Support for Variability in Use Case Modeling with Refinement

Sofia Azevedo, Ricardo J. Machado
Universidade do Minho
Dep. de Sistemas de Informação
Guimarães, Portugal
+351 253 510 319
{sofia.azevedo,rmac}@dsi.uminho.pt

Alexandre Bragança
ISEP
Dep. de Eng. Informática
Porto, Portugal
+351 22 834 05 24
alex@dei.isep.ipp.pt

Hugo Ribeiro
Primavera BSS
Rua Cidade do Porto, 79
Braga, Portugal
+351 253 309 900
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. We also explore the functional refinement of use cases with *«extend»* relationships between them. This work allows understanding the activities of use case modeling with support for variability and of use case modeling with functional refinement when variability is present.

Keywords
Use case, software product line, variability, *«extend»*, alternative, option, specialization, refinement.

1. INTRODUCTION
Use case diagrams are one of the modeling artifacts 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 in the context of use cases and allows for the use case modeling activity to be applicable to the product line software development approach. That is possible by determining the locations in use case diagrams where variation will occur when instantiating the product line. This paper's contribution is on the formalization and understanding of the use case modeling activity with support for variability. We will illustrate our approach with the Fraunhofer IESE's GoPhone case study [1], which presents a series of use cases for a part of a mobile phone product line particularly concerning the interaction between the user and the mobile phone software. We propose an extension to the UML (Unified Modeling Language) metamodel [2] in order to formally provide for both the concrete and abstract syntaxes to represent different types of variability in use case diagrams. We consider use cases in different abstraction levels to elaborate on the (functional) refinement of use cases with *«extend»* relationships between them. In this paper we focus on the variability support as well as on the process point of view with regards to the use case modeling activity.

The paper is structured as follows. Section 2 elaborates on the differences between others' approaches and this paper's approach. Section 3 elaborates on the different types of variability we propose to be used in the context of use case modeling. Section 4 provides for the analysis of the UML *«extend»* relationship in contexts of variability and also for the extension we propose to the UML metamodel to support the different variability types. Section 5 analyzes the process of handling variability in use case diagrams in the context of the functional refinement of use cases. Section 6 illustrates our approach with the GoPhone case study. Finally Section 7 affords some concluding remarks.

MOMPES'10, September 20, 2010, Antwerp, Belgium.

2. RELATED WORK
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 [3].

This paper's work is inspired on the approach of Bragança and Machado to variability modeling in use case diagrams [4]. Bragança and Machado represent variation points explicitly in use case diagrams through extension points. Their approach consists of commenting *«extend»* relationships with the name of the products from the product line on which the extension point shall be present. Their approach to product line modeling is bottom-up (rather than top-down), which means that all the product line's products are known *a priori*. A top-down approach would consider that the product line would support as many products as possible within the given domain. In [5] 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. The approach in this paper adopts the top-down approach for product line modeling, therefore discarding the comments to the *«extend»* relationships.

In [5] 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. 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 [5].

Gomaa and Shin [6] 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 [6]. Besides alternative and optional use cases, Gomaa and Shin consider kernel use cases (use cases common to all product line members). Gomaa models in [7] 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 and with a stereotype in the use cases).

Halmans and Pohl propose in [8] use cases as the means to communicate variability relevant to the customer and they also propose extensions to use case diagrams to represent 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 using 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 also 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. Halmans and Pohl end up by introducing additional graphical elements to use case diagrams to represent variation points and variability cardinality explicitly in use case diagrams. We do not agree with this approach since it introduces more complexity to use case diagrams than modeling variability with stereotypes and use case relationships as well as it introduces a reasoning about variability that should be present in decision models (the selection of the variants to be present in the system and the system/product to which that selection applies according to the features).

Maßen and Lichter talk about three types of variability in [9]: 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).

According to Gomaa [7], and John and Muthig [5], use cases can be tagged with some stereotypes concerning variability. Table 1 shows the applicability of those stereotypes in our approach.

Table 1. Some use case stereotypes concerned with variability.

Stereotype	Applicability
«kernel»	Use cases in general
«alternative»	*«extend»* relationships
«optional»	Use cases in general
«variant»	Use cases in general

Some examples of approaches to functional decomposition of software systems are the 4SRS (Four Step Rule Set) method [10], KobrA or RSEB (Reuse-Driven Software Engineering Business) [11, 12]. However neither KobrA nor RSEB clearly contemplate a technique for refining use cases like the 4SRS method does.

Greenfield and Short [13] refer to refinement as the inverse of abstraction or the process of turning a description more complex by adding information to it. They refer to the process of developing software through refinement as progressive refinement. The process starts with requirements and ends up with the more concrete description of the software (the executable). They consider refinement as a concatenation of interrelated transformations mapping a problem to a solution. The goal of refinement is to smoothly decrease the abstraction levels that separate the problem from the solution. In general terms, Greenfield and Short talk about refinement as the stepwise decomposition of features' granularity. In the context of use cases, refinement is their detailing. However we defend that use cases can themselves be refined in order to facilitate the transformation of a problem (which can be modeled with use cases) to a solution (which shall be modeled with design artifacts e.g. logical architectures).

Gomaa [7] explored refinement in the context of feature modeling, where a feature can be a refinement of another. But in order to get to the features, use cases have to be modeled and mapped to features. Our approach eliminates this mapping activity. To Gomaa the refinement is expressed through *«extend»* relationships in the context of use cases. To us the refinement shall be expressed through the *«refine»* relationship we proposed in [14].

Cherfi, *et al*. [15] (in their work on quality-based use case modeling with refinement) describe the refinement process as the application of a set of decomposition and restructuring rules to the initial use case diagram. Their approach is iterative and incremental. It consists of decomposing the initial use case diagram into smaller and more cohesive ones to decrease the complexity of the diagram and increase its cohesion. In the approach of Cherfi, *et al*. to refinement, use cases are not actually detailed (like in ours), rather they are decomposed without detail being added to the description of those use cases.

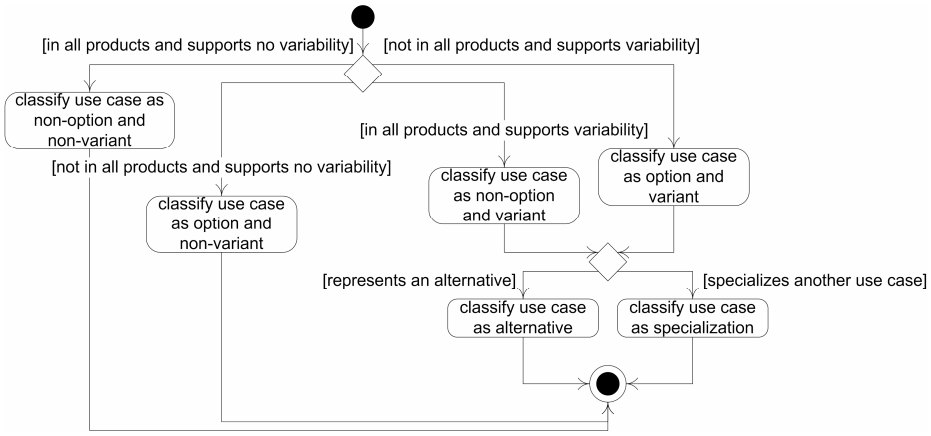

Figure 1. The use case variability types.

3. HANDLING VARIABILITY IN USE CASE MODELING

Figure 1 illustrates the variability types we consider and propose to be applicable in the context of use cases [16]. Use cases can be non-option or option. Non-option use cases are present in all product line members. Option use cases can be present in one product of the product line and not in another. It is not mandatory that option use cases are present in all products of the product line. Non-variant use cases are use cases that do not support variability. Variant use cases are use cases that support variability. This means that different products will support different alternatives for performing the same functionality or that different products will support different specializations of the same functionality. Later on during the modeling activity variant use cases are realized into alternatives or specializations respectively. Alternative use cases represent alternatives for performing the same system's use in mutually exclusive products or sets of products from the product line. Specialization use cases represent a special kind of alternatives. A specialization use case is a specialization of another use case. Specialization use cases that specialize the same use case represent alternatives for performing the same system's use in mutually exclusive products or sets of products from the product line. Option, alternative and specialization use cases are the representation of the three variability types that will be translated into stereotypes to be applicable to use cases. The use cases that do not represent options and are not variant (later alternatives or specializations) are non-option and non-variant, and shall not be marked with any stereotype. Non-option and option use cases are mutually exclusive as well as non-variant and variant use cases. Figure 1 represents the activity of classifying use cases with variability types: either *non-option and non-variant* or *option and non-variant* or *non-option and variant* or *option and variant*. These last two variability types can be realized into the *alternative* or the *specialization* variability types (as already explained). The activity of classifying use cases with the variability types is important for applying the corresponding stereotypes to the use cases (except for the *non-option and non-variant* variability type, which shall not be marked with any stereotype). The conditions of the decision nodes express the semantics of each one of the variability types. We would like to give emphasis to a particular variability type: the *option and variant* variability type. This variability type is applicable to a use case that is not present in all product line members but the different members in which it is present support different alternatives for performing that use case's functionality or different specializations of that use case's functionality. *Option and non-variant* use cases shall be marked as option use cases; *non-option and variant* as variant use cases; and *option and variant* use cases as both option and variant use cases.

4. THE *«extend»* RELATIONSHIP

The *«extend»* relationship allows modeling alternative and specialization use cases in use case diagrams.

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. As any other use case, an extending use case represents a given use of the system by a given actor or actors.

In the context of alternatives [16] both extending and extended use cases represent supplementary functionality since both represent alternatives, which are not essential for a product without variability to function. It shall be noted that alternatives are no longer supplementary when product line members are instantiated from the product line. Alternatives can be modeled with the generalization relationship in use case diagrams, but we recommend to model alternatives with the *«extend»* relationship in order to evidence their supplementary character according to the UML semantics.

If the intention is to use differential specification, specializations [16] 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 [16] 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).

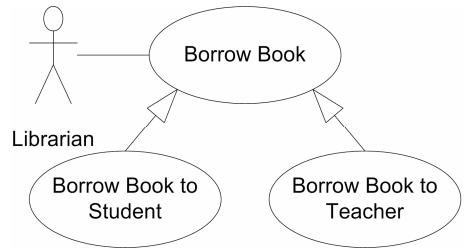

Figure 2. The specialization of the variant use case *Borrow Book* with a single actor.

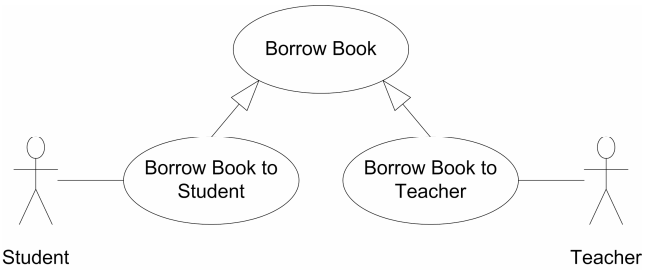

Figure 3. The specialization of the use case *Borrow Book* with two different actors.

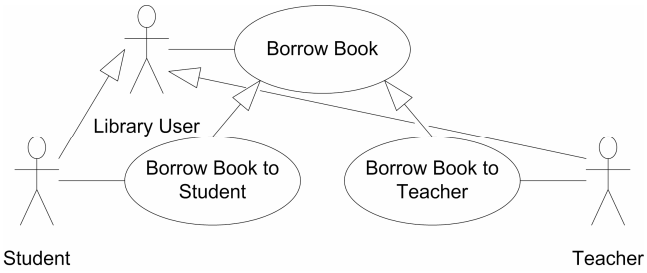

Figure 4. The specialization of the variant use case *Borrow Book* with two different actors.

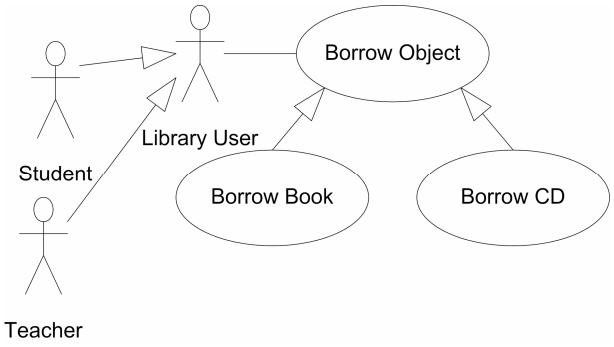

Figure 5. The specialization of the variant use case *Borrow Object*.

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).

In principle an extending use case is a use case that extends another use case both in the case of alternatives and in the case of specializations. In the case of specializations we consider that there is no multiple inheritance, therefore it is impossible for an extending use case to extend more than one use case. 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. The extended use case is not aware of the functionality described in the extending use case.

As previously mentioned if the intention is not to use differential specification, generalization relationships shall be used because specializations are complementary under those circumstances. However we may argue in a different way that the generalization relationship shall not be used to represent specializations in contexts of variability. Consider the examples depicted in figures 2 through 5 Figure 2. The example is an exception in terms of the (GoPhone) case study we will use further on in this paper. The figure shows that the use case *Borrow Book* can be specialized into *Borrow Book to Student* and *Borrow Book to Teacher*. If the actor is the same (the *Librarian*, who registers the borrowing), then the use cases that specialize the *Borrow Book* use case are alternatives to borrowing a book as both can be performed by the same actor. If the actor is not the same (the *Student* in the case of the *Borrow Book to Student* and the *Teacher* in the case of the *Borrow Book to Teacher*), then the use cases that specialize the *Borrow Book* use case are not alternatives to borrowing a book as both cannot be performed by the same actor (the same actor does not have an alternative way of borrowing a book). In this case in order for the generalization to be considered as variability, the actor of *Borrow Book* has to be the *Library User* (connected to *Borrow Book*) specialized into the *Student* (connected to *Borrow Book to Student*) and into the *Teacher* (connected to *Borrow Book to Teacher*). Another example: the use case *Borrow Object* can be specialized into *Borrow Book* and *Borrow CD*. In this case the actor can be the same for all of the use cases (the *Student* OR the *Teacher*). In order to support all the actors at the same time (the *Student* AND the *Teacher*), the *Library User* has to be specialized into them (the *Student* and the *Teacher*) and connected to the *Borrow Object* use case. This way the same actor (the *Library User*) can borrow an object (a *Book*) or alternatively another (a *CD*).

Figure 6 depicts the extension we propose to the UML metamodel concerning the *«extend»* relationship and use cases. We have added the stereotypes *«alternative»*, *«specialization»* and *«option»* to the standard UML stereotypes in order to distinguish the three variability types that were to be translated into stereotypes to be applicable to use cases. We have also added the stereotype *«variant»* to the standard UML stereotypes in order to mark use cases at higher levels of abstraction before they are realized into alternatives or specializations. We propose the stereotype *«option»* to be applicable to use cases that represent options. We also propose the stereotypes *«alternative»* and *«specialization»* to be applicable to the *«extend»* relationship for modeling alternatives and specializations respectively. Extending use cases involved in *«alternative»* relationships do not need to be marked with the stereotype *«alternative»* to evidence them as alternatives since they do not make sense without being involved

in that kind of relationships (an alternative use case is always alternative to another use case). The same happens with the stereotype *«specialization»* (a use case involved in a specialization relationship always specializes another use case). Regarding Figure 6 and the *Extend* metamodel element, as far as the unidirectional association is concerned, the end named *extendedCase* references the use case that is being extended (the extended use case) and the association means that many (zero or more) *«extend»* relationships refer to one extended use case. Regarding the aggregation, the end named *extend* references the *«extend»* relationships owned by the use case, and the end named *extension* references the use case that represents the extension (the extending use case) and owns the *«extend»* relationship. The metamodel means that one *«extend»* relationship is owned by one extending use case. Summarily a use case can be extended by many use cases and a use case can extend another use case. There can be zero or more alternatives (*«alternative»* relationships) to a use case. There can also be zero or more specializations (*«specialization»* relationships) for a use case. Although it can be argued that specializations are only worth the effort when there are two or more specialization use cases, we do not want to take freedom away from the modeler.

From now on we either use the *«extend»* relationship without stereotypes or with one of the two stereotypes applicable to this relationship from the proposed extension to the UML metamodel (depending on whether we are modeling alternatives or specializations).

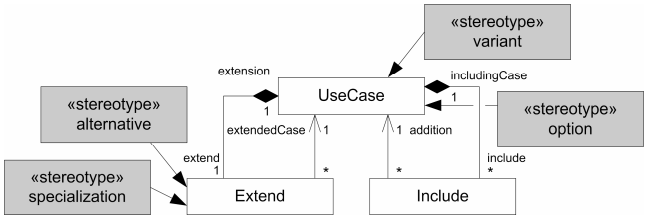

Figure 6. The proposed extension to the UML metamodel for modeling variability in use case diagrams.

It is important to distinguish alternatives from generalizations in contexts of variability. In the case of alternatives the extending use case is an alternative to the extended use case. In the case of specializations the extending use cases are alternatives to each other. Figure 7 shows the specialization of two alternative use cases from the GoPhone case study: *Insert Picture* and *Insert Picture or Draft Text*. It is possible to transform alternatives into specializations and the other way around. Again we are not restrictive on this since we do not want to take freedom away from the modeler.

5. HANDLING VARIABILITY IN USE CASE MODELING WITH REFINEMENT

Use cases can be decomposed with or without detailing their non-stepwise textual descriptions. Without detailing those descriptions we propose to represent the decomposition of use cases in use case diagrams with the *«include»* relationship. This decomposition suits the purpose of e.g. modeling later on an alternative to a part of the decomposed use case or modeling a part of the decomposed use case that is an optional part).

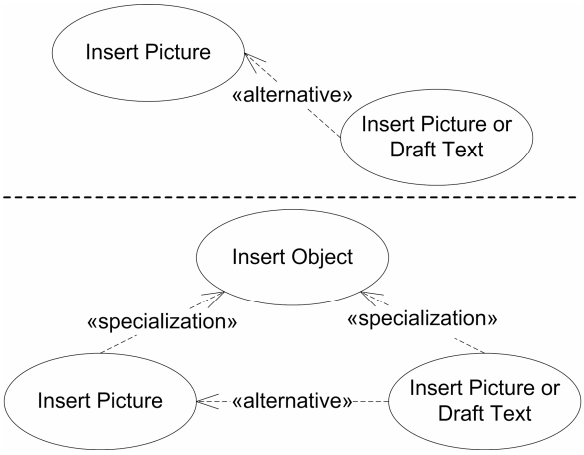

Figure 7. The specialization of *Insert Picture* and *Insert Picture or Draft Text*.

We consider that refining means decomposing and simultaneously detailing use cases. By refining use cases, the artifacts resulting from the refinement process (the refining use cases) are situated in lower abstraction levels comparatively to the refined use cases (the use cases that were submitted to the refinement process). In order to represent in the use case diagram this decrease in the abstraction level when refining use cases, we proposed in [14] to use the *«refine»* relationship (as a sort of traceability between use cases at different levels of detail).

In this section of the paper we depict in Figure 8 use cases according to the perspectives of detail*variability to illustrate in abstract terms our approach to use case modeling with support for variability. The detail perspective is intimately related to the activity of use case refinement. In this sense use cases can be more detailed if they are refined. The variability perspective is associated with the modeling of variability for product line support. The two perspectives (detail and variability) have been converted into axes of the illustrated space: y=detail and z=variability. Each level of the z axis corresponds to a (parallel) plan, which means that we position use cases in variability plans. Thus variability plans are plans that contain use cases representing variability in the three different types that have been translated into stereotypes to be applicable to use cases. The plan z=0 contains none of these use cases that represent variability.

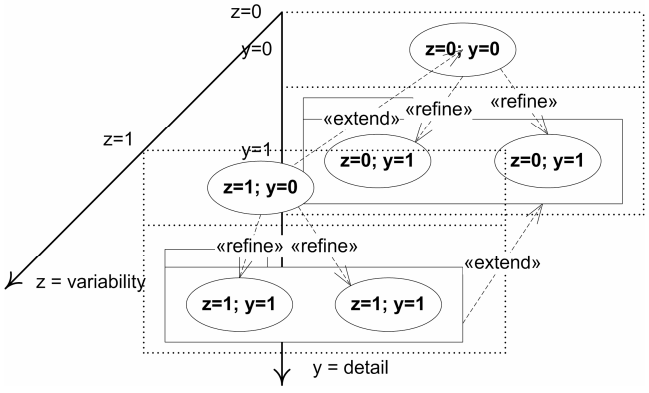

Figure 8. Use cases positioned according to the perspectives of detail*variability.

Use case name: Send Message
Use case description: The mobile user writes the message in a text editor. The GoPhone connects to the network to send the message. In order for the GoPhone to show an acknowledgement to the mobile user (stating that the message was successfully sent), it receives an acknowledgement from the network. Upon request from the GoPhone, the mobile user chooses to save the message into the sent messages folder.
Alternatives: The mobile user sends some different kinds of messages through the GoPhone.
The mobile user inserts objects into a message.
The mobile user attaches objects to a message.
The mobile user chooses the recipient's contact.
Specializations: -
Options: When writing the message, the mobile user activates letter combination (T9).

Use case name: Compose Message
Use case description: The mobile user writes the message in a text editor.
Alternatives: The mobile user sends some different kinds of messages through the GoPhone.
The mobile user inserts objects into a message.
The mobile user attaches objects to a message.
Specializations: -
Options: When writing the message, the mobile user activates letter combination (T9).

Use case name: Archive Message by Request
Use case description: Upon request from the GoPhone, the mobile user chooses to save the message into the sent messages folder.
Alternatives: The GoPhone automatically archives the message
Specializations: -
Options: -

Use case name: Automatically Archive Message
Use case description: The GoPhone saves the message into the sent messages folder and notifies the mobile user on the successful message saving into that folder.
Alternatives: -
Specializations: -
Options: -

Figure 9. Non-stepwise textual descriptions from the GoPhone use case *Send Message* and some of its related use cases.

The figure clarifies that the *«refine»* relationships imply increasing the detail level, whereas the *«extend»* relationships do not imply increasing the detail level but rather changing from one variability plan (z plan) to another. Extending use cases represent alternative or specialization use cases, therefore they must be situated at the same level of detail but in different variability plans (z plans). Variabilities do not imply adding detail to the non-stepwise textual descriptions of the use cases, like refinements do.

The figure shows the general case of the refinement of two use cases connected through an *«extend»* relationship. The refinement of a use case stereotyped as *«option»* is not relevant here, since it is not the case of an *«extend»* relationship connecting two use cases. The figure evidences that the refinement of two use cases connected through an *«extend»* relationship originates more detailed use cases organized in two packages that have also an *«extend»* relationship connecting them. That is not always the case. It is possible to have two use cases connected through a *«specialization»* relationship, which produces *«specialization»* relationships connecting more detailed individual use cases (and not packages) in different variability plans (an example of such case is in the next section of this paper).

6. THE VARIABILITY IN THE GoPhone CASE STUDY

The non-stepwise textual descriptions in Figure 9 were elaborated based on the functional requirements for the GoPhone. We rely on non-stepwise textual descriptions of use cases (the opposite of stepwise textual descriptions of use cases) to model variability in use case diagrams. Stepwise textual descriptions are structured textual descriptions in natural language that provide for a stepwise view of the use case as a sequence of steps, alert for the decisions that have to be made by the user and evidence the notion of use case actions temporarily dependent on each other. Stepwise descriptions shall be treated after modeling the use cases.

The *«include»* relationship involves two types of use cases: the including use case (the use case that includes other use cases) and the included use case (the use case that is included by other use cases). In the context of the *«include»* relationship the UML Superstructure states that the including use case depends on the addition of the included use cases to be complete. Nevertheless in our opinion the functionality of the included use cases shall be described in the including use case. Since we rely on non-stepwise textual descriptions of use cases to determine the *«include»* relationships, the including use case has to contain the description of the included use cases so that the modeler is able to define the parts that compose the including use case in order to decompose that use case (e.g. as can be seen from Figure 9 the functionality of the *Compose Message* use case is described in the *Send Message* use case).

In the context of the *«extend»* relationship the UML Superstructure states that an extending use case consists of one or more behavior fragment descriptions to be inserted into the appropriate spots of the extended use case. This means that the functionality of the extending use case is not described in the

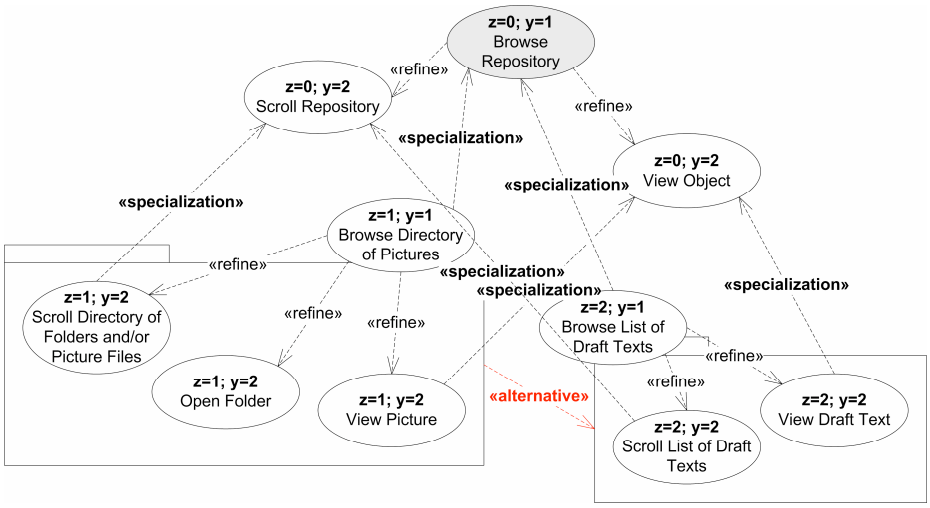

Figure 11. An example of refinement of the specialization type of variability from the GoPhone.

extended use case. The extended use case is not aware of the functionality described in the extending use case (e.g. as can be seen from Figure 9 the functionality of the *Automatically Archive Message* use case is not described in the *Archive Message by Request* use case). As Figure 10 depicts, the use case *Automatically Archive Message* is an alternative to the use case *Archive Message by Request* (they are connected through a kind of *«extend»* relationship, tagged with the stereotype *«alternative»* in order to evidence that the use case *Automatically Archive Message* is an alternative to the use case *Archive Message by Request*). It must be noticed that *Archive Message by Request* is an (included) use case included by the including use case *Send Message*, which means that the functionality of the use case *Archive Message by Request* is described in the *Send Message* use case. For this reason we could have extended the *Send Message* use case with the use case *Automatically Archive Message*, but then we would not be evidencing to which part of the functionality of the *Send Message* use case the use case *Automatically Archive Message* is an alternative to. Figure 10 also depicts that the *Browse Directory of Pictures* use case is a specialization of the use case *Browse Repository* (they are connected through another kind of *«extend»* relationship, tagged with the stereotype *«specialization»* in order to evidence that the use case *Browse Directory of Pictures* is a specialization of the use case *Browse Repository*).

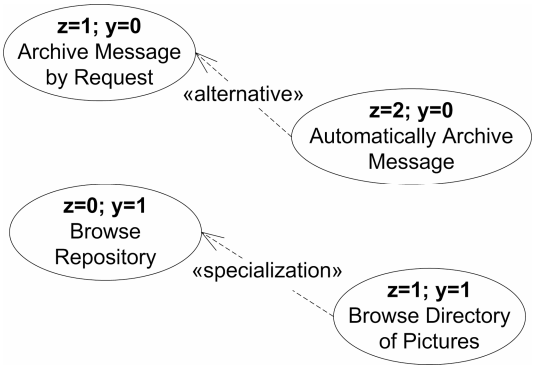

Figure 10. Some examples of variability modeled for the GoPhone use case *Send Message*.

6.1 Refinement of Specializations and Alternatives

Figure 11 shows the refinement of the specialization type of variability. The figure shows that both the use case that has been specialized (the *Browse Repository* use case) and the specialization use cases (the *Browse Directory* and *Browse List* use cases) were refined. Some use cases that refine the specialization use cases are specializations of the use cases that refine the use case that has been specialized (e.g. the *View Picture* use case is a specialization of the *View Object* use case). The use case *Open Folder* represents functionality that is not common to both specialization use cases since it is only applicable to one of the objects the specialization use cases refer to (the *Directory of Pictures*). Having in mind that specializations are a special kind of alternatives, specialization use cases are alternatives to each other. Figure 11 illustrates that the use cases that refine the specialization use cases are alternatives to each other as packages.

Figure 12 depicts that the use cases that refine two use cases connected through an *«alternative»* relationship are alternatives to each other as packages.

7. CONCLUSIONS

This paper has elaborated on the representation of variability in use case diagrams. It began by providing an in depth analysis of the state-of-the-art concerned with this topic. Based on our position towards the related work we proposed an extension to the UML metamodel to represent the three types of variability we have synthesized: alternatives, specializations and options. We concluded that alternatives and specializations shall be adequately modeled with the *«extend»* relationship, and that options shall be adequately modeled with a stereotype on use cases. This conclusion was based on the UML metamodel's semantics associated with the relationships for connecting use cases in use case diagrams: alternatives, specializations and options represent supplementary functionality. Although not being the core of this paper's contribute, we have also introduced the functional refinement of use cases connected through *«extend»* relationships due to its pertinence in large-scale product line contexts.

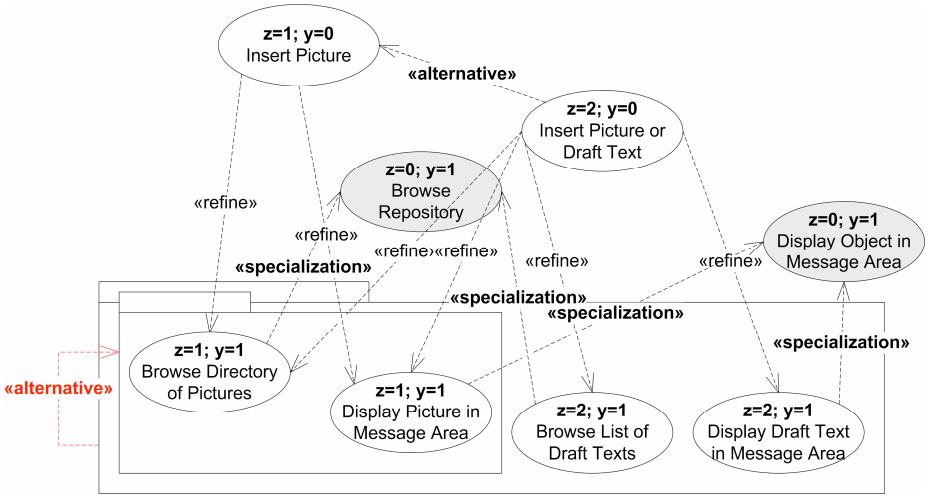

Figure 12. An example of refinement of alternative variability from the GoPhone.

8. REFERENCES

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

[2] OMG *Unified Modeling Language: Superstructure - version 2.2*. Object Management Group, 2009.

[3] Bragança, A. and Machado, R. J. Extending UML 2.0 Metamodel for Complementary Usages of the «extend» Relationship within Use Case Variability Specification. In *Proceedings of the SPLC 2006* (Baltimore, Maryland, USA, August 21-24, 2006). IEEE Computer Society, 2006.

[4] Bragança, A. and Machado, R. J. Deriving Software Product Line's Architectural Requirements from Use Cases: An Experimental Approach. In *Proceedings of the MOMPES 2005* (Rennes, France, June 6, 2005). TUCS General Publications, 2005.

[5] John, I. and Muthig, D. Product Line Modeling with Generic Use Cases. In *Proceedings of the Workshop on Techniques for Exploiting Commonality Through Variability Management* (San Diego, California, USA, August 19, 2002). Springer-Verlag, 2002.

[6] Gomaa, H. and Shin, M. E. Multiple-View Modelling and Meta-Modelling of Software Product Lines. *Institution of Engineering and Technology Software*, 2, 2 2008), 94-122.

[7] Gomaa, H. *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*. Addison-Wesley, Upper Saddle River, New Jersey, 2004.

[8] Halmans, G. and Pohl, K. Communicating the Variability of a Software-Product Family to Customers. *Software and Systems Modeling*, 2, 1 2003), 15-36.

[9] Maßen, T. v. d. and Lichter, H. Modeling Variability by UML Use Case Diagrams. In *Proceedings of the REPL 2002* (Essen, Germany, 2002). Avaya Labs, 2002.

[10] Machado, R. J., Fernandes, J. M., Monteiro, P. and Rodrigues, H. Transformation of UML Models for Service-Oriented Software Architectures. In *Proceedings of the ECBS 2005* (Greenbelt, Maryland, USA, April 4-7, 2005). IEEE Computer Society, 2005.

[11] Atkinson, C., Bayer, J. and Muthig, D. Component-Based Product Line Development: The KobrA Approach. In *Proceedings of the SPLC 2000* (Denver, Colorado, USA, August 28-31, 2000). Kluwer Academic Publishers, 2000.

[12] Jacobson, I., Griss, M. and Jonsson, P. *Software Reuse: Architecture, Process and Organization for Business Success*. Addison-Wesley, Upper Saddle River, New Jersey, 1997.

[13] Greenfield, J. and Short, K. *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. Wiley, Hoboken, New Jersey, 2004.

[14] Azevedo, S., Machado, R. J., Bragança, A. and Ribeiro, H. The UML «include» Relationship and the Functional Refinement of Use Cases. In *Proceedings of the SEAA 2010* (Lille, France, September 1-3, 2010). IEEE Computer Society, 2010 (accepted for publication).

[15] Cherfi, S. S.-s., Akoka, J. and Comyn-Wattiau, I. Use Case Modeling and Refinement: A Quality-Based Approach. In *Proceedings of the ER 2006* (Tucson, Arizona, USA, November 6-9, 2006). Springer-Verlag, 2006.

[16] Azevedo, S., Machado, R. J., Bragança, A. and Ribeiro, H. The UML «extend» Relationship as Support for Software Variability. In *Proceedings of the SPLC 2010* (Jeju Island, South Korea, September 13-17, 2010). Springer-Verlag, 2010 (accepted for publication).