A Preliminary Comparison of Formal Properties on Orthogonal Variability Model and Feature Models

Fabricia Roos-Frantz *
Universidade Regional do Noroeste
do Estado do Rio grande do Sul (UNIJUÍ)
São Francisco, 501.
Ijuí 98700-000 RS (Brazil)
frfrantz@unijui.edu.br

Abstract

Nowadays, Feature Models (FMs) are one of the most employed modelling language by managing variability in Software Product Lines (SPLs). Another proposed language also in order to managing variability in SPLs is the Orthogonal Variability Model (OVM). Currently, the differences between both languages, FMs and OVM, are not so clear. By considering that a formal language should have a well defined syntax and semantics, some authors had defined syntax and semantics of FMs explicitly. However, in in the definition of OVM, its semantic domain and semantic function are not well discussed. Without this clear definition, we could have a misinterpretation when using OVM diagrams. Our aim in this paper is to clarify and better explore the abstract syntax, the semantic domain and the semantic function of OVM, and to emphasize the differences between FMs and OVM concerning such aspects.

1. Introduction and Motivation

Documenting and managing variability is one of the two key properties characterising Software Product Line Engineering (SPLE) [7]. Over the past few years, several variability modeling techniques have been developed aiming to support variability management [10]. In this paper we take into account two modelling languages: FMs, that are one of the most popular, and OVM. We want to discuss about the differences between both languages.

FM was proposed for the first time in 1990 and currently it is the mostly used language to model the variability in SPL. This model capture features commonalities and variabilities, represents dependencies between features, and determines combinations of features that are allowed and forbidden in the SPL [4].

OVM is a variability model proposed by Klaus Pohl et al. [7] for managing the variability in the applications in terms of requirements, architecture, components and test artifacts. In an OVM only the variability of the product line is documented. In this model a *variation point (VP)* documents a variable item, i.e a system functionality which can vary and a *variant (V)* documents the possible instances of a variable item. Its main purpose are: (1) to capture the VPs, i.e. those items that vary in an SPL, and (2) to represent Vs, i.e. how the variable items can vary and (3) to determine constraints between Vs, between Vs and VPs and between VPs and VPs.

A fundamental concern, when we want to do a reasoning about a language, is to make it a formal language [4]. In the words of Schobbens et al. [9], "formal semantics is the best way to avoid ambiguities and to start building safe automated reasoning tools for a variety of purposes including verification, transformation, and code generation". According to Harel and Rumpe [3], a language is formal when it has a well defined syntax (the notation) and a well defined semantics (the meaning).

Nowadays we have a well defined syntax and semantics to FM languages [9], i.e. we can construct FMs without misinterpretation, because we know what is a correct model and what it means exactly. However, if we are working with OVM we are not sure about the correct meaning of these models and also about the real differences between FMs and OVM. This paper focus on doing a review about OVM's syntax and semantics, which were proposed in the literature, and discuss about the differences between FMs and OVM in order to avoid misunderstanding.

The remainder is organized as follows: Section 2 discusses about the abstract syntax of OVM and compares some of its properties with FMs; Section 3 we comment

*PhD student at the University of Sevilla

about the OVM's semantic domain and FM's semantic domain, and we suggest another semantic domain to OVM; Section 4 we discuss about the OVM's semantic function and FM's semantic function; Section 5 presents our conclusions.

1.1. Feature Models (FMs)

The first feature model was proposed in 1990 [5] as part of the method Feature-Oriented Domain Analysis (FODA). Since then, several extensions of FODA have been proposed. A FM represents graphically a product line by means of combinations of features. A FM is composed of two main elements: features and relationships between them. Features are structured in a tree where one of these features is the root. A common graphical notations is depicted in Figure 1.

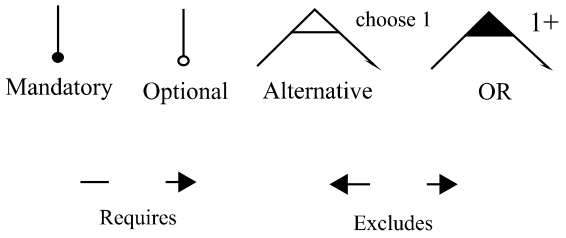

Figure 1: Graphical notation for FM

Figure 2 is an example of feature model inspired by the mobile phone industry. It defines a product line where each product contains two features: *Call* and *Connectivity*. Where *Call* is a mandatory feature and *Connectivity* is an optional feature. It means that all application that belongs to this SPL must have the feature *Call* and can have the feature *Connectivity*. Each product must have at least one of the two types of call, *voice* or *data*, because of the relationship OR. If the product has the feature *Connectivity*, then it must have at least one of the two features, *USB* or *Wifi*.

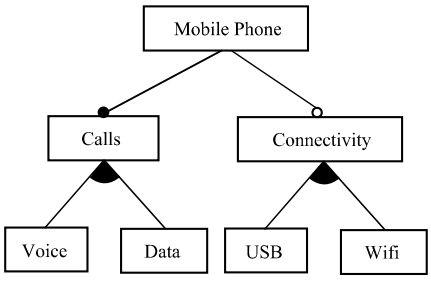

Figure 2: Example of FM

1.2. Orthogonal Variability Model (OVM)

OVM is a proposal for documenting software product line variability [7]. In an OVM only the variability of the product line is documented. In this model a *variation point (VP)* documents a variable item and a *variant (V)* documents the possible instances of a variable item. All VPs are related to at least one V and each V is related to one VP. Both VPs and Vs can be either optional or mandatory (see Figure 3). A mandatory VP must always be bound, i.e, all the product of the product line must have this VP and its Vs must always be chosen. An optional VP does not have to be bound, it may be chosen to a specific product. Always that a VP, mandatory or optional, is bound, its mandatory Vs must be chosen and its optional Vs can, but do not have to be chosen. In OVM, optional variants may be grouped in *alternative choices*. This group is associated to a cardinality *[min...max]* (see Figure 3). Cardinality determines how many Vs may be chosen in an alternative choice, at least *min* and at most *max* Vs of the group. Figure 3 depicts the graphical notation for OVMs [7, 6].

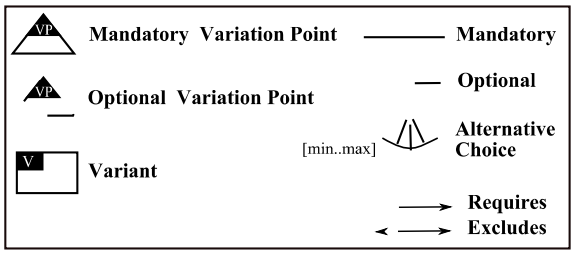

Figure 3: Graphical notation for OVM

In OVM, constraints between nodes are defined graphically. A constrain may be defined between Vs, VPs and Vs and VPs and may be an *excludes* constraint or a *requires* constraint. The excludes constraint specifies a mutual exclusion, for instance, a variant *excludes* a optional VP means that if the variant is chosen to a specific product the VP must not be bound, and vice versa. A *requires* constraint specifies an implication, for instance, a variant *requires* a optional VP means that always the variant is part of a product, the optional VP must be also part of that product. Figure 4 depicts a example of an OVM inspired by the mobile phone industry.

2. Syntax: abstract and concrete syntax

In graphical languages, such as FMs and OVM, the physical representation of the data is known as concrete syntax. In other words, what the user see, like arrows and squares, is only the concrete syntax. Defining rigid syntactics rules

	Nodes	Hierarchical structure	Multiple inheritance	Variation Points	Complex constraints
FM (Batory [1])	Features	yes	no	not explicit	yes
OVM-KP (Klaus Pohl et al. [7])	VPs and Vs	no	yes	Mandatory	no
OVM-M (Metzger et al. [6])	VPs and Vs	no	no	Mandatory / Optional	no

Table 1: Summary of abstract syntax properties.

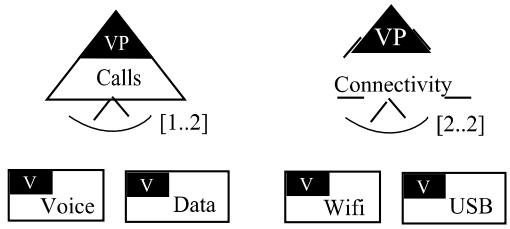

Figure 4: OVM example: mobile phone product line

in visual languages is a difficult task, for this reason, a common practice is to define a formal semantics of the language based on its abstract syntax. The abstract syntax is a representation of data that is independent of its physical representation and of the machine-internal structures and encodings [4].

The first OVM's abstract syntax was proposed by Klaus Pohl et al. [7]. The authors proposed a metamodel, which describes what is a well-formed diagram. Later, Metzger et al. [6] proposed an OVM's abstract syntax which use a mathematical notation to describe a well-formed diagram. In this section we will compare both abstract syntax, underlining the main differences between them. At the same time, we will compare the properties of OVM language with FMs languages according to the abstract syntax. In order to compare both languages we will use the FM proposed by Batory [1], and we will refer to FMs as the proposed in [1].

Table 1 compares some properties about different abstract syntaxes. Each row of this table represents an abstract syntax proposed in the literature. The first one is FM-Batory, which was proposed in [1]. The second is OVM-KP, which was proposed in [7] and the third is OVM-M, proposed in [6]. Each column represents a property of the language. Bellow we describe and comment each one of these properties.

- *Nodes*. We use the term "nodes" to say what a node represents in a graph. For example, in a FM the nodes of the graph are features. It means that each node represents an increment in system functionality. On the other hand, either in OVM-KP or OVM-M, the nodes are VPs (variation points) and Vs (variants), i.e. each functionality of the system that vary is represented by a VP, and each V represents how the VP can vary.

- *Hierarchical structure*. This property states if a graph has a hierarchical structure or not. The FM is represented by a tree and there is one node that is a root. Each node of the tree, with the exception of the root, have one parent. On the other hand, OVM diagrams do not have a hierarchical structure. This diagrams are composed for variation points, which always have child variants. In this diagram there is no a root node, the diagram is composed of a set of VPs with its possible variants.

- *Multiple inheritance*. Happens when a well formed diagram allows a node to have two different parents. The FM does not allow a feature to have more than one parent. When dealing with OVM, this property is defined in two different ways. In Pohl's abstract syntax the diagram can have variants with different VP parents; however, in the second proposal, a variant can have only one VP parent.

- *Variation Points*. Here we consider if graph nodes represent variation points explicitly. In FM all nodes are features, there is no explicit way to represent variation points. The way that FMs represent the variation points, identified in requirements, is through optional or alternative feature. On the other hand, in OVM, all variable item in an SPL is represented by a specific node called VP. In OVM-KP a VP only can be mandatory, i.e. all products of an SPL share this VP. However, in OVM-M, a VP can be mandatory or optional, i.e. if it is mandatory it will be in all products of the SPL; otherwise if it is optional, it will be only in those products which such VP was bound.

- *Complex constraints*. In both languages, FM and OVM, in addition to diagrams there are constraints that restrict the possible combinations of nodes. In FM we can specify constraints more complex than only excludes and requires. For example, we can write constraints like: (*F requires A or B or C*), this means F needs features A, B, or C or any combination thereof.

In OVM, only constraints of type excludes and requires can be specified.

3. Semantics: semantic domain

According to Harel and Rumpe a language's semantics "must provide the meaning of each expression and that meaning must be an element in some well defined and well-understood domain" [3]. In this definition we have two important information about the definition of semantics. First of all the semantics must give a meaning to each element defined in the language's syntax. Second, to define such meaning we need a well defined and well-understood domain. This domain, called "semantic domain" is an abstraction of reality, in such a way that determine what the language should express.

When we were reviewing the literature we realized that the OVM models are treated like FMs, namely, they represent the same domain. But, what means to represent the same domain? Have they the same semantic domain? According to Batory [1] a FM represents a set of products, and each product is seen as a combination of features. Then, a FM represents all the products of the SPL. By considering this, the semantic domain of a FM is considered a product line [9], i.e. the set of sets of combinations of features $\mathcal{PP}(f)$.

The semantic domain of OVM is also considered a product line [6]. Hence, product line is a set of products and each product is a combination of VPs and Vs, then the semantic domain of OVM is a set of sets of combinations of VPs and Vs, i.e. the $\mathcal{PP}(VP \cup Vs)$.

Until now, the semantic domain of OVM has been considered like in FM, the set of sets of combinations of nodes. But, if in OVM we were interested only in variations, we can consider that the semantic domain of OVM is a set of sets of combinations of only variants. Then, the semantic domain of OVM is a product line and each product is a set of variants, i.e. the set of sets of combinations of variants $\mathcal{PP}(V)$. In this way we consider that each product of the product line has only variants and not variation points. For example, for the model of the Figure 4 the semantic domain is the $\mathcal{PP}(V)$, where V is the set {Voice, Data, Wifi, USB}.

4. Semantics: semantic function

The definition of semantics, proposed by Harel and Rumpe, stated that it must provide a meaning of each expression and that meaning must be an element in some domain. The domain is the semantics domain ($\mathcal{S}$) and the expressions are represented by the syntactic domain ($\mathcal{L}$). According to Heymans et al. [4], the set of all data that comply with a given abstract syntax is called the syntactic domain. The function that relates ($\mathcal{L}$) and ($\mathcal{S}$) by giving a meaning to each expression is called semantic function ($\mathcal{M}$). Then, $\mathcal{M} : \mathcal{L} \rightarrow \mathcal{S}$. To every diagram of $\mathcal{L}$, the function $\mathcal{M}$ assigns a product line in $\mathcal{S}$.

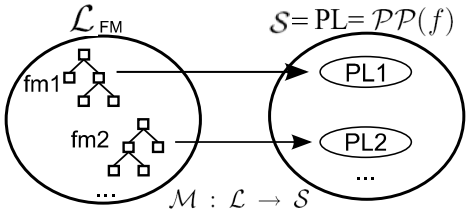

Figure 5: Semantic function of FM

Figure 5 gives an illustration of the FM's semantic function. In this figure we have two different FMs that comply with a FM's abstract syntax, and we have a semantic function that assigns to each diagram a different product line in the semantic domain $\mathcal{M}_F : \mathcal{L}_F \rightarrow PL$ where $PL = \mathcal{PP}(f)$, i.e the power set of set of features. For example, if we have those two diagrams of the Figure 6 (a) and (b), and we apply this semantic function, we will have respectively the product line M_F(fm1) and M_F(fm2).

M_F(fm1)= {{*f1,f2,f3,f4,f7*}, {*f1,f2,f3,f4,f7,f6*}, {*f1,f2,f3,f5,f7*}, {*f1,f2,f3,f5,f7,f6*}}

M_F(fm2)= { {*f1,f2,f4*}, {*f1,f2,f5*}, {*f1,f2,f4,f3,f6*}, {*f1,f2,f5,f3,f6*}, }

Figure 7 depicts an illustration of the OVM's semantic function proposed by Metzger et al. [6]. In this figure, each different OVM diagram that comply with the OVM's abstract syntax are assigned by the semantic function to each product line in the semantic domain. The semantic function is $\mathcal{M}_{OVM-M} : \mathcal{L}_{OVM-M} \rightarrow PL$, where $PL = \mathcal{PP}(VP \cup V)$. In this case, Metzger et al. define that in OVM a product line is defined like a combination of VPs and Vs. For example, if we have the two diagrams of the Figure 8 (a) and (b), and we apply the semantic function, we will have respectively the product line M_{OVM-M}(ovm1)

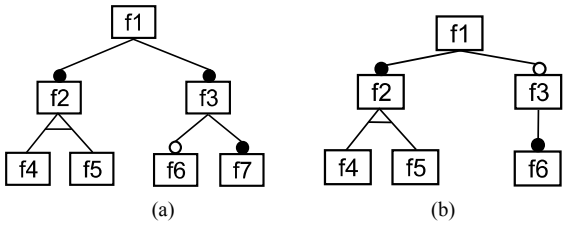

Figure 6: Concrete syntax of fm1 (a) and fm2 (b)

and M_{OVM-M}(ovm2).

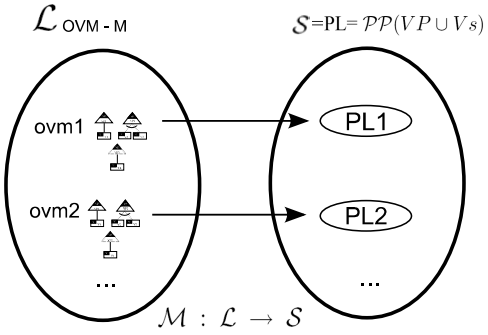

Figure 7: Semantic function of OVM-M

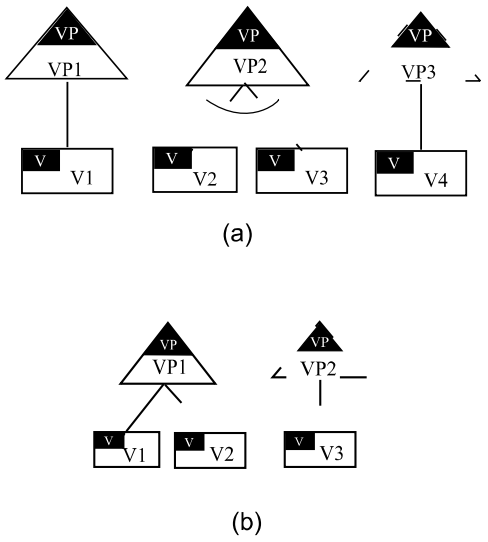

Figure 8: Concrete syntax of ovm1 (a) and ovm2 (b)

M_{OVM-M}(ovm1)= $\{\{VP1,V1,VP2,V2\}, \{VP1,V1,VP2,V2,VP3,V4\}, \{VP1,V1,VP2,V3\}, \{VP1,V1,VP2,V3,VP3,V4\}\}$

M_{OVM-M}(ovm2)= $\{\{\{VP1,V1\}, \{VP1,V1,V2\}, \{VP1,V1,V2,VP2\}, \{VP1,V1,V2,VP2,V3\}, \{VP1,V1,VP2\}, \{VP1,V1,VP2,V3\}\}$

If we consider that a semantic domain of OVM is $\mathcal{PP}(V)$, we have another semantic function. But, as we already have the semantic function to the semantic domain $\mathcal{PP}(VP \cup V)$, we can achieve the semantic domain $\mathcal{PP}(V)$ excluding all VPs of the products. For example, the product line M_{OVM-M}(d2) would be

M_{OVM-M}(d2)= $\{\{\boldsymbol{V1}\}, \{\boldsymbol{V1,V2}\}, \{V1,V2,V3\}, \{V1,V3\}\}$

We can notice that with this semantic domain ($\mathcal{PP}(V)$) we have 4 products instead of 6, because two of them are duplicated $\{V1\}$ and $\{V1,V2\}$. This happens because of the Optional VP2. When we consider that the VPs are part of the products, and in the model we have a optional VP with an optional child, we will have two products that are the same when implemented. For example, the products $\{VP1,V1\}$ and $\{VP1,V1,VP2\}$. In fact the functionality that will be implemented will be V1, both products are the same.

To discuss about the difference between both OVM's semantic domain, we will use as an example the equivalence problem discussed in [8]. The equivalent models operation checks whether two models are equivalent. Two models are equivalent if they represent the same set of products [2]. According to OVM-M, if we observe the example depicted in the Figure 9, we can say that both models are equivalent, because they represent the same set of products. In the product of the OVM_1, *Media* is a *variation point* and in OVM_2, *Media* is a *variant*. In this example we have considered that the semantic domain of OVM was $\mathcal{PP}(VP \cup V)$, then the models seem to be equivalents because they represent the same set of products: $\{$*Media, MP3, MP4*$\}$ = $\{$MP3, Media, MP4$\}$.

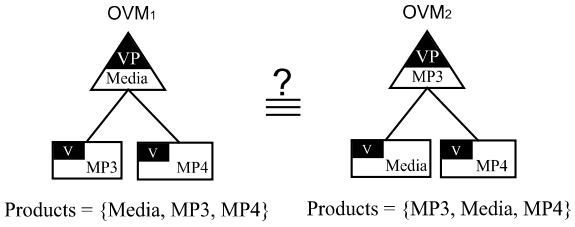

Figure 9: Equivalent models?

But, if we consider that the semantic domain of OVM is $\mathcal{PP}(V)$, then the models are not equivalents because they represent different set of products, $\{MP3, MP4\} \neq \{$Media, MP4$\}$.

5. Conclusion and future work

The main contribution of this paper is to go forward in the discussion about the proposal existent in the literature regarding the formalization of OVM. We want to clarify what are the main differences between FMs and OVM to avoid future misinterpretation. There are differences between their abstract syntax like, the sort of nodes, the graph structure, types of information that capture, and the constraint that can be specified. On the other hand, in spite of

their semantic domain is considered the same, i.e a set of sets of combinations of nodes ($\mathcal{PP}(nodes)$), we consider that should be possible define the semantic domain of OVM as a set of sets of combinations of variants ($\mathcal{PP}(V)$). We think that we need to find out what is the most adequate semantic domain to deal with OVM in order to design a reasoning tool.

We trust that a well understood formal language is the starting point for our future work toward a safe automated reasoning tool for analysis of OVM models. In order to provide this tool, the next step of our work is to specify all the analysis operations that may be applied to OVM and formally define them.

6. Acknowledgement

We would like to thank David Benavides and Antonio Ruiz Cortés for their constructive comments. This work was partially supported by Spanish Government under CICYT project Web-Factories (TIN2006-00472) and by the Andalusian Government under project ISABEL (TIC-2533) and Evangelischer Entwicklungsdienst e.V. (EED).

References

[1] D. S. Batory. Feature models, grammars, and propositional formulas. In *Software Product Lines Conference*, volume 3714 of *Lecture Notes in Computer Sciences*, pages 7–20. Springer–Verlag, 2005.

[2] D. Benavides. *On the automated analysis of software product lines using feature models*. PhD thesis, University of Sevilla, 2007.

[3] D. Harel and B. Rumpe. Meaningful modeling: What's the semantics of "semantics"? *IEEE Computer*, 37(10):64–72, 2004.

[4] P. Heymans, P.-Y. Schobbens, J.-C. Trigaux, Y. Bontemps, R. Matulevicius, and A. Classen. Evaluating formal properties of feature diagram languages. *IET Software*, 2(3):281–302, June 2008.

[5] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. Feature–Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, November 1990.

[6] A. Metzger, K. Pohl, P. Heymans, P.-Y. Schobbens, and G. Saval. Disambiguating the documentation of variability in software product lines: A separation of concerns, formalization and automated analysis. In *Requirements Engineering Conference, 2007. RE '07. 15th IEEE International*, pages 243–253, 2007.

[7] K. Pohl, G. Böckle, and F. J. van der Linden. *Software Product Line Engineering: Fundations, Principles and Techniques*. Springer–Verlag, Berlin, DE, 2005.

[8] Roos-Frantz and S. Segura. Automated analysis of orthogonal variability models. a first. In *1st Workshop on Analyses of Software Product Lines (ASPL08)*, 2008.

[9] P.-Y. Schobbens, P. Heymans, J.-C. Trigaux, and Y. Bontemps. Generic semantics of feature diagrams. *Computer Networks*, 51(2):456–479, Feb 2007.

[10] M. Sinnema and S. Deelstra. Classifying variability modeling techniques. *Information & Software Technology*, 49(7):717–739, 2007.