Limiting the Scope of the Domain Language to Describe the Application Language

Leandro Antonelli[1], Gustavo Rossi[1], Julio Cesar Leite[2], and Alejandro Oliveros[3]

[1]Lifia, Fac. de Informática, UNLP, Bs As, Argentina
{lanto,gustavo}@lifia.info.unlp.edu.ar
[2]Dep. Informática, PUC-Rio, Gávea, RJ, Brasil
julio@inf.puc-rio.br
[3]INTEC, UADE, Bs As, Argentina
oliveros@gmail.com

Abstract. Agile methodologies arise as a way to cope with problems of estimation and planning that occur in traditional software development. Nevertheless, the transition to agile could be hard. The key challenge reported by literature refers to requirements, in particular with the identification and description of User Stories. It is stated that User Stories are often described vaguely by the wrong people at the wrong time. To cope with this problem, there is an already proposed strategy which consists in deriving User Stories from the domain language captured through the Language Extended Lexicon (LEL). This strategy produces an initial set of User Stories with small effort. Since the LEL aims to capture the language of the domain while the User Stories must be limited to a specific application we believe that it is necessary to adjust the LEL. In this paper, we propose a strategy to limit an LEL that describes the language of a domain to describe the vocabulary of a specific application. Consequently, User Stories derived from the limited LEL using the already proposed approach will only describe the functionality of the application. Based on a preliminary evaluation, we state that teams with no experience in writing User Stories obtain better products. Although the results are preliminaries, we think that they are promising.

Keywords: Agile Development · Language Extended Lexicon · User Stories.

1 Introduction

The development of software systems is a complex activity. Many actors with different concerns are involved, and they build several software products which are interrelated. Besides, the nature of software also makes its development a complex task. Brooks [7] identifies four characteristics that make software a different kind of artifact altogether, contributing to the complexity of its construction: invisibility, modifiability, conformity, and complexity. These characteristics are some of the reasons why there are so many budget overruns in software projects, as well as failure to deliver what was expected [29].

Agile methodologies (like Scrum) rely on incremental and iterative development to cope with these problems. Moreover, agile methodologies also rely on a close communication between clients and development teams to provide a valuable and immediate feedback during software development. Nevertheless, it is difficult to be agile. Development teams suffer the transition to agility and enjoy at first the freedom that agile development provides, and later, they find out that challenges arise, some of them related to requirements.

User Stories play a critical role in this transition. They are often written by the wrong people at the wrong time [13]. Therefore, their descriptions take significant effort and are finally imprecise [28]. Furthermore, after the functionality described by the User Stories is coded, there is little information above the trace between the User Stories and the related code that implement the functionality [6].

The Language Extended Lexicon (LEL) is a technique for specifying domain (context) language [20]. LEL is a very convenient tool for experts with no technical skills, although people with such skills will profit more from its use [26]. LEL effectively captures and models the domain language because it conforms to the mechanism used by the human brain to organize expert knowledge [32]. In particular, the convenience of LEL as a tool arises from three significant characteristics: it is easy to learn [10], easy to use [12] and it has good expressiveness [17].

Antonelli et al. [1] propose a strategy to derive User Stories from LEL. The User Stories obtained with the strategy are preliminaries and must be analyzed and enriched. Nevertheless, they are obtained automatically from the LEL. Thus, the strategy is a good contribution to teams that are initiating in agile because the User Stories has proven to be useful. Nevertheless, it is needed an improvement because the LEL captures the language of a domain, while User Stories must describe the functionality of an application. Consequently, the scope of the application to describe with User Stories must be defined.

In this paper, we propose a strategy to limit an LEL that describe a domain language to capture only an application language. Then, User Stories derived from the LEL with the already proposed strategy will only describe the functionality of the application. It is important to emphasize that the strategy proposed does not change in any way the derivation process proposed by Antonelli et al. [1]. The strategy proposed consists in adjusting the LEL that the derivation process will use as input. The rest of the paper is organized as follows: Section 2 presents some background necessary to understand the strategy. Section 3 describes the strategy. Section 4 shows a preliminary evaluation of such strategy. Section 5 discusses some related works. Finally, Section 6 presents some conclusions and future works.

2 Background

This section describes some basic concepts about User Stories, the Language Extended Lexicon, and the derivation of User Stories from LEL.

2.1 User Stories

A User Story is a description in natural language that captures what the user wants

to achieve. User Stories generally adjust to a template which considers three attributes: a *role*, a *goal/desire*, and *reason* [9]. The *role* defines the user who interacts with the application. The *goal/desire* represents the requirement that the application must fulfil for the role. Both these attributes refer to elements within the scope of the application. In contrast, the *reason* belongs to the context of the application and states why the user requires that the application provides the functionality described in *goal/desire* (Figure 1).

As a <role>,
I want <goal/desire>,
so that <reason>.

Fig. 1. User Story template.

To illustrate the approach, it is used a real example involving the management of vacation periods for the employees of a company. The employees have a certain number of vacation days per year. This number depends on how many years the employee has been working for the company. When the employee requests a vacation period, the request needs two authorizations to become effective. First, the Human Resources Department must verify that the employee is not requesting more days than those he is allowed to take. After that, the employee's manager must analyse the project schedule to determine whether or not the employee's presence is essential at work in the vacation period he has requested. A schedule planner is in charge of building the project schedule. He defines the tasks and assigns employees to those tasks in a period of time. Figure 2 shows an example of a User Story.

As an employee,
I want to request vacation period,
So that I take some days off work.

Fig. 2. User Story example.

2.2 Language Extended Lexicon (LEL)

LEL is a glossary whose goal is to register the definition of terms that belong to a domain. It is tied to a simple idea: "understand the language of a problem without worrying about the problem" [20]. Terms (called symbols within LEL) are defined through two attributes: notion and behavioral responses. Notion describes the denotation, i.e. the intrinsic and substantial characteristics of the symbol, while behavioral responses describe symbol connotation, i.e. the relationship between the term being described and other terms.

Each symbol of the LEL belongs to one of four categories: subject, object, verb or state. This categorization guides and assists the requirements engineer during the description of the attributes. Table 1 shows each category with its characteristics and how to describe them.

Some symbols identified for the domain of the example are summarized in Table 2. They are organized according to their category. Some examples of LEL symbols description are: *employee* (Figure 4), *request vacation period* (Figure 5), *build sched-*

ule (Figure 6) and *analyze request* (Figure 7). Underlined words identify other symbols which are defined in the LEL of the example.

Table 1. LEL categories.

Category	Characteristics	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 found	Situation represented	Actions that must be performed to change into another state

Table 2. LEL symbols of vacation application domain.

Category	Symbols
Subject	Employee, Human Resources Department, Manager, Schedule planner
Object	Vacation Request, Project schedule, Vacation Period, Task, Resoure, Software
Verb	Define vacation period, Request vacation period, Verify vacation request, Analyze vacation request, Check vacation request, Build project schedule, Authorize vacation request, Define tasks, Define resources, Assign people, Organize tasks, resources and people, Design software, Code software, Test software
State	Vacation request created, Vacation request analysed, Vacation request approved, Vacation request accepted, Software analyzed, Software designed, Software coded, Software tested

Subject: Employee
Notion
Person who works for the company and has some skills in software development.
Behavioral responses
Employee defines vacation period
Employee requests vacation period
Employee designs software
Employee codes software
Employee tests software

Fig. 3. *Employee* symbol description.

Verb: Request vacation period
Notion
Act of asking for permission to take some days off work
Behavioral responses
The employee defines the vacation period
Human Resources Department verifies vacation request
Manager analyzes vacation request
Human Resources Department authorizes vacation request

Fig. 4. *Request vacation period* symbol description.

2.3 Obtention of User Stories from an LEL

The LEL has useful knowledge that makes it possible to obtain User Stories from it [1]. User Stories are described through three attributes: *a role* ("As a <role>"), a *goal / desire* ("I want <goal/desire>") and *reason* ("so that <reason>"). The first attribute to be considered in the process of obtain User Stories from the LEL is the *goal / desire*. It is related to a verb since verbs are actions performed in the scope of an application. Then, a *role* is necessary to perform the action. Subjects are naturally related to verbs because the behavioral responses of subjects describe the actions that the subjects perform. Thus, the *role* is the subject who performs the action stated by the verb. Finally, the *reason* for the *goal / desire* can be found in the notion of the verb symbol. Figure 7 shows ATL transformation [4] to obtain User Stories from LEL.

Verb: Build project schedule
Notion
Act of organizing people, resources and tasks in a timeline
Behavioral responses
Schedule planner defines tasks
Schedule planner defines resources
Schedule planner assigns people
Schedule planner organizes tasks, resources and people in a timeline

Fig. 5. *Build schedule* symbol description.

Verb: analyze vacation request
Notion
Act of analyzing the project schedule needs for the period requested
Behavioral responses
Manager checks vacation request with project schedule
Manager authorizes vacation request

Fig. 6. *Analyze vacation request* symbol description.

According to the example, if it is considered the symbols *employee* (Figure 3) and *request vacation period* (Figure 4), the User Story obtained through the transformation is described in Figure 8.

```
rule LEL2UserStory  {
from s : Symbol (s.isVerb())
to u : UserStory (
  u.role <- s.referencedInBehaviouralResponsesFrom() ->
select (x| x.isSubject()) -> first()
  u.goal/desire <- s.name
  u.reason <- s.notion }
```

Fig. 7. ATL transformation for derivation of User Stories from LEL.

As an employee,
I want to request vacation period
So that I take some days off work

Fig. 8. User Story derived from *employee* and *request vacation period* symbols.

Oliveira et al. [27] refer the definition of a goal as a condition that an actor would like to achieve. They find useful the usage of states to identify goals. And the states are modified and influences by actions. These goals make it possible to obtain functional requirements as well as non-functional requirements. In this paper, we are going to deal only with functional requirements.

3 Our Approach

Our proposed approach has the goal of defining the scope of the application in an LEL that captures the language of a domain. The definition of the scope is performed reducing an LEL that captures the language of the domain to a smaller LEL that captures only the language exclusively used in the application. Therefore, this LEL that only describe the application can be used with the derivation strategy described in the Background section to derive User Stories.

Our approach is a process that requires an LEL as input and produces a new LEL as output. The input LEL must capture the language of the domain for which an application will be developed. The output LEL will describe the vocabulary used within the scope of the application (and not the whole domain). Both LELs, the domain and the application LELs, has the same structure [20]. The difference is that the domain LEL pursues the traditional aim of capturing the language of the domain and must be constructed in a regular way. While the application LEL is a subset of the domain LEL and only includes elements that are within the boundaries of the application.

Our approach consists of two steps. The first step refers to identify the scope of the application, while the second step refers to reduce the LEL to include only elements that will be located within the boundaries of the application. To identify the scope of the application, the behavioral responses of the symbols must be analyzed and must be selected the behavioral responses that represent functionality that must be included in the application. Then, to reduce the LEL to elements included in the boundaries of the application, all the symbols must be analyzed and the ones which are used in the behavioral responses must be selected (Figure 9).

The following sections describe each step.

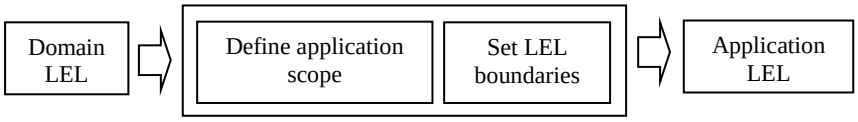

Fig. 9. Our approach in a nutshell.

3.1 Define application scope

IEEE standard 830-1998 states that requirements must describe clearly what the software system must do [15]. They recommend using the expression “the system shall…” because it states clearly the functionality and the obligatory condition that the software system must implement. Kovitz [18] propose the template “<role> must <action>” to define requirements. This template refines the IEEE template because it defines clearly the *role* that will perform the *action*. Behavioral responses of the LEL

symbols are recommended to be described using the template "<subject> + <verb> + <object>" [2]. This template is similar to template proposed by Kovitz, because the *subject* is related to the *role*, and the *verb* is related to the *action*. Then, the description of the behavioral responses is more precise than Kovitz template, because LEL description also considers the *object* that receives the *action*.

Since behavioral responses are similar to requirements and more precise, to define the scope of the application, the behavioral responses of the symbols must be analyzed, and it must be selected the behavioral responses that refer to functionality that will be included in the application.

Let's consider that the scope of the system includes only functionality to manage the vacation request and authorization workflow. And the project schedule and its management are outside of the boundaries. The symbol *request vacation period* (Figure 4) include behavioral responses that are inside the boundaries of the application. The symbol *build schedule* (Figure 5) does not include any behavioral response inside the boundaries of the application, because all of them are related to the construction of the project schedule. Then, the symbol *analyze vacation request* (Figure 6) has one behavioral response inside the boundaries of the application ("*Manager authorizes vacation request")*, while the other behavioral response ("*Manager checks vacation request with project schedule*") is outside the boundaries of the application.

3.2 Set LEL boundaries

Good practices in constructing the LEL recommend to describe behavioral responses using the template "<subject> + <verb> + <object>" [2]. In particular, behavioral responses of *subject* symbols must describe the action that the subject performs and behavioral responses of *object* symbols must describe the action that the object receives. That is, the behavioral responses of some symbol S must begin with S (S + <verb> + <object>). And the behavioral responses of some symbol O must end with O (<subject> + <verb> + O). In consequence, there is a reflexive reference in behavioral responses of subjects and objects. Behavioral responses of verbs symbols are different because they describe a set of steps to perform the verb which is described. Hence, to describe a verb V, the behavioral responses do not have to use the same verb V to avoid a cyclic definition.

The subject *employee* (Figure 3) has some behavioral responses related to the workflow for requesting vacation and others related to developing software. All of them begin with the symbol *employee*. The verb *request vacation period* (Figure 4) is performed by three steps that are enumerated in the behavioral responses, and none of them use the verb *request vacation period*.

The first step of our proposed approach (described in section 3.1) defines the scope of the application by selecting the behavioral responses that refer to actions that occur inside the boundaries of the application. Thus, if the action V_1 is inside the boundaries of the application, some behavioral response BR_1 with a description $S_1+V_1+O_1$ must be selected. That means that a subject S_1 performs the action and the object O_1 receives the action. Then, both S_1 and O_1 will include BR_1 within their descriptions of behavioral responses. Moreover, the LEL must include one verb V_2, which also include BR_1 within their descriptions. The four symbols: V_1, S_1, O_1, and V_2 should be considered as symbols of the application LEL. The symbol V_1 is explic-

itly identified as an action inside de boundaries of the application. The symbols S_1 and O_1 give more information about the action. And one of the steps to perform the symbol V_2 is the action V_1that it is inside the boundaries of the application. Therefore, V_2 is also inside the boundaries (At least, partially).

There are two rules to set the LEL boundaries. The first rule determines to consider the symbols that are explicitly mentioned in every behavioral response that is selected. The second rule determines to consider the symbols which include a behavioral response that was chosen be to be inside the application boundaries. For *subject* and *object* symbols both rules are redundant because if a *subject* and *object* is referenced in a behavioral response, the *subject* or *object* must include the behavioral response in their description. In *verbs* symbols, both rules are complementary, because *verbs* descriptions of behavioral response do not include the verb which is defined in their behavioral responses.

For example, let's consider that the behavioral response "*The employee requests vacation period*" is chosen to be inside de application scope in step 1. This expression is included in subject *employee* (Figure 3). Although the symbol *vacation period* is not shown in this paper, the behavioral of this symbol also includes the expression. Thus, according to the first rule, both symbols *employee* and *vacation period* must be included in the application LEL. Then, the verb *request vacation period* must also be included according to the same first rule. In this example, the verb *request vacation period* is a high-level action, and no action contains it. But there could exist a symbol *go on holiday* that would include "*the employee requests vacation period*" and it that case, the symbol *go on holiday* should be included to build the application LEL.

Let's consider another behavioral response "*Schedule planner builds schedule.*" This behavioral response was not chosen to be in the application scope. That means that the subject *schedule planner* must not be chosen, neither *schedule* (considering that no other behavioral response justify the selection of these symbols). The verb *build schedule* neither has to be considered nor another verb that could have included this behavioral response.

Let's consider a mix situation. For example, the behavioral response "*manager authorizes vacation request*" makes the subject *manager*, the object *vacation request,* and the verbs *authorize* and *analyze* should be included in the application LEL. At this point, it is important to mention that the description of the behavioral responses must be adjusted according to the behavioral responses which were chosen in the first step of the approach. In the case of the symbol *analyze vacation request* must include only the behavioral response "*manager authorizes vacation request,*" because the behavioral response "*manager checks request with project schedule*" was not chosen.

4 Preliminary Evaluation

A preliminary quasi-experiment was performed to assess the strategy proposed. The aim of limiting the scope of the LEL is to improve the User Stories obtained from it through the derivation process proposed by Antonelli et al. [1]. The experiment consisted in comparing the User Stories constructed with two different techniques. One technique consisted in limiting the scope of a domain LEL to obtain an application LEL and obtain User Stories from the application LEL. The other technique con-

sisted in writing the User Stories in a traditional way, that is, from the knowledge that product owner elicited from stakeholders without using any kind of LEL.

The analysis was performed in two ways. We consider the production process (because it is related to the definition of the scope of the language and the derivation of User Stories). And we also consider the consumption of the User Stories, because it is a way assessing the quality of the User Stories produced. Thus, the goal of the quasi-experiment is described according to the Goal/Question/Metric (GQM) method formulated by Basili et al. [5]:

> Analyze *the traditional approach and our approach for producing User Stories*
> for the purpose of *comparing the writing and reading processes*
> with respect to *time and easiness*

To do that, we measured four variables: the time needed to write User Stories, easiness to write, the time required to understand the User Stories and understandability. The variables can be grouped in: one pair of variables to assess the writing process and the other to evaluate the reading process. Each pair of variables has an objective measure in minutes (time) and a subjective measure, the perception of the subject about how easy it is to write and how understandable the User Stories are. These subjective variables were measured in a range of 5 values: very easy, easy, medium, hard and very hard. Participants were asked to complete a questionnaire ranking each User Story after writing or reading it.

We decided to design a quasi-experiment with two different teams that had never used agile methodologies before. In particular, we took benefit from the transformation of the development process to agile (Scrum) as an opportunity to run this experiment. The teams were working in different modules of the same software system. And they have never used User Stories. The system was of business intelligence application. It analyzes the performance of different lawyers in the prosecution of citizens who have debts of taxes. After the analysis, the application recommends rewarding or penalizing the lawyers regarding their performance.

Each team had a product owner (who had used LEL technique before) responsible for writing User Stories and for providing the information about the time and easiness to write User Stories. Then, during the planning session, the whole team had to read and analyze the User Stories, so they provided information about time and level of understandability. The experiment lasted four sprints. Both teams used the two approaches alternately. One team used our proposed approach first while the other started with the traditional approach of writing User Stories.

The participants of the experiment were eight practitioners divided into two teams of four people. The development team was composed of members with a degree in Computer Science and experience in the software development industry for more than ten years. Product owners were requirements engineers who also had a degree in Computer Science and experience in the software development industry for more than 15 years.

The participants (developers and product owners) received training in User Stories and Scrum in general as part of the migration of the development process in which they were involved. The training consisted of weekly workshops during two months where different topics of agile development were taught and practiced: (i) introduction to agile, (ii) Scrum roles, (iii) Scrum ceremonies, (iv) management of the product backlog, (v) estimation of User Stories, (vi) prioritization of User Stories and (vii)

sprint execution and overcoming impediments. Additionally, we provided training in the application of our proposed approach, that is, how to limit the domain LEL to obtain an application LEL. And we also provided training in the approach to derive User Stories from LEL.

The experiment was carried out during four sprints of fifteen days each, where both product owners received information about the functionality they had to specify. There were different types of User Stories. Some User Stories define strategies to analyze the performance of the lawyer with different criteria. Other User Stories recommend actions to reward or penalize lawyers considering their performance. Also, there were a third group of User Stories to configure the process of analysing and recommend, and to show the results of such processes. Then, they wrote the User Stories and provided us with the information about the time they had needed for specifying and their perception about its easiness. Products owner alternatively used the two different techniques to write User Stories. They wrote User Stories directly from the knowledge they had received. And they used an already prepared LEL that captured the domain language. They applied our proposed approach to limit the domain LEL to an application LEL. And after that, they applied the technique to derive User Stories from the application LEL.

In the planning session, the product owners presented the User Stories to their teams. The teams analyzed them and provided us with the information about the time they had needed to understand them and about how understandable they were. We collected information about 48 User Stories: 24 were specified using the traditional approach, and 24 were specified using our approach. Every team worked with a different set of functionality, so, they had to write and read different User Stories.

Although we were not able to perform a quantitative analysis due to only two people were writing User Stories, the case analysis provided us with some preliminary results about the advantages of the proposed approach. The experiment showed that our approach has benefit in the writing process. The time required to apply our approach was lower that the time needed to apply the conventional approach (averages time were 12 and 18 minutes). Moreover, the complexity perceived in applying our approach was very small in comparison with the conventional approach. There is a significant intellectual effort in understanding a new domain, defining the scope of the application and writing User Stories. Thus, our approach relies on an already prepared LEL that describes the domain. The intellectual effort of applying our approach to reducing the domain LEL to an application LEL is low. There is no creativity process involved since our approach relies on reviewing symbols and deciding whether they belong or not to the application limits. Then, obtaining User Stores through the derivation strategy proposed by Antonelli et al. [1] is also straightforward. The results revealed that there were no differences in the reading processes. Since time and understandability were similar in both approaches, we can argue that the quality of the User Stories produced with both approaches was similar. Considering that our approach has advantages about the writing process, and the quality of the products are similar in our approach and the conventional approach, we can claim that our approach has benefits over the conventional approach.

Feldt et al.[11] state the importance of analyzing threats to the validity of the study and the results. Wohlin et al. [31] group validity threats into four categories: conclu-

sion, internal, construct and external validity. The following paragraphs analyze different threats from each category.

Concerning the conclusion category, one possible threat is random heterogeneity of subjects. The participants were heterogeneous as regards years of experience in the industry, but there is homogeneity regarding overall experience since no one had had experience in the application of both techniques.

The second category of threats to analyze is internal validity. Selection is the main threat to internal validity. To tackle the effect of natural variation in human performance, we selected people with no experience in both strategies to be applied. Then, we carried out a paired design where all the subjects had to apply both treatments.

According to the construct validity category, we observed that the experiment did not suffer from such threats referred to as hypothesis guessing, evaluation apprehension or experimenter expectancies. The subjects were not familiar with the approaches, nor they knew the approach to be tested, so they could not force specific results.

5 Related Work

Maiden [24] states that writing requirements is a complex cognitive task that needs a diverse range of interleaved cognitive processes. He states that we need more empirical studies of requirements work. Then, Savolainen et al. [28] report a transition to agile development and they state that they have problems using User Stories because they fail to obtain User Stories with the necessary expressiveness. Our work was inspired by both works. We faced the same problem as Savolainen et al. And the strategy proposed has the aim of reducing the cognitive effort, because the huge effort is performed while creating the LEL. After that, our proposed approach is quite straightforward to produce the application LEL and User Stories.

Maiden et al. [23] state that writing requirements, in particular, agile requirements, demands a level of creativity that is not easy to reach by requirements engineering. In this context, integrating software product line engineering can be beneficial, as Mohan et al. reported in their work [25]. Our work is based on both ideas. Conforming to the philosophy of the product line engineering, our approach is focused on the use of two types of LEL: a domain LEL and an application LEL. The domain LEL is similar to the base of knowledge that software product lines use to build a family of products. Lan Cao et al. [8] perform an empirical study of the agile requirements engineering practices regarding their importance and the cost/benefit of their application. They discover that iterative requirements engineering is one the most important and valuable practice; nevertheless, it is costly to implement. Issa et al. [16] developed a catalog to reduce the effort to specify requirements. They showed that the catalog they proposed satisfies the attribute of completeness required by requirements. Our strategy can be considered an iterative requirements engineering practice since the same domain LEL is used sprint by sprint to produce different application LEL. Moreover, we can also consider that the domain LEL is a kind of catalog.

Leite et al. [21] propose a scenario construction process, where the scenarios are obtained from the LEL. The LEL describes the vocabulary while the scenarios describe the application. One important thing is that LEL evolves as the scenario building process progress. Our strategy is quite similar to this strategy. Both strategies

begin with the LEL, although we make a distinction between two types of LEL. After that, the User Stories produced in our approach from the application LEL are simpler that the Scenarios of Leite. Finally, it is important the consideration of the evolution of the LEL. We think that this is an important issue to consider in our approach. Breitman et al. [6] propose a framework for managing User Stories, in particular, they propose a way of organizing User Stories and keeping track to Scenarios, which are also linked to User Stories. Although Breitman proposal aims if different from ours, both approaches rely on the same elements. Both approaches have an LEL that provides the vocabulary, and the User Stories are linked to the LEL.

Ianzen et al. [14] propose an approach to defining the scope in software product lines. They propose a tool to analyze the documents and extract features that specialist must analyze instead of analyzing the whole documents. Our approach is similar because our domain LEL can be considered the summary their approach obtain from the documents. Lucassen et al. [22] propose an approach to produce high-quality User Stories. Their way of reaching high quality is in part linking User Stories to a language with a similar structure the LEL has. Waldmann et al. [30] propose an approach to developing agile requirements in any types of software development. Although our experiment compared User Stories, the derivation strategies proposed by Antonelli et al. in [1] also considers Use Cases. So, our proposed approach of limiting the domain LEL to an application LEL could be used in combination with Use Cases.

6 Conclusions and Future Works

Agile methodologies seem to be the solution to problems of time and cost overrun in software development. Nevertheless, it is difficult for organizations to move to an agile development process. The key challenge for them is mastering the process of identifying and describing User Stories. We have presented an approach to limit the domain language captured with LEL to produce a new LEL that only describe the application language. Consequently, User Stories can be derived from this application LEL.

We believe that the proposed approach is useful from several points of view. Teams that regularly use the LEL will be benefited with the proposed approach. The preliminary experiment showed the advantage of our approach in the writing process of User Stories, which are of similar quality to the ones created in the regular process and are obtaining with low effort.

People that do not use regularly LEL has the extra effort of building the LEL. Nevertheless, the LEL captures the language of the domain, and the process of capturing the language is implicitly performed although the LEL is not produced. Thus, the extra effort relies on writing the LEL. It was experienced the advantages of constructing the LEL in the regular way [10] [12] [17]. Moreover, we worked in a process to build the LEL in a collaborative way [3], and we are currently working on a tool that makes it possible this construction. We are going to perform further experimentation to compare the whole process of writing User Stories including the LEL domain construction.

We can relate the benefits of our approach with the use of domain specific languages against the general-purpose modeling languages. Kosar et. al. [19] did an ex-

periment that compares DSLs with GPLs, and they have proven the benefits of the DSLs in all the cognitive dimensions.Regarding the transformation from domain LEL to application LEL we are working on the possibility that new elements will appear in the application LEL that are not present in the domain LEL. Sometimes, software application adapts or transforms the context where the application is introduced, and sometimes application includes elements that only belong to the domain. That is why we are working in extending the approach.

References

1. Antonelli, L., Rossi, G., Leite, J.C.S.P., Oliveros, A.: Deriving requirements specifications from the application domain language captured by Language Extended Lexicon, Workshop in Requirements Engineering (WER), Buenos Aires, Argentina (2012)
2. Antonelli, L., Rossi, G., Leite, J.C.S.P., Oliveros, A.: Buenas prácticas en la especificación del dominio de una aplicación, Workshop in Requirements Engineering (WER), Montevideo, Uruguay, April 8 – 10 (2013)
3. Antonelli, L., Rossi, G., Oliveros, A.: A Collaborative Approach to Describe the Domain Language through the Language Extended Lexicon, Journal of Object Technology 15(3):3:1, DOI: 10.5381/jot.2016.15.3.a3, June (2016).
4. ATL a model transformation technology, http://eclipse.org/atl/
5. Basili, V.R., Caldiera, G., Rombach, H.D.: The goal question metric approach, in Encyclopedia of Software Engineering, John Wiley & Sons, Vol. 1, pp.528-532 (1994)
6. Breitman, K.K., Leite, J.C.S.P.: Managing User Stories, in Proceedings of the International Workshop on Time Constrained Requirements Engineering, Essen, Germany (2002)
7. Brooks, F., The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley Professional, 2 edition (1995)
8. Cao, L., Ramesh, B.: Agile Requirements Engineering Practices: An Empirical Study, IEEE software volume 25 issue 1, DOI 10.1109/MS.2008.1, Jan-Feb, pp. 400-403 (2008)
9. Cohn, M.: User Stories Applied, Addison Wesley, ISBN 0-321-20568-5 (2004)
10. Cysneiros, L.M., Leite, J.C.S.P.: Using the Language Extended Lexicon to Support Non-Functional Requirements Elicitation, in proceedings of the Workshops de Engenharia de Requisitos, Wer'01, Buenos Aires, Argentina (2001)
11. Feldt, R., Magazinius, A.: Validity threats in empirical software engineering research - An initial survey, in proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE), July 1-3, San Francisco Bay, pp 374-379 (2010)
12. Gil, G.D., Figueroa, D.A., Oliveros, A.: Producción del LEL en un Dominio Técnico. Informe de un caso, in proceedings of the Workshops de Engenharia de Requisitos, Wer'00, Rio de Janeiro, Brazil (2000)
13. Hudson, W.: User Stories Don't Help Users: Introducing Persona Stories, Interactions of ACM, DOI: 10.1145/2517668, Nov/Dec (2013)
14. Ianzen, A., Malucelli, A., Reinehr, S.: Scope definition in software product lines: A semi-automatic approach through linguistic annotation, XXXVIII Conferencia Latinoamericana En Informatica (CLEI), Medellin, doi: 10.1109/CLEI.2012.6427193 , pp. 1-9 (2012).
15. IEEE, IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998 (Revision of IEEE Std 830-1993)
16. Issa, A.A., Ali, A.I.: Automated requirements engineering: Use case patterns-driven approach, IET software vol 5 issue 3, DOI 10.1049/iet-sen.2010.0014, pp 287-303 (2011)

17. Kaplan, G., Hadad, G., Doorn, J., Leite, J.C.S.P.: Inspección del LexicoExtendido del Lenguaje, Workshops de Engenharia de Requisitos, Wer'00, Rio de Janeiro, Brazil (2000)
18. Kovitz, B.: Practical software requirements: A manual of content and style, ISBN978-1884777592, Ed. Manning Publications (1998)
19. Kosar, T., Oliveira, N., Mernik, M., Pereira, M. J. V., Črepinšek, M., da Cruz, D., Henriques, P. R.: Comparing General-Purpose and Domain-Specific Languages: An Empirical Study, Computer Science and Information Systems, Vol. 7, No. 2. pp. 247-264 (2010)
20. Leite, J.C.S.P., Franco, A.P.M.: A Strategy for Conceptual Model Acquisition, in Proceedings of the First IEEE International Symposium on Requirements Engineering, San Diego, California, IEEE Computer Society Press, pp 243-246 (1993)
21. Leite, J.C.S.P., Hadad, G., Doorn, J.H., Kaplan, G. N.: A Scenario Construction Process, Requirements Engineering, Volume 5, Issue 1, pp 38–61 (2000)
22. Lucassen, G., Dalpiaz, F., Brinkkemper, S., van der Werf, J. M. E. M.: Forging High-Quality User Stories: Towards a Discipline for Agile Requirements. In Proceedings of the IEEE International Requirements Engineering Conference (2015)
23. Maiden, N., Jones, S.: Agile Requirements Can We Have Our Cake and Eat It Too?, IEEE software volume 27 issue 3, DOI 10.1109/MS.2010.67, May-Jun, pp 87-88 (2010)
24. Maiden, N.: Exactly How Are Requirements Written?, IEEE software volume 29 issue 1, DOI10.1109/MS.2012.6, Jan-Feb, pp26-27 (2012)
25. Mohan, K., Ramesh, B., Sugumaran, V.: Integrating Software Product Line Engineering and Agile Development, IEEE software volume 27 issue 3, DOI 10.1109/MS.2010.31, Feb, pp48-55 (2010)
26. Oliveira, A.d.P.A., Leite, J.C.S.P.: Cysneiros, L.M., Cappelli, C.: Eliciting Multi-Agent Systems Intentionality: from Language Extended Lexicon to i* Models, in proceedings of XXVI International Conference of the Chilean Society of Computer Science (SCCC '07), 8-9 Nov, pp 40 – 49 (2007)
27. Oliveira, A.d.P.A., Leite, J.C.S.P.: Building Intentional Models Using the ERi*c Method, Cadernos do IME: Série Informática, Vol. 31, Dezembro 2011, pp 46 / 53 (2011)
28. Savolainen, J., Kuusela, J., Vilavaara, A.: Transition to Agile Development - Rediscovery of Important Requirements Engineering Practices, in Proceeding of the 18nd IEEE International Requirements Engineering Conference (RE), IEEE, ISBN 978-1-4244-8022-7, DOI 10.1109/RE.2010.41, sept 27-oct 1, Sydney, pp 289 – 294 (2010)
29. The risk digest, forum on risks to the public in computers and related systems, ACM committee on computers and public policy, available at http://catless.ncl.ac.uk/Risks/, accessed on September (2015)
30. Waldmann, B., Phonak, A.G.: There's never enough time: Doing requirements under resource constraints, and what requirements engineering can learn from agile development, 19th IEEE International Requirements Engineering Conference (RE), IEEE, ISBN 978-1-4577-0921-0, DOI 10.1109/RE.2011.6051626, Trento, pp 301 – 305 (2011)
31. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering an introduction, ISBN 0-7923-8682-5 Academic Publishers (2000)
32. Wood, L.E.: Semi-structured interviewing for user-centered design, Interactions of the ACM, april-may, pp48-61 (1997)