2014 9th International Conference on the Quality of Information and Communications Technology

A study on the viability of formalizing Use Cases

Rui Couto, António Nestor Ribeiro, José Creissac Campos
Dept. Informatics, University of Minho & HASLAB / INESC TEC
Braga, Portugal
{ruicouto, anr, jose.campos}@di.uminho.pt

***Abstract*—Use case scenarios are known as powerful means for requirements specification. On the one hand, they join in the same modeling space the expectations of the stakeholders and the needs of the developers involved in the process. On the other hand, they describe the desired high level functionalities. By formalizing these descriptions we are able to extract relevant informations from them. Specifically, we are interested in identifying requirements patterns (common requirements with typical implementation solutions) in support for a requirements based software development approach. This paper addresses the transformation of use case descriptions expressed in a Controller Natural Language into an ontology expressed in the Web Ontology Language (OWL), as well as the query process for such information. It reports on a study aimed at validating our approach and our tool with real users. A preliminary set of results is discussed.**

I. Introduction

In previous work [1] we have targeted our efforts at the correctness of software systems with respect to stakeholders' expectations. In traditional approaches [2], requirements are specified at the beginning of the development process, and used as guides for development. However, they are only validated at the end of the process. Such leads to a gap between user requirements and the software development process, which might result in the misunderstanding of the stakeholders' concerns. We have been working in reducing this gap, by taking use cases as the requirements specification means and formalizing them, which allow us to further process them.

Notably use cases are a popular method for requirements specification. They were proposed by Jacobson [3] and later adopted by the Object Management Group. A use case model is composed of two parts: a graphical representation that summarizes the user interactions with the software system being described; and the specifications of each individual use case. Use cases mainly represent user functionalities and, as such, provide knowledge to derive high level information about the software systems they describe.

Despite the inexistence of an accepted standard for use cases specification, textual formats are usual approaches (e.g., as proposed by Fowler [4] or Cockburn [5]). We proposed a new language to specify user requirements that combines rigor with a style of writing closer to natural language. It is concretized as a Controlled Natural Language (CNL), with a twofold objective: first to integrate the stakeholders in the specification process; second to transform use case specifications into other languages.

It is possible to find several works regarding the use of ontologies in support for requirements engineering. Castañeda *et al.* present a review on the uses of ontologies in requirements engineering [6]. In this context the use of ontologies tends to address requirements specification and high level analysis (such as elicitation, verification, etc.). Generally, the proposed approaches do not use ontologies to further operationalize the requirements. We propose the Web Ontology Language (OWL) [7] both to represent use cases' knowledge, and to support reasoning about it (see [1] for the rational). The expressiveness of OWL allows us to define requirements' ontologies as well as their instances, and to perform queries over such knowledge. The pattern inference capabilities are provided by the Semantic Web Rule Language (SWRL) and Simple Protocol and RDF Query Language (SPARQL). Kirasić and Basch's work presents an example of pattern inference resorting to SWRL [8].

We have developed an approach to enable the pattern inference from use case specifications by means of an automated process. Due to space restrictions we present only a brief description of the process. Further details, an example and a description of the developed tool can be found in [1]. Our process starts with the specification of the use case descriptions. To do such, we have create a Domain Specific Language which takes the form of a CNL [9], the Restricted Use-Case Statements (RUS) (and the corresponding meta-language, the Restricted Use-Case Statements Template (RUST)). This language enables the possibility to map use case descriptions into Manchester OWL statements.

Resorting to our language it is also possible to extract several entities, namely the individuals present in specifications as well as their relationships. The final step required in order to achieve an ontology, is the specification of a set of classes, to which the individuals belong. After defining such classes (and associate the respective individuals) the ontology is produced. In order to gain the reasoning leverage needed for identifying the patterns, we propose to use the inference capabilities of an ontology (in this case the OWL) as this language allows us to not only specify ontologies and create instances, but also to perform queries over such knowledge. In order to extract requirement patterns, we specify them as SPARQL queries. These queries, when applied to the knowledge base, will allow to extract relevant patterns. Our approach allows us to formally describe use case scenarios, in order to perform, for instance, knowledge inference and formal validation. In order to support the proposed approach, we have previously developed the Use Cases Analysis Tool (uCat). The objective of our tool is to allow the users to input and validate the use case specifications (in the RUS format), and to automate the extraction of knowledge from those specifications (as an OWL ontology).

After developing the process, it remained to test the pro-

978-1-4799-6133-7/14 $31.00 © 2014 IEEE
DOI 10.1109/QUATIC.2014.23

IEEE computer society

cess' viability against real use cases and users. In this paper we address the viability of our approach. Its objective is twofold. In the one hand, we wanted to validate our language expressiveness capabilities, and ease of adoption and acceptance by users. On the other hand, we wanted to analyze how our tool performs in handling the language, and collect users' feedback. Above, we briefly presented the approach we are developing. Next, we describe how we propose to validate it. Preliminary results of this validation are also put forward.

II. Study

In order to assess both the proposed language (RUS) and the developed tool, we propose a study. We started by defining a set of questions about our language and tool. From the questions we extracted a set of tasks (actions that the user should perform, for instance *to write a use case description*) that will answer them. Finally we propose to collected data during and at the end of the modeling tasks.

A. Objectives

With the study, we proposed to test two topics: how our language performs with real users, and, how the presented tool performs at supporting the language. The objectives for our language are: **1.1.)** to provide formalism to use cases with minimal extra costs for the user (such includes for instance a seamless transition from natural language to RUS, without losing the meaning or expressiveness of the original statements; furthermore, we do not want to demand from users a background in formal methods); **1.2.)** to be able to support the users' specifications (our language should support the users' use case specifications needs, in order to make it viable; even if not all statements are supported, the language should at least support the most common; at the same time, we intend to improve the specification process, by encouraging the use of templates (or patterns), by demanding users to follow the RUS); **1.3.)** be easy enough to understand and manipulate according to the users' needs (if so, the users will be more likely to adopt it).

As the tool supports our language, tool specific objectives are somehow related, and may be elicited as: **2.1)** be easy to learn (this objective corresponds to a tool that is easy to use, and does not requiring an extensive adaptation period; we propose a familiar and self explanatory user interface, with familiar terminology). **2.2)** be a possible complement/substitute for other tools (we want to evaluate how likely it is that our tool will be able to complement or even substitute other UML supporting tools, regarding the formalism it provides); **2.3)** provide a good support for the language (the tool must be able to both support and improve the usage of our language).

B. Study setup

In order to evaluate the expressiveness of the language, a number of tasks was defined. A collection of use cases was defined on which these tasks should be carried out (see Table I). Each participant had a computer and a printed script of the study, and individually performed each task. Every task was previously explained, and then the users performed them. The steps were performed sequentially, and all the users performed them at the same time: 1) We have converted the use cases, into RUS beforehand; we performed the translation, in order to avoid any contact from the users with the language prior to the test. 2) The participants were asked to interpret and textually describe the use cases. 3) The textual descriptions were handed to the original authors, which evaluated them. 4) The participants were presented with the original use cases, and asked to point out any missing information from the RUS version.

TABLE I. Excerpt of a RUS use case

	User Input	**System Response**
1	user inserts name	
2		system searches tournament
3		system shows tournaments
4	user selects tournament	
5		system shows tournament
6	user selects remove	
7		system requests confirmation
8	user provides confirmation	
9		system removes tournament
10		system informs success

In order to evaluated the expressiveness of the language and the acceptance of the tool, the same users were asked to perform the following additional steps: 5) A new scenario describing a system's functionality was textually shown to the participants, which were asked to write the corresponding use case in Natural Language (NL), following Fowler's approach. 6) After presenting the tool and the language, the users were asked to convert the use case into RUS, with the tool. 7) A new scenario was presented, and the users asked to write it in RUS, on the tool. 8) The use cases were handed to other users, which interpreted them; each author evaluated the descriptions' correctness. 9) A RUS entry was presented, and the users were asked to write the corresponding RUST. In relevant tasks the time required to perform them was measured. At the end, a questionnaire was applied.

C. Addressing the objectives

This section relates the presented tasks with the proposed objectives. Objective 1.1 is addressed by measuring the overall time required by the users in order to adopt our language, and by how correct their specifications are. In tasks 2, 3 and 4, we measure the language acceptance, and in 6, 7 and 8 the time required by the users. Objective 1.2 is mainly evaluated by the reports about how the users understood our statements. Task 3 will measure how able is our language to support the use case specifications. We evaluate objective 1.3 with two approaches. First by measuring how valid the users' inputs are, regarding a provided set of RUST entries. Second by how the users were able to manipulate RUST and the corresponding required time.

In order to evaluate objective 2.1, we propose to measure the time spent using the tool, as well as the number of tries required in order to successfully create a use case specification. In objective 2.2 we propose to rely in the users' feedback, by asking them the likelihood to adopt the language. Objective 2.3 is measured by the capability of the users to write the use case specification (overall number of supported statements) and take advantage of the language's capability (for instance, alternatives and exceptions).

III. Preliminary results

In order to achieve a first set of results, and understand the study setup, we performed a preliminary test with five participants. The participants were students from an Informatics Engineering course, at the University of Minho. They had obtained final grade on the 80th percentile in a previous software engineering course. They all had previous contact with the use cases tabular representation, as proposed by Fowler. The participants performed the study in an isolated environment and without interaction with each other. None of them had previous contact either with RUS, the tool or even our work. As for the use cases, they were written by the participants as part of an assignment for a course.

In tasks 1, 2 and 3, we concluded that all of the users were able to correctly describe the presented RUS. Only a minor issue was pointed out in one of the descriptions, referring that the context might not be clear enough. Such issue would be solved by presenting the use case title (which was intentionally removed). Regarding the understandability of RUS, in task 4 all the participants reported that there was no information missing from the RUS statements. Only one of the participants reported a missing detail. The missing information concerned a platform specific detail. This was not found significant, as such details are not intended to be present in use case specifications. These results are directly related with objective 2.1. The participants were able to both understand and express the use cases without major issues.

Tasks 5, 6 and 7 allowed us to achieve several conclusions. First, the participants required an average of 23 seconds per statement (s/s) when creating the use cases in NL. When writing the same description in RUS, resorting to our language for the first time, they required an average of 51s/s. However, the second time writing the use cases in RUS, the participants required an average of 44s/s (5s/s less). This is an indication that through learning users are able to reduce the required time to write the statements. In order to achieve further conclusions a more extensive study is required.

In task 8 all of the descriptions matched the corresponding use case. Only in one of the use cases a minor issue was pointed out, but assumed as as mistake by the use case author. This capability of the participants to convert the use case specifications, or even write new ones in the tool is related with task 1.1. We can conclude that only a reduced learning time is required by the participants.

Finally, task 9 allowed us to conclude about the manipulation of RUST. Users had an average of 88% correct answer, in an average of 94 seconds to write each RUST statement. The overall results of this task give us hints of how easy the language was to use and manipulate, as proposed by objective 1.3.

Overall, we can draw two main conclusions. First, the initial results suggest that the users are receptive to our approach. Also, the overhead to achieve formal use cases (either from existing ones, or new ones) seems very low. Second, the presented study seems to be appropriate to a large set of participants. The users were able to perform the presented tasks independently, and only sporadic questions were raised, for instance about the meaning of a sentence. Such questions allowed us to improve our tasks descriptions and write them in a clearer way. We concluded also that our tool is able to support the presented tasks, while only minor bugs were pointed out, which will be solved for a next study.

TABLE II. Questionnaire results (a)

	Question (`0 to 7`)	**Avg.**
1	Number of statements which required major changes in order to be mapped into RUS	1.6
2	Number of statements which lost their meaning	0.2
3	Number of unsupported statements	0
16	Minutes spent in adjustments (to match RUS)	3.8

TABLE III. Questionnaire results (b)

	Question (`low to high`)	**Mode**
4	How much sense does the user interface makes	6
5	How familiar was the terminology	6, 7
6	How much the tool helps in the specification	5, 6
7	How easy to use is the RUS	6
8	How easy to understand is the RUS	4
9	How much easier is RUS to understand than NL	4, 6
10	How easy is it to manipulate RUST	4
11	Is NL easier to use than RUS	2
12	Is NL easier to understand than RUS	5
13	Likelihood to adopt RUS	6
14	How easy is it to understand RUST	5
15	How clear is the language	5
17	How close to NL is RUS	3
18	How easy was it to understand the tool	5, 6
19	How this tool is preferred over VP	6, 7
20	How useful and adequate was the output	5
21	How acceptable are the tools' limitations	6
22	How easy to use is the tool	6
23	How much the user liked the language	6
24	How much the user liked the tool	6

The questionnaire included questions to be answered in a 7-point Likert scale (with 0 meaning low and 7 meaning high – see Table III). Other questions were answered with a numeric value (see Table II).

Regarding objective 1.1, we have questions 9, 11, 12 and 17. The users' answers show that they would like to write the statements in RUS over NL. Questions 1, 2 and 3 are indicators for objective 1.2. The average value of 0.6 statements raising issues shows that the language has a good support for the statements. Furthermore, in question 23 the users expressed empathy with the language itself. Objective 1.3 is related with questions 7 to 15. The reported results suggest that the users consider the tool to be easy to understand and use, even when compared with NL.

We consider questions 5, 18, 20, 22 to be directly related with objective 2.1. The mode value for this question is 6, which means a good acceptance by the users, being a hint about how the users consider the tool easy to learn and use. Also, we consider the results of question 24 as an indicator of acceptance by the users. Questions 4, 19, 21 are considered directly related with the objective 2.2. Its mode is 6, showing that the participants are highly receptive as replacing/complement other tools with ours. The objective 2.3 is related with the questions 6, 20, 22. The mode values for these questions are 5 and 6, showing that overall, the participants consider the tool able to support the language.

The questionnaire included also open questions. The first question was *"What became harder by using RUS (over NL)?"*. 60% of the participants pointed out the need to convert the statements, 20% the lack of explicit support for white spaces in the statements, and the remaining 20% answered nothing. However, when asked *"What became easier by using RUS?"*, 80% of the participants referred that the statements are overall easier and simpler to understand. 20% pointed out also that the statements are quicker to understand due to its simplicity, and 20% also referred formalization capability.

The users were also asked *"What did they like in the language?"*, at which 40% pointed out the language simplicity to write the statements. The remaining 60% stated that they liked the standardization/restriction of the format which makes them more concise. When asked *"What did they dislike in the language?"*, 80% of the users said that they disliked nothing, while the remaining 20% referred the required learning time.

Regarding only the capability to write use cases, the users were asked why they preferred the RUS format over tools which allow free text format input (as for instance, Visual Paradigm). 80% of the users referred the user interface is simpler or more user friendly, and the remaining 20% stated that the organization of the scenarios is better (in tabs).

The users where asked *"What did they like the most in the tool"*, at which 40% answered the interface simplicity to support the specifications. Also, 40% of the users pointed the RUS support and verification capability. The remaining 20% pointed the similarity to Fowler's template. When asked what did they disliked, 80% didn't point any issue, and the remaining 20% referred the inability to see all the three scenarios (main, alternative and exception) in the same interface.

IV. Conclusions and Future Work

In this paper we have introduced our approach to formalize use cases in OWL, and to support the identification of requirements patterns in these descriptions. We presented also the study designed to validate our language and tool capabilities. Our approach starts with the definition of a restricted natural language for requirements specification. Such format, the RUS, enables us to specify how user input should be expressed and how the resulting use cases can be transformed into an OWL ontology. The transformation rules provide us with most of the required information to create the ontology. Resorting to OWLs' query engines, we are able to perform queries over the ontology to identify requirements patterns in the use cases. A prototype tool was developed to support the process. We have previously illustrated our approach in [1].

Based in [1], we have presented a study to validate our approach. The preliminary results suggest that our approach is simple and easy to adopt, while providing formalization to requirements. The users have shown no difficulties in adapting to this format or even manipulating it. Regarding the tool, we concluded that it is able to support the language and even improve the requirements specification process. It was easily accepted and the learning time was relatively low, as the users were able to use it without previous learning time. The feedback provided by the users was positive, and overall they reported a good experience with both the tool and the language. Generally the comments reported how the participants liked the language's simplicity and objectivity, and how the formalism is a desired benefit.

Building on the results thus far, we plan to perform a larger user study in order to obtain more reliable results. We are also interested in further exploring the evaluation of the RUS language. A possible approach would be to use a full system specification, and map it into our format. This would enable also the possibility to create a catalog of patterns, which we aim too, as our approach is part of a wider project. To help in this process, we propose to focus on a specific domain in order to write and process the specifications. We propose to associate the requirements patterns with architectural patterns. Then, by extracting the requirements patterns it would be possible to obtain hints about the final system. In order to improve our approach, it is proposed the possibility to extract other kind of data from the specifications, such as behavioral informations.

Acknowledgment

This work is partly funded by project ref. NORTE-07-0124-FEDER-000062, co-financed by the North Portugal Regional Operational Programme (ON.2 - O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by national funds, through the Portuguese foundation for science and technology (FCT). We would like to thank to the participants which helped us in the study, namely Pedro Ferreira, Filipa Rocha, Mariana Capelo, Henrique Pacheco and Daniel Coelho.

References

[1] R. Couto, A. N. Ribeiro, and J. C. Campos, "Application of ontologies in identifying requirements patterns in use cases," in *11th International Workshop on Formal Engineering approaches to Software Components and Architectures*, ser. Fesca'14, Grenoble, France, 2014.

[2] W. W. Royce, "Managing the development of large software systems: Concepts and techniques," in *Proceedings of the 9th International Conference on Software Engineering*, ser. ICSE '87. Los Alamitos, CA, USA: IEEE Computer Society Press, 1987, pp. 328–338.

[3] I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard, *Object-Oriented Software Engineering - A Use Case Driven Approach*. Addison-Wesley, 1992.

[4] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[5] A. Cockburn, *Writing Effective Use Cases*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000.

[6] V. Castaeda, L. Ballejos, M. L. Caliusco, and M. R. Galli, "The use of ontologies in requirements engineering," *Global Journal of Researches In Engineering*, vol. 10, no. 6, 2010.

[7] D. L. McGuinness and F. van Harmelen, "OWL web ontology language overview, W3C recommendation," 2004, http://www.w3.org/TR/owl-features/.

[8] D. Kirasić and D. Basch, "Ontology-based design pattern recognition," in *Proceedings of the 12th international conference on Knowledge-Based Intelligent Information and Engineering Systems, Part I*, ser. KES '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 384–393.

[9] R. Schwitter, "Controlled natural languages for knowledge representation," in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, ser. COLING '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 1113–1121.