Automated Analysis of Orthogonal Variability Models. A First Step

Fabricia Roos-Frantz *†
Universidade Regional do Noroeste do Estado do RS (UNIJUI)
São Francisco, 501.
Iju 98700-000 RS (Brazil)
frfrantz@unijui.edu.br

Sergio Segura
Department of Computer Languages and Systems
University of Seville
Av. de la Reina Mercedes S/N,
41012 Seville, Spain
sergiosegura@us.es

Abstract

The automated analysis of variability models is a challenge to be reached in SPLE (Software Product Line Engineering). Only recently researchers have devoted their attention to the reasoning on these models. However, their work has focused on Feature Models. Orthogonal Variability Modeling (OVM) is one of the approaches for modeling variability in software product line. Hence, an automated support is needed to reasoning on orthogonal variability models (OVMs). Although the automated analysis of OVMs has been proposed, it only deals with a small number of analysis operations, which are implemented using a specific logical representation and solver. In this position paper, we present the proposal that we will carry out to achieve an adequate tool to the analysis on OVMs. As part of this paper, we informally define some analysis operations on OVMs. In addition, we propose to study the possibility of extending FAMA framework for supporting analysis on OVMs. We consider that FAMA (FeAture Model Analyzer) could be a suitable option to automate this analysis since it provides a formal basis, integrate multiple solvers and already provide tools.

1. Introduction and Preliminaries

The automated analysis of variability models is a challenge to be reached in SPLE (Software Product Line Engineering). Although there are several kinds of variability models, the majority of the research works on analysis of these models has focused on Feature Models. In the literature, there are different proposals providing automated support for the analysis of feature models [3, 4, 8, 9, 10, 11, 13, 18, 22]. Each one of them use different logical paradigm or formalism to provide the automated support (e.g. description logic, propositional logic, constraint programming). Most of them use SAT, BDD or CSP off-the-shelf solvers to automate various analysis operations, e.g. , checking if a product is valid, checking if a model is void, detecting dead features, etc.

To the best of our knowledge only Metzger et al. [13] provide a automated support for the analysis of OVM. They introduce a formalization of OVMs and propose using SAT[1] solvers to automate analysis operations on OVMs. This proposal only deal with five operations and whose semantics is based on feature models. Besides, they use just one type of solver to automate the analysis. The various solvers (SAT, BDD and CSP solvers) have varying degrees of performance and coverage with regard to analysis operations [2, 5].

Benavides [4] proposes a formal framework FAMA-F for the automated analysis of software product lines in general and feature models in particular. The framework is independent of the variability model (VM) used for the analysis. FAMA-F integrate some of the most commonly used logic representations and solvers proposed in the literature (BDD[2], SAT[3] and CSP[4] solvers are implemented). It integrates different solvers in order to combine the best of all of them in terms of performance. We wonder if is possible to extend this framework to automate various reasoning tasks on OVMs, e.g., verifying if a product is valid, checking if a model is void, detecting "dead" nodes, etc. (see Sect. 3). We consider that FAMA-F could be a suitable option to automate the analysis of OVM since it provides a formal basis, integrate multiple solvers and has available tools.

To achieve a suitable automated support for the analysis of OVMs, we identify three issues to be addressed, namely: *i)* specification of operations, *ii)* formal representation of

*PhD student at the University of Sevilla

†Supported by the Evangelischer Entwicklungsdienst e.V. (EED)

[1] They use SAT4j solver http://www.sat4j.org

[2] JavaBDD solver, http://javabdd.sourceforge.net

[3] SAT4j solver http://www.sat4j.org

[4] Constraint Satisfaction Problem www.4c.ucc.ie/

model and operations, and *iii)* implementation of operations. These aspects motivated some of the main research questions to be addresses in our future work.

1.1. OVM (Orthogonal Variability Model)

OVM is a proposal for documenting software product line variability [14]. 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 1). 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 1). Cardinality determines how many Vs may be chosen in an alternative choice, at least *min* and at most *max* Vs of the group. Figure 1 depicts the graphical notation for OVMs [14, 13].

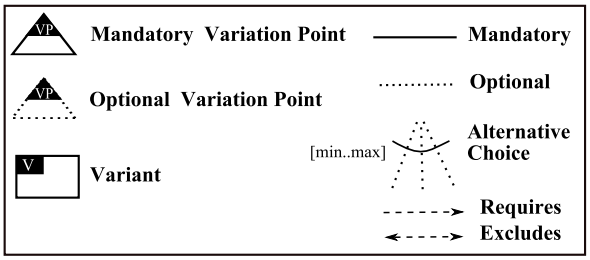

Figure 1. Graphical notation for OVM

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 2 depicts a example of an OVM inspired by the mobile phone industry.

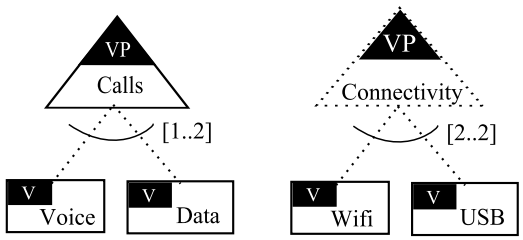

Figure 2. OVM example: mobile phone product line

1.2. Automated Analysis of OVM

To the best of our knowledge, there is only one proposal dealing with the automated analysis of OVM [13]. Metzger et al. are working in a tool support for variability management, which offers support for the analysis of OVM. Their prototype uses the off-the-shelf SAT solver library SAT4J. This SAT solver request a Boolean formula in CNF (conjunctive normal form) and delivers all variable assignment that evaluate the input formula true. If no such assignment exists, the formula is unsatisfiable. This proposal provides analysing of only five operations and using one solver. Furthermore, it makes the automated reasoning on OVM using the VFD semantics.

VFD (Varied Feature Diagram) is based on FFD (Free Feature Diagrams) which is a parametric construct designed to define the syntax and semantics of FODA-inspired FD (Feature Diagram) languages in a generic way [16, 17]. Metzger et al. propose reusing this formalization of feature diagrams, in other words VFD, to introduce a formalization of OVMs. They introduce a formal version of OVMs abstract syntax and describe a translation from OVM to VFD, thereby they give OVM a formal semantic.

1.3. FAMA framework

FeAture Model Analyzer (FAMA-F), proposed by Benavides [4], is a formal framework for the automated analysis of software product lines in general and feature models in particular, in other words, this is a framework independent of the variability model. This framework defines different reasoning operations on feature models, like calculating the number of products in a Software Product Line (SPL), getting a list of its products, filtering products according to a criterion or detecting and explaining errors. Thus, we presume it could be extended with OVM. It is defined with a high abstraction level, provides support for the most extended feature model notations and can be extended with new operations and solvers as needed.

The FAMA-F is defined in four layers from a higher (i.e. abstract foundation layer) to a lower abstraction level (i.e. implementation layer), see Figure 3.

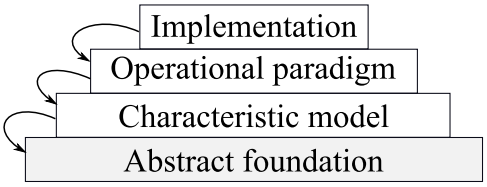

Figure 3. The four-layers FAMA-F

The FAMA-F abstract foundation layer provides an abstract and formal definition of software product lines and the operations of analysis that can be performed on them. The abstract foundation layer defines characteristic models as those that describe the allowed products configurations of the software product line. In the FAMA-F characteristic model, a specific variability model must be formally defined. Up to now feature models are the only variability model considered. In this layer the semantics of a specific feature model (FAMA-FM) is formalized with a formal language Z. The operational paradigm layer depends on the variability model used and provides a logical representation to the semantics of feature model and analysis operations, it allows using different logical representation. In the implementation layer, a translation from the logical representation described in the operational paradigm layer to the real solvers is provided. FAMA-F allows the usage of multi solvers to resolve a logical representation, may be used CSP, SAT and BDD solvers.

Additionally, Benavides et al. [7] presented an implementation of this framework, the *FAMA Eclipse plug-in* (FAMA-EP)[5]. It is an extensible tool for the automated analysis of feature models that integrated three logic paradigms and their respective solvers: Constraint Solver Programming (CSP) by means of JaCoP, Propositional Satisfiability (SAT) by means of SAT4j and Binary Decision Diagram (BDD) by means of JavaBDD. FAMA-EP allows the integration of different logic representations and solvers in order to optimize the analysis process.

In addition to FAMA-EP, Benavides et al. [20] provide the *FAMA Framework (FAMA-FW)* with the intention of allowing third parties to integrate their automated reasoning techniques into a workspace where some basic features are provided by default. FAMA-FW is a tool for the automated analysis of variability models (VM). Its main objective is providing an extensible framework where current research on VM automated analysis might be developed and easily integrated into a final product. It is built following the SPL paradigm supporting different variability metamodels, reasoners or solvers, analysis questions and reasoner selectors, easing the production of customized VM analysis tools. FAMA-FW is the result of research presented in [8, 5, 2, 19].

[5] http://www.isa.us.es/fama

1.4. Structure of this paper

The remainder of this paper is structured as follows: Section 2 provides an overview about semantics OVM. Section 3 provides an informal specifications of some operations on OVM. In Section 4 we propose the use of FAMA framework as tooling to automated analysis of OVMs. Finally, we summarize our future research in Section 5.

2. What specific formal semantics of OVM should be used?

To carry out the automated reasoning on OVM, we need a well defined semantics of these model. There are several options to give formal semantics to OVM. One of them is translating OVM to a feature model, since the semantic of these has already been defined [16, 17]. Another way is using a formal language, such as Z [21] or B [1].

To the best of our knowledge, there is only one formal semantics OVM in the literature, proposed by Metzger et al.[13], which has been obtained through a translation from OVM to VFD. This way to give semantics to OVM makes its semantics dependent of VFD. Such dependency may result in an drawback, due to the increase in the model size. When a OVM model is translated into a VFD model, a larger number of non-primitive nodes are generated and consequently the model becomes bigger, therefore it may damage the performance of the operations. Besides, one of the advantages that OVM offers is a significant reduction in the model size and complexity, because only the variable aspects of a product line are documented in a first-class model [15]. We wonder using VFD to give semantics to OVM contradict the advantage of OVM concerning the model size.

We will study other ways to give semantics to OVM aiming to evaluate these alternatives and have more conclusions. The first work will be on translating OVM to the feature model metamodel used in FAMA-F. This approach would allow us to reuse the formal foundation and support multi solver [6] provided by FAMA-F for reasoning on OVM. The other alternative is to study the usage of a formal specification language Z or B, without the translation to another language.

3. Operations on OVM

In the last years, different analysis operations over feature models have been identified [5, 4, 17, 3]. We select some of them and informally describe these on OVM. These operations observe the properties of a model without modifying it, they take a OVM as an input and provide a response as a result. Next, we define informally those operations, namely:

Valid product. This operations checks whether a given product belongs to the set of products represented by the OVM or not. For instance, let us consider the products P1 and P2, described below, and the OVM of Figure 2.

P1 = {*Connectivity, USB, Wifi*}

P2 = {*Calls, Data, Connectivity, USB, Wifi*}

The product P1 is not a valid product for the OVM since it does not include the mandatory variation point *Calls*. On the other hand, the product P2 is a valid product for the OVM because it is included in the set of products represented by the model.

Void OVM. This operation checks whether a OVM is void or not, i.e. if it represents at least one product. The reasons that may make a OVM to be void are related with a wrong usage of the constraint dependencies. As an example, Figure 4 depicts a void OVM. The excludes_VP_VP constraint makes not possible the selection of the mandatory VP *Functions*, what adds a contradiction to the model.

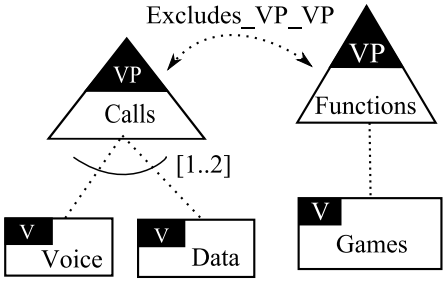

Figure 4. Void OVM

Core nodes. This operation returns the set of nodes (variants or variation points) that appear in all products of the software product line. For instance, the set of core nodes of the OVM presented in Figure 2 is {*Calls*}.

Dead nodes. This operation returns a set of dead nodes (if any), i.e. those that do not appear in any product. Dead nodes are caused by a wrong usage of constraint dependencies and are the responsible of making a OVM to be void. The Figure 5 depicts some common cases of dead nodes on OVMs. Dead nodes in the figure are labeled with D.

All products. This operation returns all the products represented by a model. As an example, the set of all the products of the OVM presented in Figure 2 is detailed below:

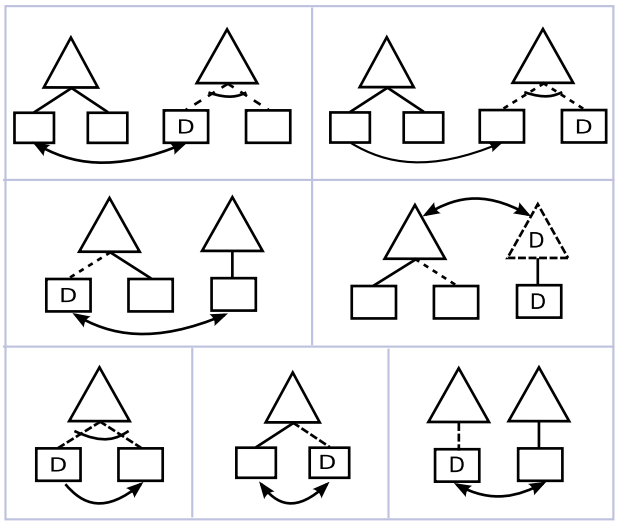

Figure 5. Common cases of dead nodes on OVM

P1 = {*Calls, Voice*}

P2 = {*Calls, Voice, Connectivity, Wifi, USB*}

P3 = {*Calls, Voice, Data, Connectivity, Wifi, USB*}

P4 = {*Calls, Data*}

P5 = {*Calls, Data, Connectivity, Wifi, USB*}

To automate the computation of those operations identified we have to formally define them. The formalization found in the literature for this operations were defined according to VFD semantics [13]. Metzger et al. suggest formal definition to the operations: *void model*, *valid product*, *core nodes*, *dead nodes* and *all products*. We will study how to formalize these and all the others operations identified in relation to OVM.

3.1. How to specify the equivalent models operation on OVM?

The equivalent models operation checks whether two models are equivalent. Two models are equivalent if they represent the same set of products [4]. If we observe the example depicted in the Figure 6, we can say that both models are equivalent, because they represent the same set of products. We believe that this operation is not correct to OVM because a *variant* is different of a *variation point*. In the product of the first model, *Media* is a *variation point* and in the second one, *Media* is a *variant*. Therefore, we believe that these models represent different set of products, then the equivalent operation should be redefined for OVM.

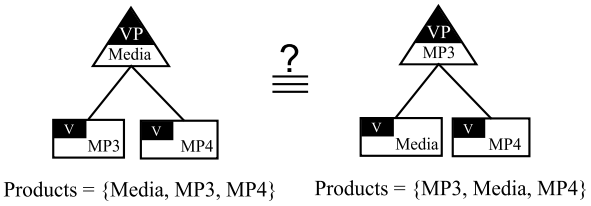

Figure 6. Equivalent models?

3.2. How to specify the merging operation on OVM?

We conclude that a new concept of merging operation can be applied to OVM, the merging into parts. This operation takes as input one part from one OVM model and another part of another OVM model and returns a new OVM model with the merging of those parts.

Three kinds of merging operations on feature models were identified by Schobbens et al. [17], particularly: Intersection, Union and Reduced product. The former, returns a feature model that encompasses the products included in both inputs models. The second, returns a feature model that encompasses all the products included in any of the inputs models. The later, returns a feature model including all the products of the input models plus all the new possible feature combinations. The merging of models may be helpful in a collaborative environment in which different people modify the model concurrently [12].

The Figure 7 depicts a visual example of the merging into parts operation on OVM, which is based on the merging operations proposed in [17]. According to the example, the *Connectivity VP* of the model *A* allows configuring three products: P1{Connectivity, Wifi}, P2{Connectivity, USB}, and P3{Connectivity, Wifi, USB}. The *Connectivity VP* of the model *B* also allows configuring three others products: P1{Connectivity, Wifi}, P2{Connectivity, Bluetooth}, and P3{Connectivity, Wifi, Bluetooth}. Then, if we merge both *Connectivity VPs*, we will have three new models as the result of the merging operation.

4. Tooling

We will study how to extend FAMA-F [4] to support the analysis of OVMs. FAMA-F is a framework supporting the usage of different logics paradigms and solvers in order to optimize the performance of the analysis process. This framework is independent of the type of variability model, e.g. it is possible to use feature models or OVM. The first step to be done is to define the FAMA's characteristic model layer using OVM as variability model of SPL. At this layer OVM must be formally defined using the specification language Z. Afterwards, OVM must be translated by a logical representation. In particular, in the operational paradigm layer of FAMA-F, a OVM will be translated into a generic Constraint Satisfaction Problem (CSP). At last, we have to translate from the abstract CSP to the real CSP solver. At this framework multiples solver can be used: CSP, SAT and BDD solvers. Additionally, we can use the tool FAMA-FW as a support for implementing our tooling.

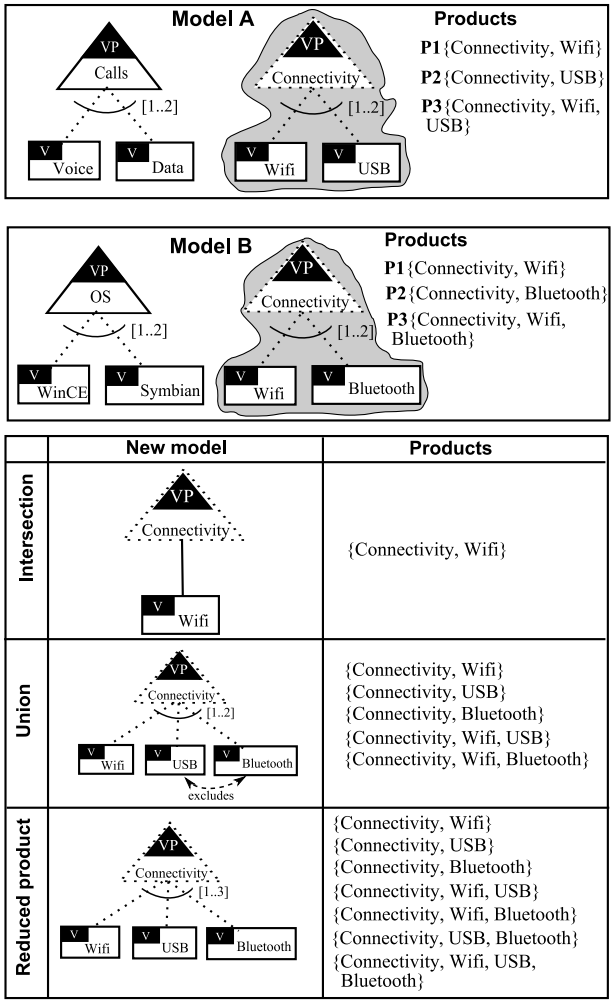

Figure 7. Merging into parts on OVM

5. Future work

We will study a way to achieve a suitable tooling to analysis of OVMs. To do this, we intend to study what is the adequate semantics OVM to be used. Besides, we will specify all the operations that may be applied to OVM and to formally define them. Additionally, as one of the main moti-

vations of our research, we will study a possible extension of the FAMA framework for supporting analysis operations on OVM.

6. Acknowledgement

This work was partially supported by the European Commission (FEDER) and Spanish Government under CICYT project Web-Factories (TIN2006-00472) and by the Andalusian Government under project ISABEL (TIC-2533).

References

[1] J. R. Abrial. *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.

[2] D. Batory, D. Benavides, and A. Ruiz-Cortés. Automated analysis of feature models: Challenges ahead. *Communications of the ACM*, December:45–47, 2006.

[3] 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.

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

[5] D. Benavides, A. Ruiz-Cortés, P. Trinidad, and S. Segura. A survey on the automated analyses of feature models. In *Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*, pages 367–376, 2006.

[6] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Corts. A first step towards a framework for the automated analysis of feature models. In *Managing Variability for Software Product Lines: Working With Variability Mechanisms (SPLC'06)*, 2006.

[7] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Corts. FAMA: Tooling a framework for the automated analysis of feature models. In *Proceeding of the First International Workshop on Variability Modelling of Software-intensive Systems (VAMOS)*, pages 129–134, 2007.

[8] D. Benavides, P. Trinidad, and A. Ruiz-Cortés. Automated reasoning on feature models. In *Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005*, volume 3520 of *Lecture Notes in Computer Sciences*, pages 491–503. Springer–Verlag, 2005.

[9] K. Czarnecki and A. Wasowski. Feature diagrams and logics: There and back again. In *11th International Software Product Lines Conference (SPLC 2007)*, volume 0, pages 23–34. IEEE Computer Society, 2007.

[10] S. Fan and N. Zhang. Feature model based on description logics. In *Knowledge-Based Intelligent Information and Engineering Systems*, volume 4252, pages 1144–1151. Springer, 2006.

[11] M. Mannion. Using first-order logic for product line model validation. In *Proceedings of the Second Software Product Line Conference (SPLC2)*, LNCS 2379, pages 176–187, San Diego, CA, 2002. Springer.

[12] T. Mens. A state-of-the-art survey on software merging. *IEEE Trans. Softw. Eng.*, 28(5):449–462, 2002.

[13] 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.

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

[15] K. Pohl, F. van der Linden, and A. Metzger. Software product line variability management. In *SPLC*, page 219, 2006.

[16] P. Schobbens, P. Heymans, J. Trigaux, and Y. Bontemps. Feature diagrams: A survey and a formal semantics. In *Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)*, Minneapolis, Minnesota, USA, September 2006. IEEE Computer Society.

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

[18] J. Sun, H. Zhang, Y.-F. Li, and H. H. Wang. Formal semantics and verification for feature modeling. In *Proceedings of the ICECSS05*, pages 303–312, 2005.

[19] P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés, and M. Toro. Automated error analysis for the agilization of feature modeling. *Journal of Systems and Software*, 81(6):883–896, 2008.

[20] P. Trinidad, D. Benavides, A. Ruiz-Corts, S. Segura, and A. Jimenez. Fama framework. In *Software Product Line Conference Tool Demonstrations (SPLC 08 Tools Demos) (in press)*, 2008.

[21] J. Woodcock and J. Davies. *Using Z: Specification, Refinement, and Proof*. Prentice Hall, 1996.

[22] W. Zhang, H. Zhao, and H. Mei. A propositional logic-based method for verification of feature models. In J. Davies, editor, *ICFEM 2004*, volume 3308, pages 115–130. Springer–Verlag, 2004.