Feature Model to Orthogonal Variability Model Transformation towards Interoperability between Tools

Fabricia Roos-Frantz[1], David Benavides[2] and Antonio Ruiz-Cortés[2]

[1] Unijuí, Departmento de Tecnologia - Ijuí, RS, Brasil
frfrantz@unijui.edu.br
[2] University of Seville, ETSI Informática. Avda. Reina Mercedes, s/n. Sevilla 41012, Spain
{benavides, aruiz}@us.es

Abstract. Documenting and managing the variability among products of a product line is an essential task in software product line engineering. Feature Model (FM) and Orthogonal Variability Model (OVM) are both modelling languages employed for this purpose. In this position paper we propose a transformation between FM and OVM models, thus allowing the interoperability between their modelling tools, particularly analysis tools. The current model transformation tools, particularly those that permit manipulating models from development environment such as Eclipse, and yet those with support to deal with metamodels, seem to be highly appropriate to carry out this task. Therefore we intend to use MOMENT2, which is a model transformation engine that provides support for graph transformation language, as a way of validating our proposal. To the best of our knowledge, there is no proposal for the automated transformation of FM into OVM, not to mention a solution based on model transformation.

Key words: Sofwtare Product Lines, Transformations, Feature Models, Orthogonal Variability Model

1 Introduction and Motivation

Software Product Line (SPL) Engineering paradigm [11,8] is one of the most recent ways of software reuse. Documenting and managing the variability among products of a product line is an essential task in this development paradigm. For this purpose, we can use variability models such as Feature Models (FM) and Orthogonal Variability Model (OVM), among others. The former is a common approach employed to represent an SPL by means of a hierarchical decomposition of features, which yields a feature tree or DAG (directed acyclic graph), comprising common and variable features of a system family. The latter is a more recent approach used to document variability in design and realisation artifacts [10,9]. Its main goal is to explicitly define and manage the variability of a SPL without take into account the common features.

In this position paper we propose a FM to OVM transformation in order to provide, in the near future, the interoperability among tools that support both variability modelling languages. In such transformation we transform the variable features of a FM into an OVM, thus providing an explicit view of variability of the software product line.

In 2006, a report by the Forrester consulting company [3] coined a term that has become one of the most relevant topics in the software engineering community: Application Lifecycle Management (ALM). ALM is defined as "*the coordination of development life–cycle activities, including requirements, modeling, development, build, and testing, through: 1) enforcement of processes that span these activities; 2) management of relationships between development artifacts used or produced by these activities; and 3) reporting on progress of the development effort as a whole*". In our current research scenario we aim at developing a reference architecture for ALM environments promoting process–quality standards compliance and integrating software engineering tools keeping traceability among artifacts. In this scenario, which focuses on ALM environment, it is necessary to have in place a FM to OVM transformation. For instance, it will be of practical importance being able to have interoperability between FM and OVM, proving a tool that can work with a variability model in different views. One view of all products in an SPL with their variabilities and commonalities represented with FMs and another view of an SPL, where only the variation points are considered.

2 Problem statement: Interoperability between FM and OVM tools

2.1 Multiple views

When a designer is defining and exploiting the variability in a software product line, he/she might be interested in having multiple views of this variability. By means of a feature model the designer can visualise all the features that a given product line can provide in its products. Common and variable features both are defined in such model. However, if he/she wants to manage and visualise only the variation points of such product line, it would be more visible if this information could be visualised explicitly. Hence, OVM would be highly suitable to show the variable features, since it expresses only the variation points. Fig. 1 illustrates how the variability of a product line represented by a FM would be represented by means of variation points in an OVM.

2.2 Automated analysis tools

The automated analysis of SPL can be considered as the computer-aided extraction of information from variability models that can be helpful for SPL engineers, designers, programmers and managers. For instance, one of them may want to know how many potential products are represented in a model, or even to know whether a specific product belongs to the model. Automated reasoning is an important task in the context of SPL, since it is practically impossible to do it manually, and on top of that it is error prone. In addition, the variability models are one of the main artifacts of the domain engineering activity and therefore their analysis in an early stage of development is essential to the success of the SPL.

In spite of the automated analysis is a challenge to be reached in SPL Engineering, only recently researchers have paid attention to the reasoning on these models, however their work has focused on FMs. Although the automated analysis of OVMs has

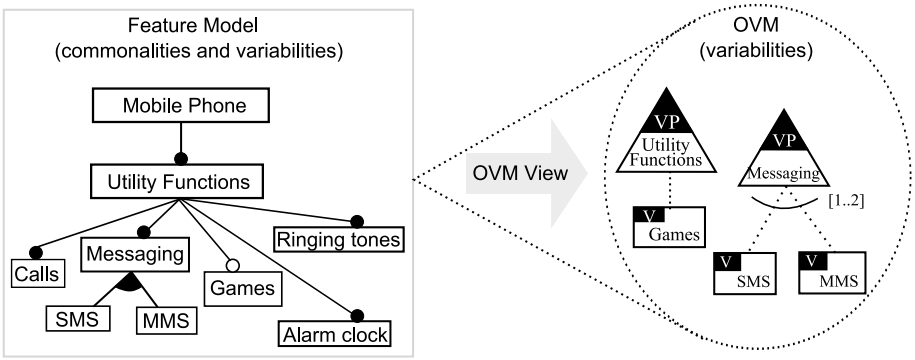


Fig. 1. Different views of variability in an SPL.

been proposed [2], it only deals with a small number of analysis operations, which are implemented using a specific logical representation and solver.

As there are already some automated analysis tools designed to work on FMs, we wonder if it is possible to reuse these tools, or part of those, to analyse OVM models. The automated analysis of OVM could be thought as a subset of the automated analysis of FMs so that the analysis of a specific OVM model can yield the same results as the analysis of the equivalent FM. Then, we believe that by means of a transformation between FM and OVM we do not need to design a new analysis tool for OVM models, but we could reuse the existing FM analysis tools.

3 First step towards a solution: FM to OVM transformation

Our first goal is to obtain the OVM equivalent from a given FM. As previously mentioned, the FMs document both the common and variable features of a SPL. The common features are those that form part of all products in a SPL, and the variable features are those that form part of some products, i.e. the variability. However, OVM only documents the variability of a SPL. Therefore, in order to transform a FM into an OVM, it is necessary that every variability represented in the FM is transformed into a variation point in the OVM.

This section describes a possible way to transform FM into OVM. We propose a FM2OVM algorithm in such a way that the variable parts of a FM is transformed into an OVM. Hence, the target model leaves aside the commonalities and gives an explicit view of the variability represented in the source FM. To our transformation we use the metamodel presented in [7] as the abstract syntax of FM (cf. Fig. 2).

As OVM metamodel we propose a metamodel based on the OVM abstract syntax defined by Metzger et al. in [2] (cf. Fig. 3). It was adapted in order to use the same concepts used in the FM metamodel aforementioned, e.g. *Set*, *Binary*, *Grouped* and *Solitary*. The element variation point has two types of relations, *Set* and *Binary*. The former is equivalent to the *alternative choice*, which has at least 2 *grouped variants* and

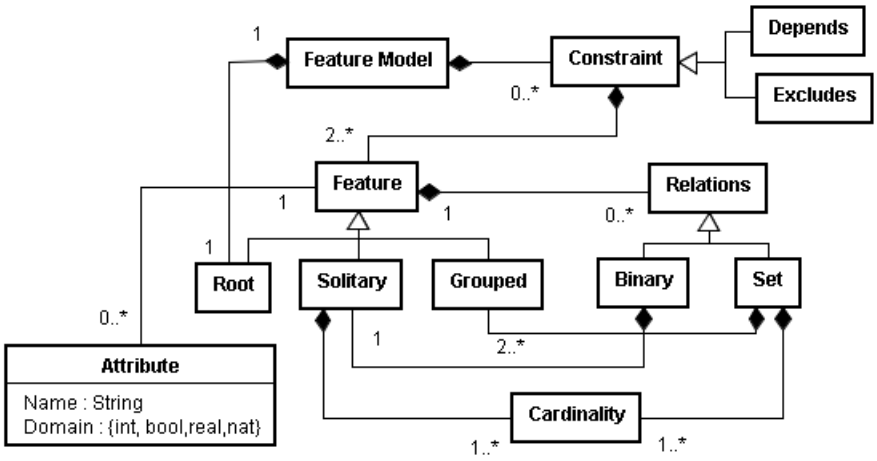


Fig. 2. FM metamodel [7]

a cardinality. The latter, is equivalent to *mandatory* or *optional*, which has one *solitary variant*. The OVM model may or may not have constraints. Some Object Constraint Language (OCL) statements were included to ensure some properties, as for instance the OCL at *context Set*. It ensures that the attribute *Card.max* is less than or equal to the number of grouped variant in the *set* relation and that *Card.min* is greater than or equals to 0, and *Card.min* is less than or equals to *Card.max*.

Our transformation algorithm has four preconditions: (1) the FM must be syntactically correct, it must satisfy the metamodel FM; (2) the FM must be valid, it represents at least one product [6,5,4]; (3) the FM does not have dead features, i.e. features that do not appear in any product [12,4]; and, (4) the FM has variability, at least one of its features is a variant feature, i.e. features that do not appear in all the products [4].

In addition to the preconditions, there are some postconditions. We consider that the translation is satisfactory if: (1) the target OVM model is syntactically correct according to the metamodel in Fig. 3; and, (2) the products of the source model without core features are the same as the ones of the target model without core features (those features that are common to all products).

The activity diagram of Fig. 4 illustrates how we propose to transform a FM into an OVM. We use the activity diagram as a tool to facilitate the understanding of the transformation algorithm, in fact it represents the main steps of our algorithm. In the following we describe the corresponding activities and the choices which determine the succession of them. The algorithm traverses the tree in preorder and performs the following operations recursively at each node, starting with the root node:

"Select Feature n in FM:" the algorithm visits the node; *"n in core features or n is the root?"* concerns the distinction between features that are common to all products (core features) and those that are not (variabilities). When n is a CF or root it does not become an element in OVM; *"Transform parent(n) in VP:"* when n is not a CF, its

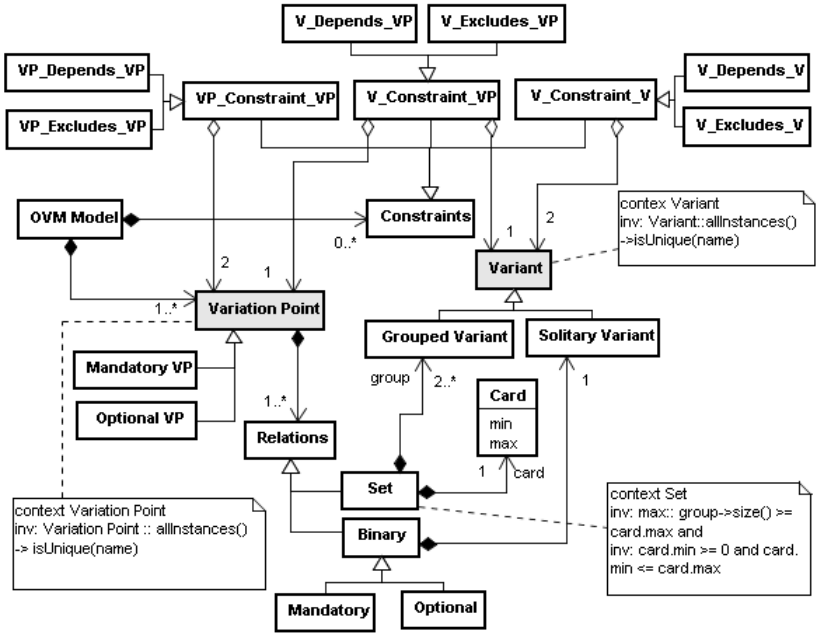


Fig. 3. OVM metamodel based on [2].

parent will be transformed in a VP; *"parent(n) in core feature?"* when transforming the parent(n) in VP, the type of such VP depends of it is a variability or a commonality; *"Set VP type as OPTIONAL VP:"* when parent(n) is a variability, then VP is Setted as OPTIONAL VP; *"Set VP type as MANDATORY VP:"* when parent(n) is a commonality, then VP is setted as MANDATORY VP; *"n is solitary feature?"* the type of n determines the type of variant when transforming n into variant; *"Transform n in GROUPED VARIANT:"* the grouped feature n in FM will be transformed in grouped variant; *"Set relationship between VP and V as SET:"* when a variant is a grouped variant, the relationship with its parent VP is SET; *"Set cardinality:"* when the relationship is SET, it has a cardinality; *"Transform n in SOLITARY VARIANT:"* the solitary feature n in FM will be transformed into solitary variant; *"n with cardinality [1..1]?"* the cardinality of a solitary feature determines if the variant will be optional or mandatory; *"Set relationship between VP and V as OPTIONAL:"* when solitary feature with cardinality [0..1] is transformed into variant, the relationship with its parent VP is OPTIONAL; *"Set relationship between VP and V as MANDATORY:"* when solitary feature with cardinality [1..1] is transformed into variant, the relationship with its parent VP is MANDATORY.

Now we can use the examples in Fig. 5 and 6 to illustrate the transformation. Taking into account that the FM fulfils the preconditions, we are able to translate a FM into an OVM. Then, by applying the algorithm transformation (cf. Fig. 4) from the FM in Fig. 5 we obtain the OVM in Fig. 6. The resulting OVM is syntactically correct and if we omit

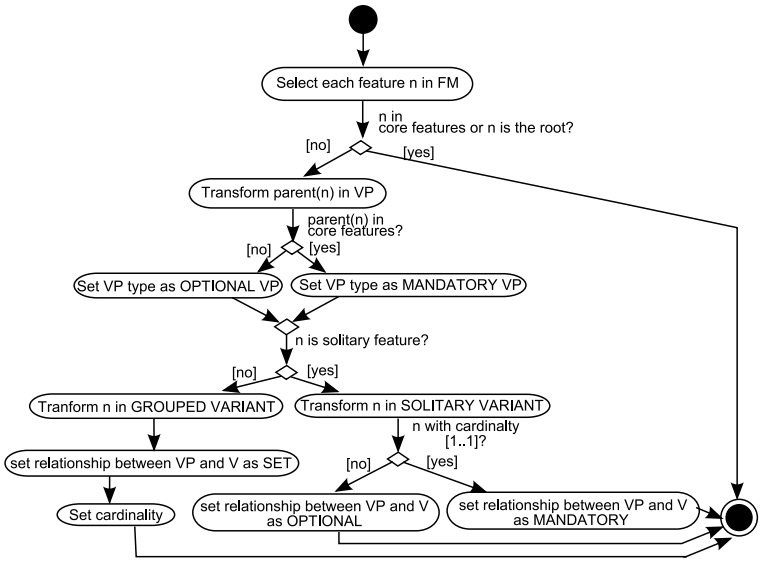
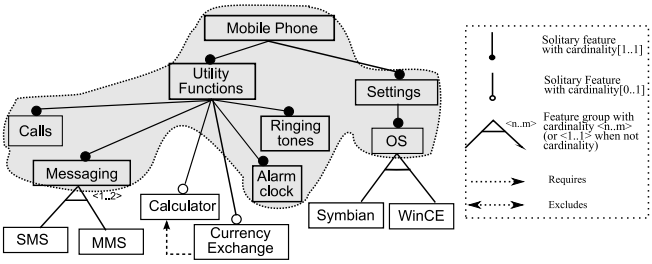


Fig. 4. FM2OVM algorithm.

all the CF of the FM set products and also of the OVM set products, we have two equivalent set of products, namely:



Core Features (CF) = {Mobile Phone, Utility Functions, Calls, Messaging, Alarm clock, Ringing tones, Settings, OS}

Fig. 5. An example of a mobile phone product line using FM.

P1= {SMS,Symbian}
P2= {SMS,WinCE}
P3= {MMS,Symbian}
P4= {MMS,WinCE}
P5= {SMS,MMS,Symbian}
P6= {SMS,MMS,WinCE}
P10= {MMS,WinCE,Calculator}
P11= {SMS,MMS,Symbian,Calculator}
P12= {SMS,MMS,WinCE,Calculator}
P13= {SMS,Symbian,Calculator,Currency Exchange}
P14= {SMS,WinCE,Calculator,Currency Exchange}
P15= {MMS,Symbian,Calculator,Currency Exchange}

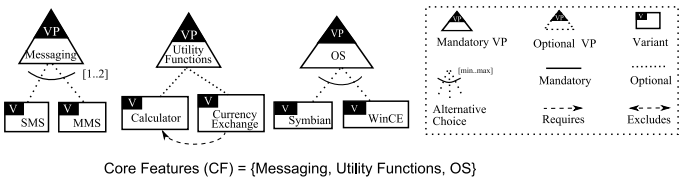


Fig. 6. OVM example: mobile phone product line.

P7= {SMS,Symbian,Calculator} P16= {MMS,WinCE,Calculator,Currency Exchange}
P8= {SMS,WinCE,Calculator} P17= {SMS,MMS,Symbian,Calculator,Currency Exchange}
P9= {MMS,Symbian,Calculator} P18= {SMS,MMS,WinCE,Calculator,Currency Exchange}

It therefore follows that given a FM, which represent a PL (a set of products with commonalities and variabilities), the translation FM2OVM results an OVM which represent the same PL as the source FM, but representing only the PL variability.

4 Discussions and Open Issues

We propose a transformation between FM and OVM models, thus allowing the interoperability between their modelling tools, particularly analysis tools. The current model transformation tools, particularly those that permit manipulating models from development environment such as Eclipse, and yet those with support to deal with metamodels, seem to be highly appropriate to carry out this task. We intend to use the MOMENT2 tooling [1], which provides a model transformation engine that provides support for graph transformation language. MOMENT2 is an algebraic model management framework that permits manipulating models in the Eclipse Modeling Framework (EMF). To the best of our knowledge, there is no proposal for the automated transformation of FM into OVM, and not to mention a solution based on model transformation.

When defining our algorithm we have found that the main issue to fulfil the post-conditions concerns the structural difference between FM and OVM diagrams. The translation of a tree into OVM is not a straight task due to it usually has more than two levels. Regardless the technique used to do this transformation, this problem comes forward. We need to work more in it to achieve a satisfactory transformation.

Bidirectional transformation. Although most of the model-to-model transformation tools do not provide support to bidirectional transformation, we intend to achieve the transformation between FM and OVM in both directions. This is essential in our research work on automated analysis of OVM. We intend to use FAMA Framework [13] to analyse OVM, which is a tool for the automated analysis of FMs. Its main goal is to provide an extensible framework where current research on variability model automated analysis might be developed and easily integrated into a final product. FAMA receives as input a model conforms to a FM metamodel and performs several analysis operations on this FM by using different solvers. We aim at

using model-to-model transformation in order to generate a target model conforms to a FM metamodel from a source model conforms to an OVM metamodel, and thus to be able to analyse this resulting FM by using FAMA. Such analysis could be thought as a subset of the automated analysis of FMs so that the analysis of a specific OVM model can yield the same results as the analysis of the equivalent FM.

Deal with commonalities. Another issue concerns how to deal with the commonalities that are not transformed. If we are interested in maintaining the traceability between both diagrams we need to preserve the common features in some way.

5 Conclusions and Future Work

In our scenario of research it is useful to have the transformation FM2OVM, and up to now to the best of our knowledge, there is no work providing such solution. Such transformation would allow us provide a tool support to work with the interoperability between both languages. In this paper we proposed a FM to OVM transformation, in which we transform the variable parts of a FM into variation points and variants in an OVM. In this way, we obtain a view of the variability of a FM by means of an OVM allowing the variability tools work with multiples views of the variability.

To achieve an automated FM2OVM engine by means of model-to-model transformation techniques, we first need better define the OVM metamodel, and then to define the transformation rules between the elements of both metamodels.

As future work we intend to develop a study case by using MOMENT2 engine to transform FM into OVM. In addition, we aim at developing a bidirectional transformation due to it is essential to achieve a more complete solution for our problem.

Acknowledgement. This work was partially supported by Spanish Government under CICYT project Web-Factories (TIN2006-00472) and project SETI (TIN2009-07366) and by the Andalusian Government under project ISABEL (TIC-2533) and Evangelischer Entwicklungsdienst e.V. (EED).

References

1. Boronat A., Heckel R., Carsí J. A., Gómez A., Ramos I., and Meseguer J. Moment2: A formal framework for model management. http://www.cs.le.ac.uk/people/aboronat/tools/moment2/.
2. Metzger A., Pohl K., Heymans P., Schobbens P., and Saval G. Disambiguating the documentation of variability in software product lines: A separation of concerns, formalization and automated analysis. In *15th IEEE Int'l Requirements Engineering Conference*, pages 243–253, 2007.
3. Schwaber C. The changing face of application life-cycle management. Forrester Research, 2006.
4. Benavides D. *On the automated analysis of software product lines using feature models*. PhD thesis, University of Sevilla, 2007.

5. Benavides D., Ruiz-Cortés A., Trinidad P., and Segura S. A survey on the automated analyses of feature models. In *JISBD*, pages 367–376, 2006.
6. Benavides D., Trinidad P., and Ruiz-Cortés A. Automated reasoning on feature models. In *Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005*, volume 3520 of *LNCS*, pages 491–503. Springer–Verlag, 2005.
7. Benavides D., Trujillo S., and Trinidad P. On the modularization of feature models. In *In First European Workshop on Model Transformation*, 2005.
8. Pohl K., Böckle G., and F. J. van der Linden. *Software Product Line Engineering: Fundations, Principles and Techniques*. Springer–Verlag, Berlin, DE, 2005.
9. Bencomo N. *Supporting the Modelling and Generation of Reflective Middleware Families and Applications using Dynamic Variability*. PhD thesis, Lancaster University, 2008.
10. Loughran N., Sánchez P., Garcia A., and Fuentes L. Language support for managing variability in architectural models. In *Software Composition*, pages 36–51, 2008.
11. Clements P and Northrop L. *Software Product Lines: Practices and Patterns.* SEI Series en Software Engineering. Addison–Wesley, August 2001.
12. Trinidad P., Benavides D., Durán A., Ruiz-Cortés A., and Toro M. Automated error analysis for the agilization of feature modeling. *J. of Systems and Software*, 81(6):883–896, 2008.
13. Trinidad P., Benavides D., Ruiz-Cortés A., Segura S., and Jimenez A. Fama framework. In *Software Product Line Conference Tool Demonstrations (in press)*, 2008.