A Use Case Textual Description for Context Aware SPL Based on a Controlled Experiment

Ismayle de Sousa Santos[1] *, Rossana Maria de Castro Andrade[1] **, and Pedro de Alcantara dos Santos Neto[2]

[1] Federal University of Ceará, Fortaleza, Ceará, Brazil
[2] Federal University of Piauí, Teresina, Piauí, Brazil
ismaylesantos@great.ufc.br,rossana@great.ufc.br,pasn@ufpi.edu.br

Abstract. A Software Product Line (SPL) that aims to develop context aware applications leaves challenges inherent to the SPL-based development approach and the type of application being developed (context-aware). These challenges leave open issues, like those related to how the context influences should be described. In the literature, there are studies that support the development of traditional SPL by using templates for use case description in a text style with variability information. However, a template that considers context awareness in SPL was not found, and thus, this paper proposes a use case template for a Context Aware SPL based on the results of a controlled experiment conducted with four of the use case templates for traditional SPL found in the literature. This paper also presents a preliminary evaluation of the proposed template.

Keywords: SPL, Use Case, Controlled Experiment, Context Awareness

1 Introduction

A Software Product Line (SPL) can be used to increase the reuse degree during application development. Some benefits of using SPL are [1]: i) productivity improvement; ii) best time to market; and iii) higher product quality. When the SPL is focused on the development of context-aware applications, which are applications that adapt their behavior based on context information [2], they are called Context-Aware Software Product Line (CASPL) [3, 4].

In this scenario, the challenges related to SPL-based development and the introduction of one more characteristic (context-awareness) make the development even more challenging than the traditional SPL development (related to non context-aware applications).

Due to the peculiarities of SPL Engineering, adaptations are required for the traditional requirements engineering. In this scenario, different approaches for use case textual description for SPL are found in the literature, but there is no template for textual description of use case for CASPL.

* Master Scholarship (MDCC/DC/UFC) sponsored by CAPES
** Researcher scholarship - DT Level 2, sponsored by CNPq

It is important to highlight that in context-aware applications there is a new input (concerning the context information) that could affect the behavior of these applications anywhere during execution [2]. So, it would be interesting to incorporate context information into the use case descriptions to catch benefits similar to the benefits of variability modeling in use cases [5].

The main contribution of this paper is to propose a CASPL use cases template based on results of a controlled experiment conducted with existing SPL use case templates.

This paper is then structured as following. In Section 2, the main results of the experiment performed to evaluate four SPL use case templates are described. Section 3 presents the proposed template for textual description of use cases for CASPL and the preliminary evaluation of this template. Section 4 describes the related work. Finally, Section 5 concludes the paper and presents future directions.

2 Experimental study

As mentioned before, we have not found a template suitable for use case description of CASPL. Due to this fact, we have conducted a controlled experiment to evaluate some of the existing SPL use case templates looking for elements to help with the definition of a suitable template for CASPL use cases.

In order to find templates for textual description of SPL use cases, a search was conducted in the databases of the ACM Digital Library and IEEE Explorer, and also in Google and Google Scholar for *gray literature*. As a result, we have identified eight templates for textual description of SPL use cases [6, 5, 7–12].

The execution of an experimental study with all the templates found would be costly, since each template requires a lot of training and that would be time consuming. Thus, two selection criteria were established to define which templates would be used in the experiment: a) the template should not model the final product in the specification, since this reduces the maintainability; b) the template should not depend on another model or tool, because this limits its use. As a result, the Bertolino et al. [6] template was excluded (item a) as well as the template proposed by Choi et al. [11], Galina et al. [9], and Anthosnysamy and Somé [10] (item b). The remaining four templates were selected for the experiment: John and Muthing [5], Gomma [7], Eriksson et al. [8] and Bonifácio and Borba [12].

Forty eight (48) volunteers have participated of this experiment. Twenty one were undergraduate Computer Science students, 20 were graduate students (16 MSc and 4 PhD) in Computer Science and seven were developers working at GREat - Group of Computer Networks, Software Engineering, and Systems [3].

In the remainder of this Section, the main results of this experiment are presented. These results are relevant to the proposal of the use case template for CASPL.

[3] http://www.great.ufc.br

2.1 Experiment Results

The study started with 48 subjects; however, some of the tasks were not approved, since the checking question of the questionnarie was not answered correctly. This checking question was associated with an important question about the use case understanding. A wrong answer in this question signaled a serious error in the use of templates, preventing the use of the data associated with a specific task. Due to this fact, there are only 134 executions able to be evaluated, since 58 executions were not approved.

The results showed that the accuracy and the time spent to perform the activities were better for the developers against the graduate and undergraduate students. The graduate students had better results than undergraduate students. This is an expected result confirmed by the experiment.

The template proposed by Eriksson et al. had better results in the experiment. The subjects who used this template spent less time and had more accuracy than the subjects using others templates. In terms of participants' preference, it was also the favorite, with a 46.3% rating. The second best template was proposed by Gomma, although the developer group had better results when using the template proposed by John and Muthing. The Gomma template had a preference of 29.1% of the experiment volunteers and the John and Muthing one had a preference of 21.6%. The template proposed by Bonifácio and Borba had the worst results for all the groups and was preferred by only 3% of the volunteers.

Based on the observation of volunteers and the experiment results, it was possible to look at which characteristics of each template generated impact on the understanding of a use case. Subjects who selected the Eriksson template as the best template, for example, reported that it possessed a simple description, and an objective and compact structure. These characteristics made it easy and intuitive to identify whether the step was mandatory, optional, or alternative.

On the other hand, the disapproval related to the Bonifácio and Borba template may be justified due to the separation among the main flow of the use case and their variations without an explicit definition about the variation type, making it difficult to understand if the variation is optional or alternative.

Moreover, it is worth mentioning that, as noticed during the experiment and commented by one of the volunteers, the use case that describes the variability along with the commonalities makes it easier to understand its operation, while the use case that describes separately the variabilities of the commonalities makes it easier for the recognition of the variabilities only. More details about this controlled experiment could be found in [13].

3 A template proposal for CASPL use case

The results of the experimental study helped us to propose the CAPLUC (**C**ontext **A**ware Software **P**roduct **L**ine **U**se **C**ase template), which allows the definition of product functionalities, variability, and the influence of context on the products of the CASPL.

The CAPLUC was based on the main advantages of the templates that were used in our experimental study. It is defined by a tuple: [Name, Use Case Extended, Extension Point, Reuse Category, Context Constraint, Summary, Actors, Precondition, Steps, Post-condition, Alternative Flows and Summary of Variations]. Most elements of the tuple are common to use case templates and work in the same way. It is noteworthy that when there are large variations, the use of a use case extension mechanism, as presented in Gomma [7], is recommended. On the other hand, small variations can be described as variation points within the use cases. We describe in detail some elements of the CAPLUC, as follows:

- Reuse Category: this element works like the Gomma proposal and serves to identify whether the use case is mandatory, optional or alternative.
- Context Constraint: specifies the context in which the use case is applicable.
- Steps: follows the tabular style proposed by Eriksson et al [8]. However, we propose the use of the "*" symbol to indicate the steps that had context constraint, besides the indication of the variabilities in the step indexes.
- Summary of Variations: this section was proposed based on the Gomma template [7]. All the variabilities are described at the end of the use case. The idea is that this section can help with the quick identification of the variations that affect the use case. In this case, each variation would be a tuple [Variation Point, Variability Type, Descriptive Question, Variants], where variants can also have context constraints associated. With respect to Descriptive Question, it was based on the "John and Muthing" template [5] and serves both to facilitate the understanding of the variations and as a mechanism that can be used to facilitate the instantiation of products by questions.

Figure 1 presents a use case modeled in CAPLUC. This use case is from the CASPL known as *Mobiline SPL* [4], whose main goal is to develop guides for context-aware mobile tours. As can be seen in the figure, this use case is comprised of a variation point (Capture) that had three variants (Sensor, Memory, External Service). The steps concerning this variation point have the same step index to indicate that they are alternative and the one related to External Service has a * indicating that it has a context constraint associated (Internet Connection). The use case of the figure describes that the capture of context can be done by a Sensor, Memory or External Service, but the latter may be chosen only if there is an internet connection.

3.1 Preliminary Evaluation

In order to evaluate the CAPLUC we conducted a preliminary evaluation with seven volunteer students from Federal University of Ceará. Two were undergraduate Computer Science students and four were master students in Computer Science. The purpose of this preliminary assessment was to verify whether CAPLUC makes easier the understanding of a CASPL use case.

In this evaluation, the first activity was to answer a pre-experiment questionnaire in a similar way as was done in the experiment described in Section

Element		Description	
Name		Context Capture	
Reuse Category		Mandatory	
Context Constrain		--	
Summary		Capture context informations	
Actors		Context Manager	
Precondition		User must be logged	
Postcondition		Context information was acquired	
Step		**User**	**System**
	1	--	System require to Context Manager some context information
[Capture] Alt [Sensor]	**2**	--	Sensor returns the context information
[Capture] Alt [Memory]	**2**	--	The Memory returns the context information
[Capture] Alt [External service]	**2***	--	Na External Service returns the context information
Alternative Flows			
(Constrain)		(Steps)	
Summary of Alternative Variations			
[Capture]: "What is the mechanism to get context informations?"		**[Sensor]** (Context Constrain: null)	Step 02
		[Memory] (Context Constrain: null)	Step 02
		[External Service] (Context Constrain: Internet Connection)	Step 02
Summary of Optional Variations			

Fig. 1. Example of a use case described using the proposed template.

2. The results from this questionnaire revealed that the four graduate students knew the basic concepts of SPL, CASPL and textual descriptions of use cases. In the case of the two undergraduate students, both had not studied anything about textual descriptions of use cases and just one knew the basics of SPL.

In the following activity, a training session was conducted trying to explore the structure of CAPLUC and CASPL use case description. Also, in this training session, some exercises were done to ascertain if everyone understood the structure of the CAPLUC.

After the training session, the participants performed two comprehension tasks (Task 1 and Task 2) to evaluate the CAPLUC. In both tasks, the goal was to analyze one CASPL use case described with CAPLUC and answer some questions about variability and context information. It is important to mention that we did not use the same use cases of the templates experiment, because these were not of a CASPL.

With the comprehension tasks, the time taken to answer the questionnaire, as well as the number of correct responses, was collected to evaluate the volunteer performance with CAPLUC.

Figure 2 presents the collected results related to Task 1 and Task 2. For Task 1, two of the six volunteers missed the checking question, which was related to the understanding of the use case. It is noteworthy that an error in these questions signaled that the volunteer could be making an incorrect use of the template.

	Task 1 (Use Case Context Capture)					Task 2 (Use Case Show Documents)				
Volunteer	What are the optional steps?	What are the alternative steps?	What steps have context constrains?	Checking Question	Time (min)	What are the optional steps?	What are the alternative steps?	What steps have context constrains?	Checking Question	Time (min)
1	1	1	1	0	10	1	1	1	0	7
2	1	1	1	1	6	1	1	1	1	6
3	1	1	1	1	5	1	0	1	0	4
4	1	1	0	1	8	1	1	1	1	7
5	1	1	1	1	5	1	1	1	0	4
6	1	0	0	0	15	0	0	1	0	10

Legend
0: wrong
1: correct

Fig. 2. Results of the preliminary evaluation of the CAPLUC

Therefore, disregarding samples related to these two volunteers, from the other four, only one (Volunteer 4) did not reach the maximum accuracy; he missed a question. As a result, for Task 1, the mean accuracy of the four participants who answered the questions correctly was 91.5%. Moreover, the average of these four volunteers was six minutes to accomplish this task.

With respect to Task 2, only two volunteers answered correctly the checking question, which signaled that either the other volunteers felt some difficulty in the use of the template, or that this question had some problems. After reviewing the incident and through dialogue with some volunteers, it was realized that the small difference between the alternatives of the checking question may have led to the large number of errors in this question. Considering only the volunteers who answered the checking question correctly, the accuracy achieved was 100% and the average time execution of this task was six minutes and 30 seconds.

Thus, the time and accuracy data indicated that the volunteers using the CAPLUC spent an average 6.25 minutes and had good accuracy, with almost everyone who answered correctly the checking question reaching 100%.

Finally, after the two tasks, the subjects were asked whether they believed the template helped to understand the use case. All volunteers said that CAPLUC helped the understanding of the CASPL use case. In general, the explanation given by participants was that the CAPLUC characteristics with respect to description of optional features, alternative features, and context constraints made the task of locating them easy.

It is interesting to mention that one of the volunteers also highlighted that he had read the descriptive questions associated with variation points in CAPLUC for a better orientation. Moreover, another volunteer said that the use of indexes to indicate the points with variability and context were interesting, but that the summary of variation also helped a lot.

Therefore, the evaluation results, through data collected and the responses of volunteers, represent an initial evidence that CAPLUC, proposed in this paper, helps the understanding of a use case of a small CASPL. However, to generalize this statement for every use case of a CASPL an evaluation is still needed with a larger quantity of volunteers and a greater diversity of use cases.

4 Related work

As mentioned before, the main characteristic of an SPL is the presence of variability. Based on that, proposals have been made to address this variability in artifacts created in an SPL development process.

As for to the context modeling in use cases for Context-Aware SPL, we have not found any work. However, when considering context aware standalone applications, there are works such as Yang [14], which suggests the description of the context influences on the application behavior as alternative flows with context constraints.

In terms of SPL use cases, we have found 8 templates that allow modeling of variability in textual specifications. Six of these eight templates considered in this research present variability and commonalities in the same use case description. Two of these six templates have merged commonalities and variabilities [5, 8]. The other four templates have specified the variabilities separately, and the linkage with the commonalities is made by using tags/variables [6, 9, 11] or using the description of the line numbers that could be affected [7].

The others two templates [10, 12] present variability described separately as advices use case, which would be use cases that modify the common use cases.

5 Conclusions and future work

This paper presents the use of the results of a controlled experiment with four templates for textual use case description of a Software Product Line to propose a use case template for Context Aware SPL.

The goal of this experiment was to evaluate the level of understanding provided by the templates' use. As a main result of this experiment, we have collected evidence that the description of commonalities together with variabilities makes the understanding of a use cases more clear. Furthermore, the description of all variations at the end of the use case promotes a quick identification of variations.

Based on the experiment results, a proposal of a template for textual use case description for Context-Aware SPL, called CAPLUC, was presented. The idea of this template is to allow the incorporation of context information in the textual use case description, since the context will define which use cases or use case variation should be executable or not. A preliminary evaluation of CAPLUC showed evidence that this template help the understanding of CASPL use cases.

As future work we intend to evaluate the proposed template with more volunteers and a greater diversity of use cases. Furthermore, a tool for requirement testing support based on this template will be developed. This tool will use the context information specified in the CASPL use cases to generate test scenarios that take into account the influence of context on behavior of the CASPL final products.

6 Acknowledgments

This work is a partial result of the UbiStructure project supported by CNPq (MCT/CNPq 14/2011 - Universal) under grant number 481417/2011-7 and the Maximum project supported by FUNCAP (FAPs/INRIA/INS2i-CNRS 11/2011).

References

1. Northrop, L.M., Clements, P.C.: A framework for software product line practice, version 5.0. Technical report, CMU/SEI - Software Engineering Institute (2007)
2. Wang, Z., Elbaum, S., Rosenblum, D.S.: Automated generation of context-aware tests. In: Proceedings of the 29th international conference on Software Engineering. ICSE '07, Washington, DC, USA, IEEE Computer Society (2007) 406–415
3. Fernandes, P., Werner, C., Teixeira, E.: An approach for feature modeling of context-aware software product line. Journal of Universal Computer Science **17**(5) (March 2011) 807–829
4. Marinho, F.G., Lima, F., Filho, J.a.B.F., Rocha, L., Maia, M.E.F., de Aguiar, S.B., Dantas, V.L.L., Viana, W., Andrade, R.M.C., Teixeira, E., Werner, C.: A software product line for the mobile and context-aware applications domain. In: Proceedings of the 14th international conference on Software product lines: going beyond. SPLC'10, Berlin, Heidelberg, Springer-Verlag (2010) 346–360
5. John, I., Muthig, D.: Product line modeling with generic use cases, splc-2 workshop on techniques for exploiting commonality through variability. In: Management, Second Software Product Line Conference. (2002)
6. Bertolino, A., Fantechi, A., Gnesi, S., Lami, G., Maccari, A.: Use case description of requirements for product lines. In: REPL. (September 2002)
7. Gomaa, H.: Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA (2004)
8. Eriksson, M., Börstler, J., Borg, K.: Marrying features and use cases for product line requirements modeling of embedded systems. In: Proceedings of the Fourth Conference on Software Engineering Research and Practice in Sweden. (2004)
9. Gallina, B., Guelfi, N., Monnat, A., Perrouin, G.: A template for requirement elicitation document of software product lines. Technical report, Laboratory for Advanced Software Systems, University of Luxembourg (2006)
10. Anthonysamy, P., Somé, S.S.: Aspect-oriented use case modeling for software product lines. In: Proceedings of the 2008 AOSD workshop on Early aspects. EA '08, New York, NY, USA, ACM (2008) 1–8
11. Choi, W., Kang, S., Choi, H., Baik, J.: Automated generation of product use case scenarios in product line development. In: Proceedings of the 8th IEEE International Conference on Computer and Information Technology. (2008)
12. Bonifácio, R., Borba, P.: Modeling scenario variability as crosscutting mechanisms. In: Proceedings of the 8th ACM international conference on Aspect-oriented software development. AOSD '09, New York, NY, USA, ACM (2009) 125–136
13. Santos, I.S.: An environment for generation of testing scenarios for context aware software product lines. Master's thesis, Federal University of Ceará (2013)
14. Yang, H.: Context-driven requirement analysis and implementation of adaptable is. Master's thesis, Faculty of Computer Science, University of Magdeburg (2010)