqu3m/AAAP4E6kxDbdU_tZcOM6APkqa?dl=0`

As the alternative hypothesis (H_a), we consider there is an improvement in the mean of the response correctness of subjects using LEL (μ_{LEL}) over the basic support of User Stories (μ_{US}): $\mu_{US} \leq \mu_{LEL}$. In this case, the correctness dependent variable was operationalized using the accuracy metric in order to understand the behavior of subjects using both techniques. We will use the precision and recall metric as auxiliary metrics to discuss in detail some results.

$$ResponseDomain = \bigcup_{s=1}^{n} Response(subject_s, aQuestion) \cup RelevantResponse \tag{3}$$

$$TruePositive = RelevantResponse \cap SubjectResponse \tag{4}$$

$$TrueNegative = ResponseDomain - SubjectResponse - RelevantResponse \tag{5}$$

$$FalsePositive = SubjectResponse - RelevantResponse \tag{6}$$

$$FalseNegative = RelevantResponse - SubjectResponse \tag{7}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{8}$$

$$Pecision = \frac{TruePositive}{SubjectResponse} \tag{9}$$

$$Accuracy = \frac{TruePossitive + FalseNegative}{ResponseDomain} \tag{10}$$

RQ2: Does LEL help improving the performance of requirement traceability tasks?

The performance has been studied considering the time it takes to answer the questions of the survey. We consider as null hypothesis H_0 that there is no difference in the performance between both approaches when performing requirement traceability tasks. The response variable is the performance, and the time required by the subjects for accomplishing the mentioned tasks is the metric.

As alternative hypothesis H_a, we consider there is an improvement in the mean of the time required to undertake the tasks by the subjects using LEL (μ_{LEL}) over the primary support of User Stories (μ_{US}): $\mu_{US} \leq \mu_{LEL}$.

RQ3: Does LEL reduce the complexity of the requirement traceability tasks?

The complexity of the tasks is an imperative aspect measured in this experiment because it depicts whether or not the LEL usage may interfere with the predisposition to use the LEL. If the LEL increases the complexity of the task, the subject would not likely use the LEL again in the future. The complexity was measured with an interval from 1 to 5 where 1 is the lowest difficulty and 5 is the greatest difficulty. The metric was reported by each subject using a questionnaire. We consider as null hypothesis H_0 that there is no difference in the complexity between both approaches when performing requirement traceability tasks.

As alternative hypothesis H_a, we consider there is a decrease in the mean of the task complexity reported by the subjects using LEL (μ_{LEL}) over the basic support of User Stories (μ_{US}): $\mu_{LEL} \leq \mu_{US}$.

The Research questions were evaluated in different traceability tasks on User Stories and LEL. The result of each task was analyzed considering a leading case to compute the precision, recall, and accuracy. We defined a set of questions that lead to traceability tasks:

1 What are the roles of the systems?

2 What are the functionalities permitted to a given role?

3 What role can perform a specific X functionality?

4 What business entity can be found in the set of requirements?

5 If a new requirement related to a business entity arises, what user stories must be reviewed to check the consistency?

The questions defined are related to the information that the User Stories provides and should be traced to the software application. The User Stories describe roles and their functionality. Thus, questions 1 to 3 explore traceability aspects of these elements. Then, in order to provide the functionality, developers need to identify business entities and implement them in the application. Thus, questions 4 and 5 deal with this issue.

5.3. Experiment design

In order to answer the research questions, we designed a completely randomized experiment where subjects were asked to perform a set of requirement traceability tasks and the outcome was evaluated in order to get different metrics: precision, recall, accuracy, time spent and complexity of the task. The factor is the provoked variation which is controlled by the researchers in order to measure its consequences, and the possible values of the factor during the experiment are named treatments. In this case, the experiment was a Complete Randomized Design. The subjects were randomly divided into two groups for the two alternatives of the approach (the factor of the experiment): User stories (Control) and User Stories plus LEL glossary (Treatment). We used in both treatments hard copy material.

5.4. Experiment objects

Based on the research questions, we produced different experimental objects. We provided the subjects three printed documents:

a) A questionnaire containing consent form to agreeing to participate in the experiment allowing them to withdraw from the experiment, demographic questions and questions to realize the professional experience. We made clear the information gathered is confidential.

b) For the Control alternative of the experiment performed by a group of subjects, we provided the requirement specification of a system honoring the User Story template. That is, we provided 71 User Stories described according to the template “As a [role] I want [goal / desire] So that [reason].” The User Stories described functionality about a course management system with features about offering courses, enrolling the student, and setting up the classes.

c) For the group performing the Treatment alternative, an LEL glossary derived from the set of User Stories used in *b).*

d) Finally, a final questionnaire with the traceability questions, which was abovementioned in subsection 5.2, was required to be fulfilled by the subjects. Based on the analysis performed by the subjects using either User Stories (*b*) or an LEL glossary (*c*), they were asked to answer questions that require to trace and review the requirements, and understand what entities, functionalities, and roles the system supports.

We decided to conduct the experiment relying on printed material in order to avoid threats to validity that could arise from using a digital tool.

In figure 3, we show an overall schema of the experiment introducing experimental units, metrics, and data sources used during the analysis.

5.5. Subjects

The subjects were 36 software engineers from different software companies and institutions. On average, they had 8,3 years of programming experience and 6,82 years in requirement analysis tasks.

Two groups of 18 subjects were formed where one group received the material related to the control case (User Stories) and the other received the material related to the treatment case (LEL, User Stories plus LEL). They were motivated and committed to the experiment, as we were sponsored by the CEOs and managers that notified the subjects about the organization’s commitment to the research experiment.

Regarding the academic degree, the 85% of subjects have a degree (bachelor, master or Ph.D.). The reader must note that bachelor degree in South America is comparable to a European Master Degree because of its curricular definition.

The subjects belong to different organizations where they actively practice software engineering from Argentina and Spain. Next, we describe them briefly introducing the organizations:

- LIFIA is an Argentinean research center of the National University of La Plata that offers its expertise in technology and products to academic communities and national and international business.
- ITW2 is a Spanish research group that has participated or participates in various projects for transfer of research results and I+D+I and it is focused on software testing, project management and software quality assurance, among others.
- Fiscalia de Estado is a government office that belongs to Buenos Aires Province, Argentina, and has two main goals: (i) defend the interest of the province and (ii) control that every administrative act is performed under a legal context. It has an IT department with 30 members approximately.

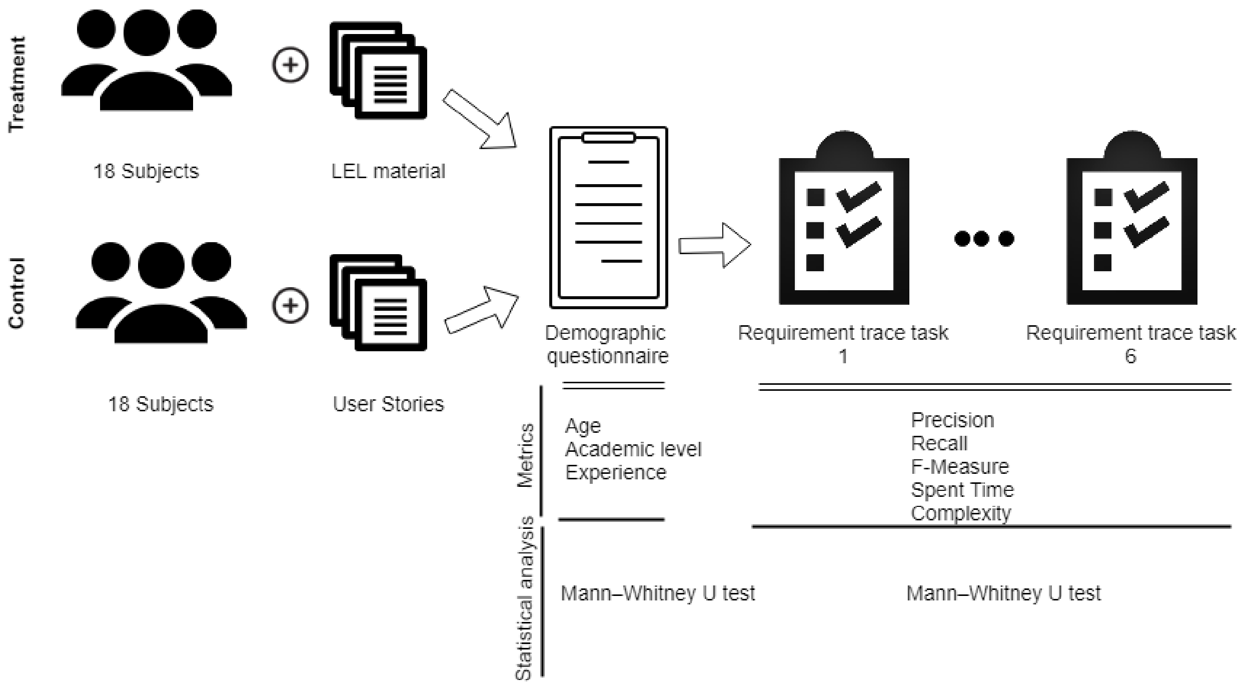


Figure 3: Overall schema of experiment

5.6. *Experiment execution*

The experiment protocol was executed in the same way by all the subjects. First of all, they received a brief introduction to the material which had to be used during the experiment. In the case where an LEL glossary was required, the subjects were trained about the LEL approach. Next, participants were asked to complete an expertise survey, read the experiment description, study the requirements, and fill out the questionnaire form that forces to perform some trace tasks.

Each subject performed fully the experiment supervised by a researcher who monitored the execution ensuring similar facilities layout, infrastructure, and subject isolation conditions. Additionally, the researcher controlled the subject to avoid any answer modification as long as she advanced in the experiment.

To achieve the task of processing the collected results, we first processed and digitalized responses. Then, we used different scripts to automate the data processing task.

5.7. *Data processing and reduction*

In order to achieve the task of processing the collected results, both automatic as well as manual processing were used.

We wrote a tool for analyzing the samples obtained for each treatment. The result of such processing is a statistical analysis of samples using proper hypothesis testing technique, and the results are presented in the next sections. The tool was written using R language (version 3.5.1) that takes as input the samples, automatically applies transformations and normalizations on data items, and evaluates the hypothesis presented in this section.

The outliers are atypical measurements that can influence the analysis. The false data points may lead to draw an invalid conclusion. To avoid this situation, outliers were removed from the dataset after their analysis. Firstly, we identified outliers using InterQuartile Range (IQR) technique [35]. The IQR is the difference between the 75th (Q_3) and 25th (Q_1) percentiles. Then, those values above Q_3 + 1.5 times IQR and below Q_1 − 1.5 times IQR are identified as candidate outliers. Then we excluded those samples which measured 0 or 1 for accuracy when these values were identified as outliers. After reviewing all the candidate outliers, we realized that the outliers different to 0 and 1 should be considered as they represent the subject experience using LEL or not.

5.8. *Analysis*

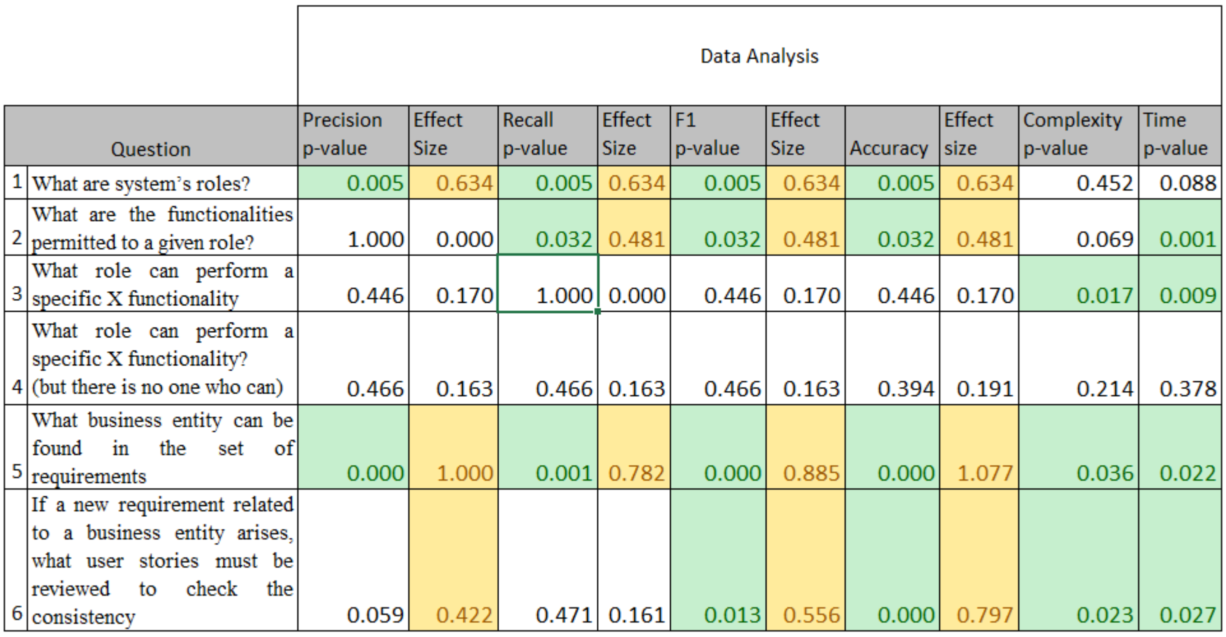
The analysis was divided into subsections. In the first one, we analyzed the data distribution for selecting the most suitable hypothesis testing technique. Secondly, we analyze the results of each trach traceability tasks. Finally, we evaluate the statistical significance of the samples supporting or rejecting the null hypothesis.

5.8.1. *Preliminary analysis*

For the analysis of samples, firstly, we defined valid responses for the requirement traceability questions that subjects were required to answer. For example, the gold standard Reference Values (RV) for the "*What are the roles of the system*?" was defined in order to compare it against the subjects' answer. This gold standard was performed by some authors of this paper based on their experience on the domain. Once samples were digitalized, we checked each response against the corresponding RV for every sample obtaining the recall and effort metric mentioned in Subsection 5.2 .

In Table 2, we present the results of the processing of the samples where the first cluster of columns reports the normality test outcome. The normality testing helps to check whether the samples fit into a normal distribution. Those samples that fit are known as parametric, and those that do not fit are named non-parametric. The distribution constraining the hypothesis testing technique to be used. We used Shapiro-Wilk [36] normality test that reported p-value lower than 0.05, therefore, samples do not come from a normal distribution. Consequently, the second cluster of columns groups the metrics used in the analysis which were processed using Mann–Whitney U test [37]. That is a a non-parametric statistical hypothesis test technique and we considered a standard confidence level (α) of 0.05. Additionally, we computed the effect size using Cliff's Delta technique [38].

Table 2: Experiment results



		Data Analysis									
	Question	Precision p-value	Effect Size	Recall p-value	Effect Size	F1 p-value	Effect Size	Accuracy	Effect size	Complexity p-value	Time p-value
1	What are system's roles?	0.005	0.634	0.005	0.634	0.005	0.634	0.005	0.634	0.452	0.088
2	What are the functionalities permitted to a given role?	1.000	0.000	0.032	0.481	0.032	0.481	0.032	0.481	0.069	0.001
3	What role can perform a specific X functionality	0.446	0.170	1.000	0.000	0.446	0.170	0.446	0.170	0.017	0.009
4	What role can perform a specific X functionality? (but there is no one who can)	0.466	0.163	0.466	0.163	0.466	0.163	0.394	0.191	0.214	0.378
5	What business entity can be found in the set of requirements	0.000	1.000	0.001	0.782	0.000	0.885	0.000	1.077	0.036	0.022
6	If a new requirement related to a business entity arises, what user stories must be reviewed to check the consistency	0.059	0.422	0.471	0.161	0.013	0.556	0.000	0.797	0.023	0.027

5.9. *Evaluation of results and implication*

In this section we will discuss the results and their implications. The results for LEL and US experiments can be found as an extra material [1] with the gold standard used to compare the samples. The data analysis outcomes that the LEL usage presents benefits in front of using User Stories alone. In five of six requirement traceability tasks which require to browse the requirements, the use of LEL reported better results in different aspects improving the response correctness (RQ1), requiring often less time than the use of User Stories to answer a question (RQ2), and implies the complexity of the tasks (RQ3). However, the use of LEL did not present benefits in the precision of the answers. It is noteworthy that most of the hypothesis testing results have a mindful size effect with values between 0,422 and 1,077 when the p-value is under the alpha level and low effect size for that p-value above alpha level. Regarding the Type II error, we mention the power of the test for those values over alpha level. The experiment design was simple but effective because it was a complete randomized design where subjects worked isolate using one approach (factor alternative). The experiment required a large number of subjects because a subject was not exposed to the two approaches in different crossed sessions.

In the experiment, we asked the subjects to perform traceability tasks so as to know how the user stories and LEL perform together.

In Table 2, we present the analysis of the samples reporting the hypothesis testing result (p-value and size effect) of precision, recall, accuracy, complexity, and time metrics. A p-value calculates whether or not there is a significant difference between the approaches (treatments). So it is used to reject the null hypothesis. Additionally, we study the magnitude of the difference between both approaches using the size effect [34]. We use For recall, precision and accuracy, the p-value evaluates if the LEL usage mean was greater than User Stories alone ($\mu_{US} \leq \mu_{LEL}$). On the other hand, the p-value tests if the task complexity mean and spent time using LEL were lower than User stories alone ($\mu_{LEL} \leq \mu_{US}$).

Moreover, we present in Figure 4a to 4f the graphs corresponding to the precision and recall results of LEL and User stories approaches. In each diagram, the results of performing management tasks are shown using a green triangle for LEL results and a red circle for User Stories results. A label is attached to each point counting the count of overlaying results at that point.

Next, we are reporting a thorough analysis of the traceability tasks along with the interpretations of the graphs corresponding to the precision and recall result (figures 4a to 4f).

5.9.1. What are the roles of the system?

The roles of the system are identified from the section “As a ...” of user story declarations denoting the role of a user in the system. In the case of LEL approach, the roles are Subjects symbols named with the user role. The mean proportion of valid system roles among the answered roles in the response is higher when using LEL; the responses include valid system roles and non-valid ones (those wrongly reported). We can support the alternative hypothesis using the accuracy a *p-value* 0,005 (*r* 0,634). The results are shown in Figure 4a presenting the two parameters required to compute the F-measure: precision and recall. The LEL results are placed further to the right (denoting a better precision) and high (denoting a better recall) than User Story results. As it is shown in the figure, the green triangle pointing LEL values are solid green because in that location all the values overlap. Red circles used to locate User story values are scattered about a precision 1 but scattered in a wide recall range from ~0.3 to ~1.0. The graph shows an improvement in the precision and recall metrics of LEL over User Stories alone.

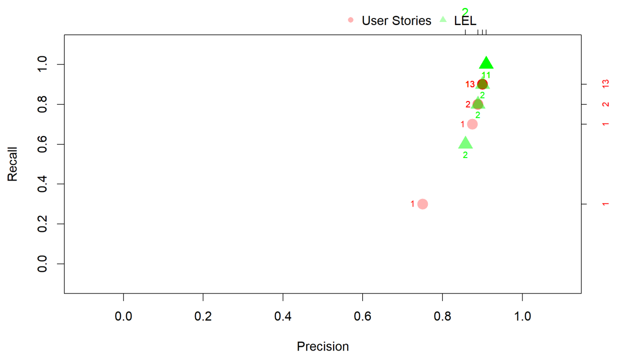
The LEL glossary structures the information grouping the symbol type (subject, verb, object, and state) and sorted by name. Therefore, it makes it easier for the user of the LEL to look up information. Oppositely, the roles of the system are scattered in user stories, grouped in Sprints and sorted by its sprint's date which requires checking each user story in order to reach the expected information.

The evidence shows there is not a remarkable complexity and performance improvement in favor of LEL. The subjects' perception of the task complexity using LEL is not much better than User Story accepting the null hypothesis. Moreover, the time required to accomplish a task using the LEL is slightly lower but no statistically significant (p-value 0,085), so the alternative hypothesis can not be supported.

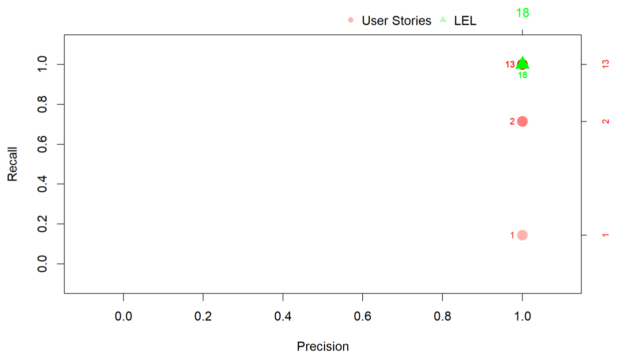
5.9.2. What are the functionalities permitted to a given role?

The available functionality to a given role is defined by the clause “... I want to....” in the user stories. In the case of LEL, the symbols of subject type have references to the user stories which have the subject in their “As a..” clause. The Figure 4b shows the precision and recall values which are far-right and high having all of the values overlapped at the same point, and User Story results are scattered mostly vertically, variating the recall. In this task, the accuracy of LEL mean is higher than User story mean when looking up the functionality being available for a given user considering the F-measure metric (*p-value* 0.032, r 0.481).

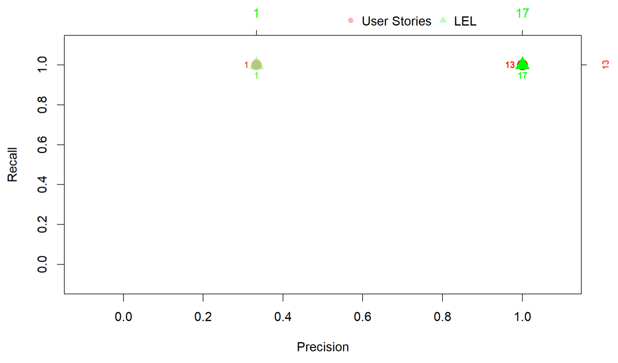
Figure 4: Precision and recall results for the traceability tasks



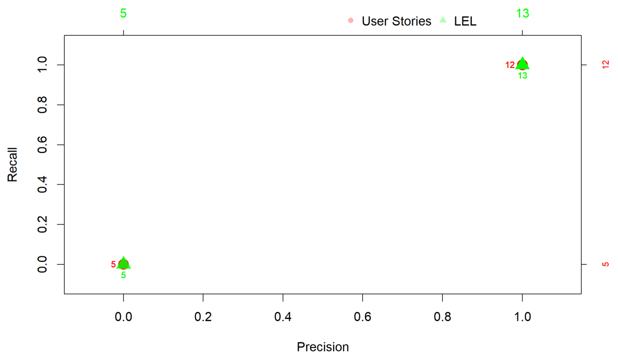
(a) What are the functionalities permitted to a given role?



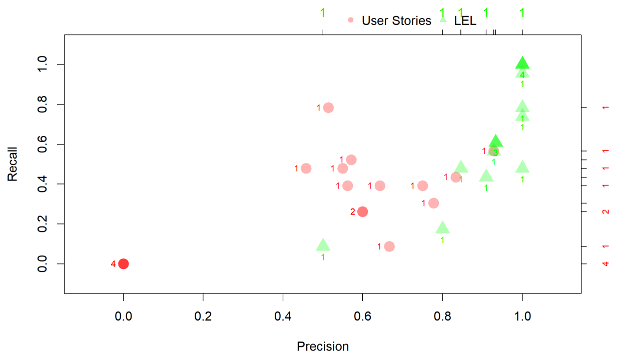
(b) What are the functionalities permitted to a given role?



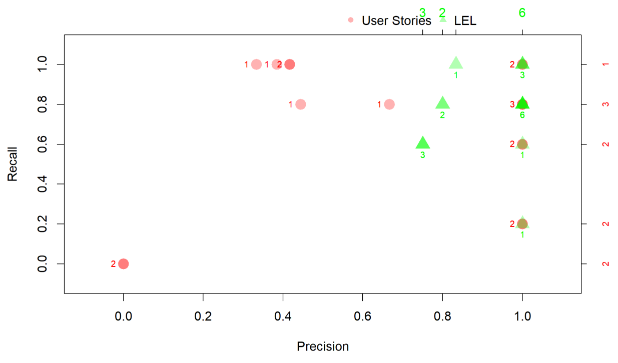
(c) What role can perform a specific X functionality?



(d) What role can perform a specific X functionality? (but there is no one who can)



(e) What business entity can be found in the set of requirements?



(f) If a new requirement related to a business entity arises, what user stories must be reviewed to check the consistency

Regarding the time required to accomplish the task, the LEL approach mean for the time required to complete the task (*p-value* 0,001) was less than the User story approach alone. The complexity using LEL did not show a difference because the *p-value* 0,085 is larger than the chosen alpha level. So, we can not reject the null hypothesis.

5.9.3. What role can perform a specific X functionality?

Often analysts or developers require to find information about specific functionality that may or may not exist in the system. When using User Stories alone, the lookup of information about a particular functionality requires, in the worst case, to browse each User Story. On the other hand, when using LEL there are few options to accomplish this task. She can either perform a search based on the role at issue like the previous case using the subject symbol, or search the verb symbol representing functionality being reached and then browse linked user stories to find what users can perform the functionality. In both LEL alternatives, the analyst only browses a subset of available user stories opposite of User stories where all user stories must be checked.

We asked the subjects two questions to search what system role can perform a given functionality. In the first one, there was actually possible to perform the functionality by a user, and, for the second question, there was no user who can perform the noted functionality.

The accuracy result is not statistically significant (*p-value* 0.446, *r* 0.170). That is to say, we can not reject the null hypothesis.

Figure 4c shows that LEL answers are further to the right than User stories which denotes a better precision. Most of the LEL results are overlapped further to the right and high. Oppositely, User Stories reported opposite and polarized results where much of them are further to the right and high – the best possible performance- and a few of them are placed further to the left and bottom – the worst possible performance-.

5.9.4. What role can perform a specific X functionality? (but no role can perform the X functionality)

In opposition to the previous question, this question asked subjects to find system users that can perform a specific task but there is no user who can do it. This question is tricky because subjects expected to find one and spent much time reviewing the whole requirement list in both experiments. That is to say, they checked each user story and LEL symbol. There was no significant difference between approaches in the precision, recall, task complexity, and effort metrics. So, we can not reject the null hypothesis. Supporting the hypothesis testing, Figure 4d shows that both LEL and User Story results overlap with polarized and opposite results.

5.9.5. What business entity can be found in the set of requirements?

Business entities are scattered through the user stories in different clauses: "I want to..." and "So that ...". On the other hand, Business entities are registered in the LEL glossary as a system object.

In this task, the LEL usage has an effect on the results, with LEL answers providing better performances. The answers reported a better ratio of valid entity among all entities enumerated, and a better ratio of valid entities listed against all possible valid entities. The accuracy was (*p-value* 0.001, *r* 1.077) denoting that LEL approach answers were more effective than User stories.

Subjects reported that the complexity was lower (*p-value* 0.036) and the recorded time was lower too when using LEL approach (*p-value* 0.022). The Figure 4e presents results scattered heterogeneously in the graph. The LEL results were located further to the right and higher than User stories result.

5.9.6. If a new requirement related to a business entity arises, what user stories must be reviewed to check the consistency?

One of the most difficult tasks during the requirement analysis is ensuring requirement consistencies. In the case of the LEL using a User Stories, the analyst must browse all requirements for such a goal. Meanwhile, when using LEL the analyst can use subject, verbs, or objects symbols to browse those user stories which mention such symbol. This task showed a better performance of those subjects that used LEL technique for the accuracy (*p-value* 0.000, *r* 0.707). Additionally, the reported complexity perception was lower (*p-value* 0.023) and required time to accomplish the task was also lower than user stories (*p-value* 0.028).

Figure 4f shows two patterns in the chart. One of them is an overlapping column of values with precision about 1. In that case, there is no difference in the responses. However, the remaining values are placed further at the right in the LEL case valuating about 0.8 of precision and Use case values are located at about 0.4 of precision.

5.10. Replication

In order to replicate the experiment, we promote an exact replication [39] by following the steps mentioned in this section and using the material provided.

The material above mentioned is available at a public repository [1] and can be used to reproduce the experiment in the future by other researchers.

5.11. Threats to validity

There are several threats to validity considered during the experiment design. We use the Wohlin [33] classification of threats to validity to discuss the impact on the experiment of different aspects.

Internal validity. These threats influence independent variables without researcher awareness. To avoid this kind of influence during the experiment, we presented each material in a brief introduction before subjects performed the experiment, and during the experiment, any inquiry related to the sentences was answered. The subjects were selected randomly from the volunteer group relieving *Statistical regression* threat, and all of them were working in software companies in Argentina and Spain. The provided material was the same for all subjects. We also checked that all the users had a computer science background.

We reduced the possibility of a *maturation* threat because only one treatment was applied to every subject per sample. This experiment design also minimizes the *testing threat* because there are not repeated measures performed by the same subject. Regarding the learning effect, we can also mention that the subjects did not have prior experience with the LEL concept. Indeed, they were introduced for the first time to the technique during the experiment. Furthermore, only six tasks were assigned in the same order to all subject so they were exposed to the same familiarity conditions in both contexts: with and without LEL.

The experiment was performed on a regular working day which minimizes the *history* threat.

Regarding the Instrumentation threat, the material was presented in a printed document and, once the result was registered in a database, the outcome processing was performed with Python script.

The subjects were volunteers setting a homogeneous group of subjects for the experiment which alleviates the *selection* threat. Despite the fact that the subjects were motivated alike for both alternatives of the factor, the experiment conclusion could be affected because they were volunteers.

Conclusion Validity. We have considered different aspects during the experiment design and execution to avoid undesired effects that constrain the drawing of a correct conclusion. To avoid a *low statistical power* threat, we conducted the experiment with 18 subjects per treatment which gives a good effect size value.

We reduced this threat using the Wilcoxcon non-parametric test. The non-parametric tests have fewer constraints than the parametric ones but make it more complex to compute the effect size. During the hypothesis evaluation, we reviewed several times the assumptions of the statistical test in order to avoid a violation (*Violated assumptions of statistical tests*). Regarding *Fishing and the error rate*, the experiment was based on objective metrics evaluated with all gathered data without any exclusion to guaranty that the outcome of the experiment analysis will be the same avoiding hypothesis fishing. Moreover, we conducted the experiment in different settings and different executors to gather independent measures. The *Reliability of measures* was considered using scripts that process the information automatically what avoids errors from manually processing the data. The data was gathered from subjects with the same profile (*Random heterogeneity of subjects*) using documentation and questionnaires which make possible to reproduce the experiment with no modification between each run (Reliability of treatment implementation), We used well-known guidelines for reporting empirical experiments as checklists for confirming the requirements of the test techniques. Finally, in order to avoid the impact of random irrelevancies on the experiment, we used a set of samples providing a significant power which helped the irrelevancies to become diluted.

Construct validity. This work presents a preliminary result that supports our claims. Some aspects were considered regarding the generalization of the results. In Section 5.3, we explained the reasons for the selected experiment design. All the minimum required details based on well-known reporting guidelines for an empirical work report were described such as goal, hypothesis, metrics, protocol, materials, Data processing, and threats to validity. Every subject was assigned to only one treatment which avoids threats from the *interaction of different treatments*. The evaluation of two different approaches avoids a *mono-method bias* threat. The provided material provided dozens of user stories which may reduce a *Mono-operation bias*. However, it would be necessary to perform the experiment considering a different domain-problem.

All participants were aware of the experiment setting; thus any behavior modifications such as concentration or concern could be present. Therefore, the experiment suffers Interaction of testing and treatment threat. We rely on a large set of samples to reduce the effect of this situation when drawing the conclusion.

The results reported in this work show the improvement registered on productivity and performance in the trace-based task of requirements. However, these should not be generalized to other activities related to the requirement engineering or software development process.

The experiment suffers from several other threats of construct type. Although we did not mention the research questions, the subjects may guess the aim of the experiment (Hypothesis guessing).

External validity. The subjects were software engineers who have played the role of developers and analysts during their careers. Although their experience levels were different, they have been exposed to the regular responsibilities of any software practitioners: meet with clients, understand requirements, develop the software, and honor deadlines for software delivery. A broader experiment considering different subjects of different cultures who have worked on different business domains will improve the generality of our claims.

The experiment is affected by the *Interaction of setting and treatment* threat. Printed questionnaires and material are not often used in the industry. We used this old-fashioned supporting tool in order to avoid an up-to-date web-based tool which would mean a new independent variable in the experiment and making it more complex. Interaction of history and treatment threat affect the experiment because we were not able to replicate real-life scenarios where the requirement tracing activities are performed; for example, an office with a sharp deadline arriving.

5.12. Data analysis

In this section, we perform a data analysis to evaluate the results of the assessment.

Regarding the first research question "*Does LEL improve the correctness of requirement traceability tasks?*", the correctness was measured in 6 different requirement management tasks. The evaluation highlights that the LEL approach helps to resolve the tasks with better precision than User stories. Four out of six tasks were resolved by subjects with a higher rate of true positive responses (valid results for the task) and a lower rate of false-positive responses (invalid results for the task). Tasks 3 and 4 did not result in a statistically significant difference in the performance of both approaches. In task 3, the result showed that LEL usage has a strong influence but not statistically significant in the lookup of roles that can perform a given task. Moreover, for the 4th task, the subjects were required to look up a system role able to perform a functionality but there is no system role valid as answer. Thus, it would require to perform further studies because the size effect *r=0,166* denotes a poor power and a chance to suffer a Type II error supporting a null hypothesis when we must not.

The preliminary result allows supporting that LEL provides benefits in tracing requirements information while performing requirement management tasks. Although results are preliminary and the experiments report benefit in some tasks, we consider that the benefits are valuable. Requirements are critical in software development since the software must satisfy and implement functionality according to them. If requirements are incorrect, the software will be useless. Requirements tasks are prone to errors because it means dealing with too much information, thus make harder the need to find the correct description of requirements. Providing a tool to improve the results of the activities will make it possible to improve the quality of the software because the team will have access to the right requirements when they need it.

The second research question "*Does LEL help improving the performance of requirement traceability tasks?*", the usage of LEL reported benefits in the time spent to accomplish the tasks in 4 of 6 tasks (Tasks 2, 3, 5 and 6). The cases of identifying system roles (Task 1) and detecting if there is one functionality available for a role but there is none (Task 4), the analysis denotes there is no significant difference from the statistics point of view in the performance of subjects using user stories and the LEL. Although tools help to find the correct information fast, agile development relies on a combination of automatic and manual support. The board with User Stories is essential, as well as, some backup of that information in a tracking application. Thus, in this context, our proposal provides a tool to improve performance when looking for some information related to requirements. The LEL built can be printed and can also be navigated digitally.

Finally, the third research question "*Does LEL reduce the complexity of the requirement traceability tasks?*" produced interesting results. The complexity perceived by the subject about the required task, the analysis denotes that three of six tasks (Tasks 3, 5 and 6) reported that LEL was less complex than using the User Story technique. The experiment showed that the complexity perceived was better in the same tasks that their execution was faster. This result in combination with the performance result reinforces the applicability and usability of the proposed approach.

6. The Tool

We implemented a prototype tool that obtains the LEL from the User Stories written in natural language. We present the prototype as a preliminary work that shows that it is possible to provide support to apply our approach. We plan to perform a further validation of the tool as well as a validation of the approach based on the tool.

We used the Stanford CoreNLP parser [40] as the base of the natural language processing. This parser works with the structure of the grammatical sentences given as input, and obtains as output the *Subjects, Objects,* and *Actions* of every sentence that composes a User Story that must be described using the structure described in Section 3.2: "**As** [role] **I want** [goal / desire] **So that** [reason].". In this context, we can not tackle the problem of processing sentences manually because the nature of the requirements requires to consider the voice of the sentence, plurals, nouns and adjectives. which makes it difficult to develop an ad-hoc tool. The use of NLP provides several benefits. First, NLP allows performing Part-Of-Speech tagging so that the verb roots, compound nouns, possessive, and other complex linguistic aspects are detected. In this way, ambiguities or similarities can be resolved considering the semantics [19]. Additionally, nowadays state-of-the-art frameworks support extension mechanisms to add custom tagging features. The prototype is an executable application that reads the data (User Stories) as raw text stored in a file, and generates the corresponding LEL in a variety of formats: terminal output, text file, or an HTML site. The source code is available at [2].

A single text file can contain many User Stories with the following format: *user story id* (an alphanumeric value that identifies the User Story), a line break, and the user story (written with the format previously described). The LEL Parser evaluates the text and transforms the input data in Java objects. Then, using CoreNLP, each User Story is analysed with the purpose of obtaining all the grammatical components (subjects, objects and actions). Finally, the LEL Parser implements a series of *formatters*, which obtains output in different human-readable formats, such as Java terminal output, text file or HTML. The processing is straightforward; we present in the algorithm 2 the above-mentioned procedure.

Algorithm 2 Pseudo-code for User Stories processing

```
1:  function PROCESS(listOfUserStories)
2:      $LEL = \emptyset$
3:      for $US \in listOfUserStories$ do
                                /* Splits the text considering the user story template*/
4:          $asA\_Sentence, iWantTo\_Sentence, soThat\_Sentece \leftarrow split(US)$
                                /* Extracts subjects,verbs and Nouns from the sentence*/
5:          $subjectSymbols \leftarrow$ PROCESSNOUNS($asA\_Sentence$)
6:          $verbSymbols \leftarrow$ PROCESSVERBS($iWantToSentence, soThat\_Sentece$)
7:          $objectSymbols \leftarrow$ PROCESSOBJECT($iWantToSentence, soThat\_Sentece$)
                                /* Register symbols and links between them and USs*/
8:          REGISTERSUBJECTANDLLNK(LEL,$subjectSymbols$,US)
9:          REGISTEROBJECTANDLINK(LEL,$verbSymbols$,US)
10:         REGISTERVERBANDLINK(LEL,$objectSymbols$,US)
        return $LEL$
```

The Standford NLP framework provides several features to process documents. In this case we used the standard stemming and lemmatization provided by the framework as well as the POS tagging. We decided not to include NER processing because there were no entities referenced in the requirements. The processing engine called LELParser was wrapped with a service layers allowing different systems to profits from the symbols extractions feature. In order to provide a tool that gives supports to the concepts presented in this work, we have implemented a Redmine extension

[2]LEL Parser engine and Redmine plugin source code repository: `https://bitbucket.org/lellifia/lelredmineplugin/`

that enhances user stories registered in the tool with the derivate LEL. Redmine (www.redmine.org) is a platform used by many organizations to manage and control projects which rely on agile development methods. We developed a plugin that uses a web service which parses user stories content and returns a dictionary that collects all the relevant data extracted. In other words, it collects subjects, verbs and objects, associated with the original input in order to ease the subsequent processing.

The flow of a plugin operation is the following: (i) the Redmine triggers a user story creation event, (ii) it sends a request to the LELParser application with the User Story object, (iii) it receives the response and parses the body, getting subjects, actions and objects, and (iv) it relates the response with a user story object in the Redmine platform.

7. Conclusions

We have presented an approach to organize and consolidate the requirements scattered over different User Stories in diverse sprints into an integrated artifact: the LEL lexicon. We performed a preliminary evaluation to assess how the LEL produced from user stories contributes to reducing the drawbacks arising from lack of proper traceability and knowledge transfer needs. This artifact presents several advantages for complementing User Stories. First, it helps to summarize several aspects related to the system functionality such as roles, features and entities, as well as their usage contexts. These elements are pulled out from complex sentences and formalized in such a way information is clearly identified and organized. Besides, the LEL plays the role of a requirement index where a stakeholder can trace efficiently those User Stories that refer to a business element. For example, a developer has available all system functionality represented as symbols of category Verb and for each one he/she has available the references to every User Story that mentions the verb. This index, like in relational databases, saves time and effort by avoiding linear lookup of certain data along with Sprint definitions and its comprised User stories. Thus, our approach provides a semi-automatic tool to organize requirements in order to provide a semantic and functional scheme of the whole application. This scheme provides traceability and more important, it synthetizes the knowledge that requirements capture and can be used to transfer the knowledge between teams' members and stakeholders in an abstract and efficient way. We also validated the suitability of our approach (the quality of the LEL and the assistance it provides) with one experiment that helped to get quantitative information from the usage of both approaches. We showed, with statistically significant evidence, that the LEL approach performs better than just using User Stories in software management.

In the paper, we have presented how the requirement indexing using an LEL improves certain project management tasks. The LEL representation enriches stakeholder knowledge, improving User Stories, when software evolution tasks require an effective traceability.

The knowledge brought by LEL lexicons could be used to obtain new User Stories, as well as new products, for example: acceptance criteria. Thus, we envision an iterative process of building User Stories and LEL lexicon, where the approach may deal with contradictions and inconsistencies in requirements. As such, the set of management tasks may cover other aspects of the daily activities of the project manager.

The fact that the LEL lexicon is obtained from User stories through rules makes symbols prone to suffer from ambiguities or conflicts in symbols. We plan to introduce a consistency checking layer on top of our approach. This issue exists beyond the use of our proposed approach. Thus, our technique helps to discover these conflicts and after the identification they can be solved with further techniques. Lucassen et al. [19] provide a framework for ensuring the quality of user stories that can be adapted to the LEL approach. Moreover, we will study different usability aspects of the LEL lexicon automation to improve its use in requirements management.

[1] R. K. Wysocki, Effective Project Management: Traditional, Agile, Extreme, Wiley, 2013. URL `https://books.google.com.ar/books?id=RTBIAgAAQBAJ`

[2] F. P. Brooks, The Mythical Man Month: Essays on Software Engineering, Addison-Wesley Professional, 1995. URL `http://www.amazon.de/dp/0201835959`

[3] P. G. Neumann, The Risks Digest - Forum on Risks to the Public in Computers and Related Systems. URL `http://catless.ncl.ac.uk/Risks/`

[4] K. Kaur, A. Jajoo, Manisha, Applying agile methodologies in industry projects: Benefits and challenges, in: Proceedings - 1st International Conference on Computing, Communication, Control and Automation, ICCUBEA 2015, 2015, pp. 832–836. `doi:10.1109/ICCUBEA.2015.166`.

[5] E.-M. Schön, D. Winter, M. J. Escalona, J. Thomaschewski, Key Challenges in Agile Requirements Engineering, in: H. Baumeister, H. Lichter, M. Riebisch (Eds.), Agile Processes in Software Engineering and Extreme Programming, Springer International Publishing, Cham, 2017, pp. 37–51.

[6] K. Dikert, M. Paasivaara, C. Lassenius, Challenges and success factors for large-scale agile transformations: A systematic literature review, Journal of Systems and Softwaredoi:10.1016/j.jss.2016.06.013.

[7] J. Cleland-Huang, Traceability in agile projects, in: Software and Systems Traceability, Springer, 2012, pp. 265–275.

[8] M. Niazi, S. Mahmood, M. Alshayeb, M. R. Riaz, K. Faisal, N. Cerpa, Challenges of project management in global software development: Initial results, in: 2013 Science and Information Conference, 2013, pp. 202–206.

[9] R. Kasauli, G. Liebel, E. Knauss, S. Gopakumar, B. Kanagwa, Requirements Engineering Challenges in Large-Scale Agile System Development, in: 2017 IEEE 25th International Requirements Engineering Conference (RE), 2017, pp. 352–361. doi:10.1109/RE.2017.60.

[10] P. Lous, M. Kuhrmann, P. Tell, Is scrum fit for global software engineering?, in: 2017 IEEE 12th International Conference on Global Software Engineering (ICGSE), 2017, pp. 1–10. doi:10.1109/ICGSE.2017.13.

[11] C. Arora, M. Sabetzadeh, L. Briand, F. Zimmer, Automated Extraction and Clustering of Requirements Glossary Terms, IEEE Transactions on Software Engineering 43 (10) (2017) 918–945. doi:10.1109/TSE.2016.2635134.

[12] A. Dwarakanath, R. R. Ramnani, S. Sengupta, Automatic extraction of glossary terms from natural language requirements, in: 2013 21st IEEE International Requirements Engineering Conference (RE), 2013, pp. 314–319. doi:10.1109/RE.2013.6636736.

[13] T. Gemkow, M. Conzelmann, K. Hartig, A. Vogelsang, Automatic Glossary Term Extraction from Large-Scale Requirements Specifications, in: 2018 IEEE 26th International Requirements Engineering Conference (RE), 2018, pp. 412–417. doi:10.1109/RE.2018.00052.

[14] J. C. S. d. P. Leite, A. P. M. Franco, A strategy for conceptual model acquisition, in: [1993] Proceedings of the IEEE International Symposium on Requirements Engineering, IEEE, 1993, pp. 243–246.

[15] L. E. Wood, Semi-structured interviewing for user-centered design, interactions 4 (2) (1997) 48–61.

[16] A. d. P. A. Oliveira, J. C. S. do Prado Leite, L. M. Cysneiros, C. Cappelli, Eliciting multi-agent systems intentionality: from language extended lexicon to i* models, in: Chilean Society of Computer Science, 2007. SCCC'07. XXVI International Conference of the, IEEE, 2007, pp. 40–49.

[17] P. Mäder, P. Jones, Y. Zhang, J. Cleland-Huang, Strategies for effective traceability in safety-critical systems, IEEE Software 30 (2013) 58–66.

[18] S. Dimitrijević, J. Jovanovic, V. Devedžić, A comparative study of software tools for user story management, Information and Software Technology 57 (1) (2015) 352–368. doi:10.1016/j.infsof.2014.05.012.

[19] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, S. Brinkkemper, Improving agile requirements: the Quality User Story framework and tool, Requirements Engineering 21 (3) (2016) 383–403. doi:10.1007/s00766-016-0250-x.

[20] A. Mavin, Listen, then use ears, IEEE Software 29 (2) (2012) 17–18. doi:10.1109/MS.2012.36.

[21] X. Franch, Software requirements patterns - a state of the art and the practice, in: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Vol. 2, 2015, pp. 943–944. doi:10.1109/ICSE.2015.298.

[22] R. Fu, X. Bao, T. Zhao, Generic safety requirements description templates for the embedded software, in: 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), 2017, pp. 1477–1481. doi:10.1109/ICCSN.2017.8230353.

[23] G. Alaa, Z. Samir, A multi-faceted roadmap of requirements traceability types adoption in scrum: An empirical study, in: 2014 9th International Conference on Informatics and Systems, 2014, pp. SW–1–SW–9. doi:10.1109/INFOS.2014.7036675.

[24] M. Trkman, J. Mendling, M. Krisper, Using business process models to better understand the dependencies among user stories, Information and Software Technology 71 (2016) 58–76. doi:10.1016/j.infsof.2015.10.006.

[25] R. Darimont, W. Zhao, C. Ponsard, A. Michot, Deploying a Template and Pattern Library for Improved Reuse of Requirements Across Projects, in: 2017 IEEE 25th International Requirements Engineering Conference (RE), 2017, pp. 456–457. doi:10.1109/RE.2017.44.

[26] H. F. Soares, N. S. Alves, T. S. Mendes, M. Mendonca, R. O. Spinola, Investigating the Link between User Stories and Documentation Debt on Software Projects, in: 2015 12th International Conference on Information Technology - New Generations, 2015, pp. 385–390. doi:10.1109/ITNG.2015.68.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7113503

[27] S. E. Sim, R. E. Gallardo-Valencia, Performative and lexical knowledge sharing in agile requirements, in: W. Maalej, A. K. Thurimella (Eds.), Managing Requirements Knowledge, Springer, 2013, pp. 199–219. doi:10.1007/978-3-642-34419-0_9.
URL https://doi.org/10.1007/978-3-642-34419-0_9

[28] R. Barbosa, A. E. 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, 2016, pp. 2–5. doi:10.1109/DSN-W.2016.27.

[29] L. M. Cysneiros, J. C. S. do Prado Leite, Using the Language Extended Lexicon to Support Non-Functional Requirements Elicitation., in: WER, Citeseer, 2001, pp. 139–153.

[30] J. Patton, P. Economy, User Story Mapping: Discover the Whole Story, Build the Right Product, Peter Economy, 2014.

[31] M. Chinosi, A. Trombetta, Bpmn: An introduction to the standard, Computer Standards Interfaces 34 (2012) 124–134. doi:10.1016/j.csi.2011.06.002.

[32] Mike Cohn, User Story Template Advantages (2008).
URL https://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template

[33] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in software engineering: an introduction, Vol. 15, Kluwer Academic Publishers Norwell, Netherlands, 2000.

[34] N. Juristo, A. M. Moreno, Basics of Software Engineering Experimentation, 1st Edition, Springer Publishing Company, Incorporated, 2010.

[35] S. M. Ross, Introduction to probability and statistics for engineers and scientists, 2004.

[36] S. S. Shapiro, M. B. Wilk, An analysis of variance test for normality (complete samples), Biometrika 52 (3/4) (1965) 591–611.
URL http://links.jstor.org/sici?sici=0006-3444%28196512%2952%3A3%2F4%3C591%3AAAOVTF%3E2.0.CO%3B2-B

[37] S. M. Ross, Introduction to probability and statistics for engineers and scientists, Academic Press, Cambridge,MA, USA, 2004.

[38] G. Macbeth, E. Razumiejczyk, R. Ledesma, Cliff's delta calculator: A non-parametric effect size program for two groups of observations, Universitas Psychologica 10 (2011) 545–555. doi:10.11144/Javeriana.upsy10-2.cdcp.

[39] F. J. Shull, J. C. Carver, S. Vegas, N. Juristo, The Role of Replications in Empirical Software Engineering, Empirical Softw. Engg. 13 (2) (2008) 211–218. doi:10.1007/s10664-008-9060-1.

[40] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, D. McClosky, The {Stanford} {CoreNLP} Natural Language Processing Toolkit, in: Association for Computational Linguistics (ACL) System Demonstrations, 2014, pp. 55–60.
URL `http://www.aclweb.org/anthology/P/P14/P14-5010`