The UML «extend» Relationship as Support for Software Variability

Sofia Azevedo[1], Ricardo J. Machado[1], Alexandre Bragança[2], and Hugo Ribeiro[3]

[1] Universidade do Minho, Portugal
{sofia.azevedo,rmac}@dsi.uminho.pt
[2] Instituto Superior de Engenharia do Porto, Portugal
alex@dei.isep.ipp.pt
[3] Primavera Business Software Solutions, Portugal
hugo.ribeiro@primaverabss.com

Abstract. The development of software product lines with model-driven approaches involves dealing with diverse modeling artifacts such as use case diagrams, component diagrams, class diagrams, activity diagrams, sequence diagrams and others. In this paper we focus on use cases for product line development and we analyze them from the perspective of variability. In that context we explore the UML (Unified Modeling Language) «extend» relationship. This work allows understanding the activity of use case modeling with support for variability.

Keywords: use case, software product line, variability, «extend», alternative, option, specialization.

1 Introduction

Use case diagrams are one of the modeling artifacts that modelers have to deal with when developing product lines with model-driven approaches. This paper envisions use cases according to the perspective of variability. The *«extend»* relationship plays a vital role in variability modeling at the level of use cases and allows for the use case modeling activity to be applicable to the product line software development approach. This paper's contribute is on the understanding of the use case modeling activity with support for variability. We will illustrate our approach with some examples from the Fraunhofer IESE's GoPhone case study [1], which presents a series of use cases for a part of a mobile phone product line. We will propose different kinds of variability in use case diagrams.

The paper is structured as follows. Section 2 elaborates on the differences between others' approaches to variability modeling and ours. Section 3 analyzes the differences between our approach to modeling different types of variability with the UML (Unified Modeling Language) *«extend»* relationship and others' approaches. Section 4 illustrates our approach with some examples from the GoPhone. Finally Section 6 provides for some concluding remarks.

2 An Outline of Software Variability Modeling

Despite use cases being sometimes used as drafts during the process of developing software and not as modeling artifacts that actively contribute to the development of software, use cases shall have mechanisms to deal with variability in order for them to have the ability to actively contribute to the process of developing product lines. For instance modeling variability in use case diagrams is important to later model variability in activity diagrams [2]. In this paper we shortly talk about alternative, specialization and option use cases as the representation of the three variability types we propose to be translated into stereotypes to mark use cases.

We consider that product line modeling shall be top-down (rather than bottom-up), which means that the product line shall support as many products as possible within the given domain. In [3] Bayer, *et al.* refer that all variants do not have to be anticipated when modeling the product line. In [4] John and Muthig refer to required and anticipated variations as well as to a planned set of products for the product line, which indicates that their approach to product line modeling is bottom-up. A bottom-up approach would consider that all the products from the product line are known *a priori*. Bragança and Machado [5] represent variation points explicitly in use case diagrams through extension points. Their approach to product line modeling is bottom-up because they comment *«extend»* relationships with the name of the products from the product line on which the extension point shall be present.

In [4] John and Muthig refer the benefits of representing variability in use cases. Although we totally agree with the position of these authors towards those benefits, we cannot agree when they state that information on whether certain use cases are optional or alternatives to other use cases shall only be in decision models as it would overload use case diagrams and make them less readable. Our position is that features as well as use cases shall be suited for treating variability in its different types. Bachmann, *et al.* mention in [6] that variability shall be introduced at different phases of development of product families. If a use case is an alternative to another use case, then both use cases shall be modeled in the use case diagram, otherwise the use case diagram will only show a part of the possibilities of the possible products John and Muthig mention in [4].

Coplien, *et al.* defend in [7] the analysis of commonality and variability during the requirements analysis in order for the analysis decisions not to be taken during the implementation stage by the professionals who are not familiar with the implications and impact of decisions that shall be made much earlier during the development cycle. They refer that early decisions on commonality and variability contribute to large-scale reuse and the automated generation of family members.

Maßen and Lichter talk about three types of variability in [8]: optional, alternative and optional alternative (as opposite to alternatives that represent a "1 from n choice", optional alternatives represent a "0 or 1 from n choice"). In this context they propose to extend the UML metamodel to incorporate two new relationships for connecting use cases. Our approach considers options and alternatives as well but we introduce these concepts into the UML metamodel through stereotypes (we consider that the *«extend»* relationship is adequate for modeling alternatives and a stereotype applicable to use cases for modeling options).

3 Different Perspectives on the *«extend»* Relationship

Gomaa and Shin [9] analyze variability in different modeling views of product lines. They mention that the *«extend»* relationship models a variation of requirements through alternatives. They also model options in use case diagrams by using the stereotype *«optional»* in use cases. We adopt these approaches to alternatives and options but we elaborate on another form of variability (specializations, which we consider to be a special kind of alternatives; Gomaa and Shin refer specialization as a means to express variability in [9]). Besides alternative and optional use cases, Gomaa and Shin consider kernel use cases (use cases common to all product line members). Gomaa models in [10] kernel and optional use cases both with the *«extend»* as well as with the *«include»* relationships (our approach is towards modeling kernel and optional use cases independently of their involvement in either *«extend»* or *«include»* relationships).

Halmans and Pohl propose in [11] use cases as the means to communicate variability relevant to the customer. Halmans and Pohl consider that generalizations between use cases are adequate to represent use cases' variants. This is not our position. We recommend to use the *«extend»* relationship instead of the generalization relationship. Halmans and Pohl consider that modeling mandatory and optional use cases with stereotypes in use cases is not adequate because the same use case can be mandatory for one use case and optional for another. Again this is not our position. We consider that a mandatory use case is not mandatory with regards to another use case, rather it is mandatory for all product line members. We also consider that an optional use case is optional with regards to one or more product line members.

Fowler suggests in his book "UML Distilled" [12] that we ignore the UML relationships between use cases besides the *«include»* and concentrate on the textual descriptions of use cases. We completely agree with Fowler on the textual descriptions but we cannot agree with the rest. The *«extend»* relationship is needed in order to formalize at an early stage (the use case modeling) where variation will occur when instantiating the product line. Bosch, *et al.* mention in [13] the need for describing variability within different modeling levels such as the requirements one.

4 Variability Types in the GoPhone Case Study

We consider that the *«extend»* relationship is adequate for modeling alternatives and specializations, and a stereotype applicable to use cases for modeling options. The three variability types we propose to be translated into stereotypes to mark use cases are: alternative, specialization and option. From now on we either use the *«extend»* relationship without stereotypes or with one of the two stereotypes applicable to this relationship (depending on whether we are modeling alternatives or specializations). We propose the stereotypes *«alternative»*, *«specialization»* and *«option»* to distinguish the three variability types. We also propose the stereotype *«variant»* to mark use cases at higher levels of abstraction before they are realized into alternatives or specializations. The stereotypes *«alternative»* and *«specialization»* shall be applicable to the *«extend»* relationship for modeling alternatives and specializations respectively, and the stereotype *«option»* shall be applicable to use cases that represent options. Figure 1 shows some examples of alternative, specialization and option use cases from the GoPhone's message sending functionality.

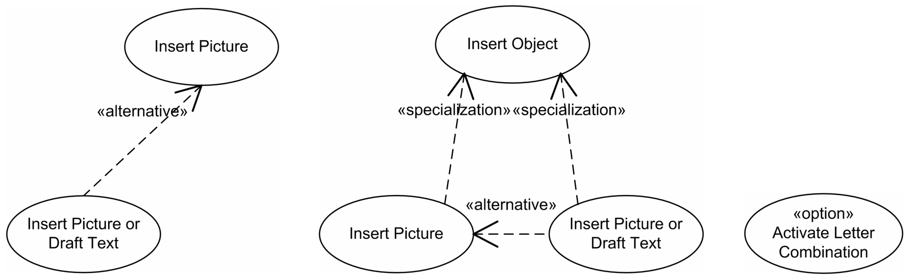

Fig. 1. Some examples of alternative, specialization and option variability types from the GoPhone's messaging domain

Consider that an extending use case is a use case that extends another use case and that an extended use case is a use case that is extended by other use cases. In the context of alternatives both extending and extended use cases represent supplementary functionality since both represent alternatives, which are not essential for a product without variability to function. If we have more than one alternative use case for the same functionality, one of those use cases shall be the alternative to all the others and extended by them. That use case is the one to be present in the products less robust in terms of functionality.

If the intention is to use differential specification, specializations shall be modeled with the *«extend»* relationship, otherwise they shall be modeled with the generalization relationship. Differential specification of specializations means that specialization use cases represent supplementary functionality regarding the use case they specialize, therefore a product without variability does not require the specialization use cases to function.

Options represent functionality that is only essential for a product with variability to function, therefore options represent supplementary functionality. However we do not recommend modeling options with the *«extend»* relationship because if the stereotype was on the relationship, the relationship itself would be optional and that is not the case (the use case is not optional with regards to any other use case, rather it is optional by itself). Options shall be modeled with a stereotype in use cases. The involvement of an option use case in either *«extend»* or *«include»* relationships, or even in none of those does not imply the presence of that use case in all product line members (which makes of it optional).

5 Conclusions

This paper has elaborated on the representation of variability in use case diagrams. It began by providing an analysis of the state-of-the-art concerned with this topic. Based on our position towards the related work we proposed three variability types to be translated into stereotypes to mark use cases: alternative, specialization and option. We proposed both alternative and specialization use cases to be modeled with the *«extend»* relationship, and a stereotype applicable to use cases for modeling option use cases.

References

[1] Muthig, D., John, I., Anastasopoulos, M., Forster, T., Dörr, J., Schmid, K.: GoPhone - A Software Product Line in the Mobile Phone Domain, Fraunhofer IESE, IESE-Report No. 025.04/E (March 5, 2004)

[2] Bragança, A., Machado, R.J.: Extending UML 2.0 Metamodel for Complementary Usages of the «extend» Relationship within Use Case Variability Specification. In: 10th International Software Product Line Conference (SPLC 2006). IEEE Computer Society, Baltimore (2006)

[3] Bayer, J., Gerard, S., Haugen, Ø., Mansell, J., Møller-Pedersen, B., Oldevik, J., Tessier, P., Thibault, J.-P., Widen, T.: Consolidated Product Line Variability Modeling. In: Käköla, T., Duenas, J.C. (eds.) Software Product Lines - Research Issues in Engineering and Management, pp. 195–241. Springer, Heidelberg (2006)

[4] John, I., Muthig, D.: Product Line Modeling with Generic Use Cases. In: Workshop on Techniques for Exploiting Commonality Through Variability Management. Springer, San Diego (2002)

[5] Bragança, A., Machado, R.J.: Deriving Software Product Line's Architectural Requirements from Use Cases: An Experimental Approach. In: 2nd International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES 2005). TUCS General Publications, Rennes (2005)

[6] Bachmann, F., Goedicke, M., Leite, J., Nord, R., Pohl, K., Ramesh, B., Vilbig, A.: A Meta-model for Representing Variability in Product Family Development. In: van der Linden, F.J. (ed.) PFE 2003. LNCS, vol. 3014, pp. 66–80. Springer, Heidelberg (2004)

[7] Coplien, J., Hoffman, D., Weiss, D.: Commonality and Variability in Software Engineering. IEEE Software 15, 37–45 (1998)

[8] Maßen, T.v.d., Lichter, H.: Modeling Variability by UML Use Case Diagrams. In: International Workshop on Requirements Engineering for Product Lines (REPL 2002), Avaya Labs, Essen (2002)

[9] Gomaa, H., Shin, M.E.: A Multiple-View Meta-modeling Approach for Variability Management in Software Product Lines. In: Bosch, J., Krueger, C. (eds.) ICOIN 2004 and ICSR 2004. LNCS, vol. 3107, pp. 274–285. Springer, Heidelberg (2004)

[10] Gomaa, H.: Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Addison-Wesley, Upper Saddle River (2004)

[11] Halmans, G., Pohl, K.: Communicating the Variability of a Software-Product Family to Customers. Software and Systems Modeling 2, 15–36 (2003)

[12] Fowler, M.: UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley, Upper Saddle River (2004)

[13] Bosch, J., Florijn, G., Greefhorst, D., Kuusela, J., Obbink, J.H., Pohl, K.: Variability Issues in Software Product Lines. In: van der Linden, F.J. (ed.) PFE 2002. LNCS, vol. 2290, p. 13. Springer, Heidelberg (2002)