Automated Analysis of Diverse Variability Models with Tool Support

Fabricia Roos-Frantz, José A. Galindo, David Benavides, and Antonio Ruiz Cortés

Department of Computer Languages and Systems
University of Seville
Avda. Reina Mercedes s/n, 41012, Seville, Spain
fabriciaroos,jagalindo,benavides,aruiz@us.es
http://www.isa.us.es

Abstract. Over the past twenty years, there have been many contributions in the area of automated analysis of variability models. However, the majority of these researches are focused on feature models. We propose that the knowledge obtained during recent years on the analysis of feature models can be applied to automatically analyse different variability models. In this paper we present FaMa OVM and FaMa DEB, which are prototypical implementations for the automated analysis of two distinct variability models, namely Orthogonal Variability Models and Debian Variablity Models, respectively. In order to minimise efforts and benefit from the feature model know–how, we use FaMa Framework which allows the development of analysis tools for diverse variability modelling languages. This framework provides a well tested system that guides the tool development. Due to the structure provided by the framework, FaMa OVM and FaMa DEB tools are easy to extend and integrate with other tools. We report on the main points of both tools, such as the analysis operations provided and the logical solvers used for the analysis.

1 Introduction

Variability modelling is an important task in software product line engineering (SPLE). In the literature, there are many kinds of models used for modelling variabilities [1]. The automated analysis of these models is a thriving research area. Over the past twenty years, there have been numerous contributions in this area including both, research papers and tools[2]. The automated analysis of variability models is defined as the computer-aided extraction of information from such models [2]. Some examples of analysis on models are: checking whether a configuration is valid, checking whether a model is void, etc. Although there are other variability modelling techniques, the majority of the research on analysis has focused on feature models. In this paper, we show an example of how to implement the automated analysis of other variability models, particularly Orthogonal Variability Model (OVM) [3] and Debian Variability Model. For implementing these analysis we used the FaMa framework (FaMa FW) [4],

which is an open source Java framework that simplifies the development of tools for the analysis of variability models. It is worth highlighting that in this paper we address OVMs and Debian variability models, however these are only two possibilities, amongst others. The idea of automated analysis of variability models in general, and its implementation using FaMa FW in particular, could be used to support other variability models in different scenarios. This is the main point that our paper wants to report.

OVM is a modelling language for representing variability in SPLE, which focuses on separating the representation of variability from the representation of the various SPLE artefacts [3]. On the other hand, regarding Debian Variability models, we found that Debian based Linux distributions use a language to manage their possible conflicts and dependencies between software packages [5]. We found that this language can be interpreted and analysed as a variability model, those variability models are what we call Debian variability models.

As reviewed by Benavides et al.[2], there are different proposals providing automated support for the analysis of feature models, by using different logical paradigm or formalism (e.g. Description logic, Propositional logic, Constraint programming). Most of them use binary decision diagram (BDD), satisfiability problem (SAT) or constraint satisfaction problems (CSP) off–the–shelf solvers to automate the analysis. However, to the best of our knowledge, only Metzger et al. explored the automated analysis of OVM [6]. They propose the usage of SAT solver to automate this analysis. Furthermore, in the case of software packaging, there have been various proposals for managing the dependencies of software packages. Most of these proposals consider the satisfaction of dependencies between packages [7, 8]. However, with FaMa DEB, we found that those models can be automatically analysed in a software product line variability model-like style. In [9] we provide a mapping from this variability model to propositional formulas. This new view also allows us to extend the number of available analysis operations for Debian repositories since nowadays only few operations are available into software packaging domain [8].

Although FaMa FW has been designed to analyse feature models and it has already been extended to support different flavours of this model, it is easily extendable to support the analysis of other variability models. Therefore, in this paper we propose a way to automate the analysis of two kinds of variability models arising from two different domains.

This paper is structured as follows, in Section 2 we present our tools FaMaOVM and FaMaDEB and the common parts they share. The concretes parts of each tool are described in subsection 2.1 for FaMaOVM and in subsection 2.2 for FaMaDEB. We present a small review of the related work that motivated the needing of analysis tools for different variability models in section 3. Finally in Section 4 we present our conclusions and future work.

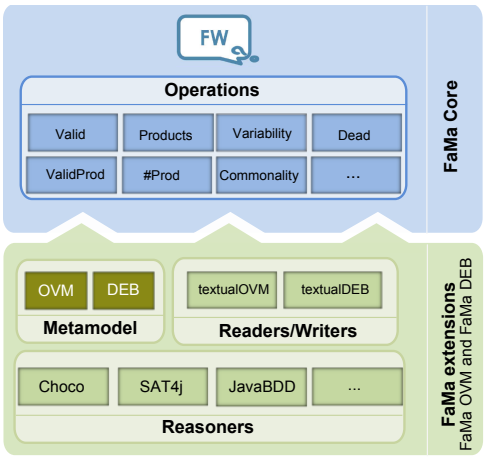

Fig. 1. FaMa OVM and FaMa DEB, extensions of FaMa FW

2 The tools

FaMa OVM and FaMa DEB are tools for the automated analysis of OVMs and Debian Variability Models, respectively. Both tools are extensions of the FaMa FW. This framework provides a number of extension points to plug in new components called metamodels, readers/writers and reasoners. Figure 1 shows the extensions needed for implementing FaMa OVM and FaMa DEB. The metamodel component represents the relationships and variability elements of each specific model. A reader/writer implements loaders/savers to an specific metamodel format, and a reasoner implements analysis operations using a specific logic solver.

FaMa FW provides a set of analysis operations to observe the properties of a model without modifying it. Each operation takes a model as input and provides a response as result. More details about analysis operations can be found in [2]. Next we summarize some of the analysis operations implemented by FaMa OVM and FaMa DEB:

- *Void model*: checks whether a model is void or not, i.e. if it represents at least one valid product. A model may become void due to the wrong usage of constraints.
- *#Products*: returns the total number of valid products represented by the model received as input.
- *Valid Product*: takes a model and a product (set of variability elements) as input and returns a value that determines whether the input product belongs to the set of products represented by the model or not.
- *All Products*: takes as input a model and returns all the valid products represented by this model.

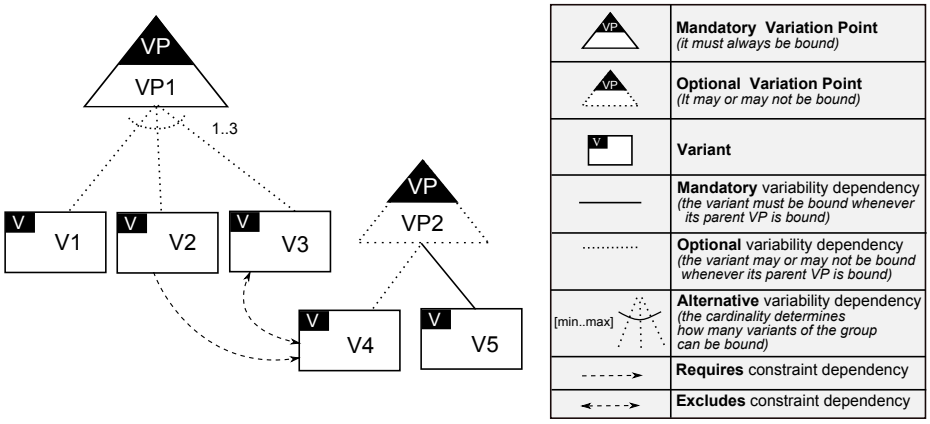

Fig. 2. OVM notation

- *Valid Partial Configuration*: takes a model and a partial configuration as input and returns a value informing whether the partial configuration is valid or not, i.e. a partial configuration is valid if it does not include any contradiction.
- *Filter*: takes as input a model and a configuration (potentially partial) and returns the set of valid products including the input configuration that can be derived from this model.
- *Dead Node*: returns a set of dead nodes (if any), i.e. those that cannot appear in any of the valid products represented by the model. Dead nodes are caused by the wrong usage of constraint dependencies.
- *Commonality*: takes a model and a configuration as inputs and returns the percentage of valid products including the input configuration.

2.1 The FaMa OVM tool

OVM is a modelling language for defining the variability of a software product line separately without change the base models (e.g. requirement model, design model). The base models realise the variability defined by the variability elements in the OVM model. In OVM the first-classes are: *variation points (VP)* and *variants*. A variation point documents the aspects that can vary in the product line, which must be chosen by the customer or engineer of the software product line. A variant is related to a variation point and documents how this variation point can vary. Figure 2 shows an example of an OVM diagram.

We have successfully used this tool to analyse OVM models using attributes in an industrial case study in the automotive domain [10]. The main ideas behind the mapping of OVM to constraint programming were presented in [11, 10].

The OVM metamodel In the context of automated analysis, an OVM is interpreted as a set of possible products that can be derived from the product

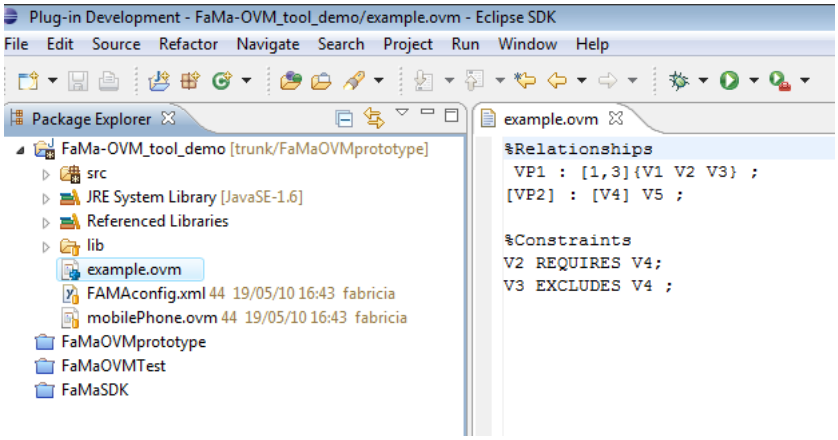

Fig. 3. OVM textual format

line. Thus, the OVM metamodel describes the different variability elements and the rules that constraint the combination of these elements in a product line. In [12] you can find a detailed description of those rules.

FaMa OVM textual format FaMa OVM uses as standard a textual format input file that describes the OVM model. Figure 3 shows a screenshot of the textual editor, in which the example of Figure 2 is represented using the OVM textual format. The *%Relationships* section of the "ovm" file describes the relationships (mandatory, optional and alternative) amongst the different variability elements, and the *%Constraints* section describes the requires and excludes relatioships.

FaMa OVM solvers Due to the benefit offered by the FaMa FW, we have the basis for implementing the analysis operations using multiple solvers. To enable the automated analysis of OVM we must define the mapping from OVM to constraints for an specific off–the–shelf solver. All the analysis operations previously described are currently implemented using Choco [13] and Sat4j [14] solvers.

2.2 The FaMa DEB tool

We found that Debian–based Linux distributions use a language to describe dependencies in software packages repositories. Software packages repositories are used in Debian individual installations in order to add, remove or update packages and manage their possible conflicts and dependencies [5]. We realised that this language can be interpreted as a variability language similar to those of feature models, OVMs and other variability description languages. Thus, if

we consider a software package as a feature of a distribution, a Debian–based Linux distribution could be interpreted as a product line and the individual installations could be therefore considered the products of the product line. In this context, a product is a set of installed packages, i.e. a Debian installation. Similarly, we assume that a set of repositories define an SPL of Debian-based distributions. We may remark that this could also be applied to other dependency languages such as RPM [7, 15] (previously called Red-Hat Package Manager) but for simplicity FaMa DEB does not cover it at the moment.

The DEB Metamodel The FaMa DEB metamodel is needed to represent and store the information present into Debian repositories. It is composed by packages (varibility elements) and the set of relationships with the meaning described in [5]. Note that not all information stored in the Debian repository describes the variability of the linux distribution (i.e. the homepage of the software maintainer). In [9] you can find a detailed description of those relations and the required mapping.

FaMa DEB textual format FaMa DEB uses as standard input the file which describes a Debian Repository. This file is usually called Debian Description Language in the literature [8]. Figure 4 shows a fragment of a repository configuration file. For each package in the file, a set of properties are presented. A property is composed of a name and one or more values associated with it. Properties may be divided into those providing basic information about the package (e.g. name, version) and those describing the relationships with other packages.

```
Package: openoffice.org-gnome
Essential: No
Version: 1:2.4.0-3ubuntu6
Replaces: openoffice.org-common (<< 2.0.4~rc1-0),
Provides: openoffice.org-gtk-gnome, openoffice.org2-gnome
Depends: gconf2, libc6 (>= 2.1.3)
Pre-Depends: dpkg (>= 1.14.12ubuntu3)
Suggests: openoffice.org-evolution
Conflicts: openoffice.org2-gnome (<< 1:2.4.0-3ubuntu6)
Task: ubuntu-desktop, edubuntu-desktop, gobuntu-desktop
```

Fig. 4. Snippet of the fileformat

FaMa DEB solvers To enable the automated analysis of DebianVML we must define the mapping from DebianVML to constraints into an specific off–the–shelf solver such as Choco [13], Sat4j [14] or JaCoP [16]. Note that these solvers are widely used for the automated analysis of feature models [2]. At the moment

we provide ChocoSolver because it allows the use of non boolean variables those variables are required in the mapping of the complex constraints that are used to resolve packages with different versions (similar to attributes in a SPL context).

3 Related Work

In this section we briefly present some others contributions that focuses into the managing different kinds of variability modelling approaches. In [17] Czarneky et al, present a review of different modelling approaches from the industry and the open–source community emphasizing in the characteristics that those models shares or differs, our work shares the same interest into the fact that diferent variability models are required to meet the requirements of each context, thus we are providing tools that aim to help with the analysis of two different variability models from contexts, one from the industry (OVM) and another from the open-source community (FaMaDEB) reusing the knowledge from feature–modelling research context and applying it into others. In [18] authors did a chronological review of several variability models, they also provide a guideline of which of those tools have tool support, our work provides new tools for some of the paradigms that they mention that there not exists any tools.

Recently Amador Durán et al, presented FLAME[1], A TestBased Validated Formal Framework for the Automated Analysis of Software Product Lines using Feature Models. They propose a formal framework where operations have an explicit meaning. Their approach differs straight–way from ours but as that work have been tested in front of FaMa indirectly benefits our work. That fact enforces our hypothesis that the use of a framework for create tools for the automated analysis of different variability models reduces costs and efforts benefiting the sharing and reusing of knowledge in a research context.

4 Conclusion and Future Work

We provided a prototypical implementation for the automated analysis of OVMs and Debian variability models. The development of these tools were based on a framework for the analysis of feature models, which is a research area with more know–how. We realised that the framework simplified the development of our tools, we did not have to start from scratch and we reused most of the analysis operations applied to feature models. Both tools offer a simple public interface, by which the user just need to entry with the model (sometimes with some more data, it depends on the operation) and to select the operation to be answered by the tool.

Due to their simple front-end Java interface, FaMa OVM and FaMa DEB are easy to integrate with other tools. They are also easy to configure, since their configuration is done by means of a unique XML file. For example, FaMa DEB can be easily integrated into Synaptics [19] into Debian distributions, to prevent

[1] `http://www.isa.us.es/fama/?FLAME_framework`

users for unwanted situations. Besides, we believe that the Debian community could benefit from this automation being able to perform several analysis operations, as for example the detection of inconsistencies in repositories, which is currently done by hand (e.g. when an error is detected, it is manually corrected).

It is worth highlighting that, on the one hand, we used small–scale models to test FaMa OVM and the analysis results were obtained in an acceptable time for all analysis operations performed with both sat4j and Choco. On the other hand, with FaMaDEB we were able to deal with realistic large-scale variability models.

These are the main directions of our future work:

- Address new operations. We are looking for OVM analysis operations, as for instance how to calculate the better configuration in terms of costs. New support operations in FaMa OVM, such as equivalent models and merging of models, are being developed [20, 10]. With FaMa DEB, using our experience on feature models (e.g. development of analysis tools, benchmarking, detection of errors) we look for new operations to detect anomalies in Debian models such as conditionally dead packages or redundancies, probably the operations with more than one variability model can be very useful to Debian community improving the usability of packaging systems. Also the explanations of the errors to help this community on its daily work could help to improve the end user experience.
- Benchmarking and optimisation. In our preliminary tests we detected that the Debian based models automated analysis are computationally very expensive, more than 6 hours in a laptop machine with *main*, *multiverse*, *restricted* and *universe* repositories of Ubuntu 8.04 enabled (24780 packages). We will work on the optimisation of the codding to improve those results.
- Testing. We will use automated generated models for the analyses of OVMs in order to evaluate our tool with larger scale models. These models will be generated according to some desired properties, such as number of variants, percentage of constraints, or percentage of a specific relationship.

Material

The prototypes of FaMa OVM and FaMa DEB are available at `http://www.lsi.us.es/~dbc/material/jisbd14/`

Acknowledgment

This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project TAPAS (TIN2012-32273) and the Andalusian Government projects THEOS (TIC-5906) and COPAS (P12-TIC-1867).

References

1. L. Chen, M. A. Babar, and N. Ali, "Variability management in software product lines: a systematic review," in *SPLC*, 2009, pp. 81–90.
2. D. Benavides, S. Segura, and A. Ruiz-Corts, "Automated analysis of feature models 20 years later: a literature review," *Information Systems*, vol. 35, no. 6, pp. 615–636, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.is.2010.01.001
3. K. Pohl, G. Böckle, and F. J. van der Linden, *Software Product Line Engineering: Fundations, Principles and Techniques.* Berlin, DE: Springer–Verlag, 2005.
4. P. Trinidad, D. Benavides, A. Ruiz-Cortés, S. Segura, and A.Jimenez, "Fama framework," in *12th Software Product Lines Conference (SPLC)*, Sep 2008, p. 359.
5. "Debian policy manual, http://www.debian.org/doc/debian-policy/ch-relationships.html," accessed February 2012.
6. A. Metzger, K. Pohl, P. Heymans, P. 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*, 2007, pp. 243–253.
7. "Package management sat solver," http://files.opensuse.org/opensuse/en/b/b9/Fosdem2008-solver.pdf, accessed February 2012.
8. "EDOS project, http://www.edos-project.org," accessed February 2012.
9. J. A. Galindo, D. Benavides, and S. Segura., "Debian packages repositories as software product line models. towards automated analysis," in *Proceeding of the First International Workshop on Automated Configuration and Tailoring of Applications (ACOTA)*, 2010.
10. F. Roos-Frantz, D. Benavides, A. Ruiz-Corts, A. Heuer, and K. Lauenroth, "Quality-aware analysis in product line engineering with the orthogonal variability model," *Software Quality Journal*, pp. 1–47. [Online]. Available: http://dx.doi.org/10.1007/s11219-011-9156-5
11. F. Roos-Frantz, D. Benavides, and A. R. Cortés, "Automated analysis of orthogonal variability models using constraint programming," in *XV Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2010)*, 2010, pp. 269–280.
12. F. Roos-Frantz, "A preliminary comparison of formal properties on orthogonal variability model and feature models," in *VaMoS*, 2009, pp. 121–126.
13. "CHOCO solver, http://choco.emn.fr/," accessed February 2012.
14. D. L. Berre, "Sat4j. http://www.sat4j.org/," accessed February 2012.
15. "Rpm package manager http://www.rpm.org," accessed February 2012.
16. K. Kuchcinski and R. Szymanek, "Jacop. http://jacop.osolpro.com/," accessed February 2012.
17. K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid, and A. Wasowski, "Cool features and tough decisions: a comparison of variability modeling approaches," in *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems.* ACM, 2012, pp. 173–182.
18. L. Chen, M. Ali Babar, and N. Ali, "Variability management in software product lines: a systematic review," in *Proceedings of the 13th International Software Product Line Conference.* Carnegie Mellon University, 2009, pp. 81–90.
19. "Synaptic package manager http://www.nongnu.org/synaptic/," accessed February 2012.
20. F. Roos-Frantz and S. Segura, "Automated analysis of orthogonal variability models. a first step," in *First Workshop on Analyses of Software Product Lines (ASPL 2008). SPLC'08*, S. Thiel and K. Pohl, Eds., Limerick, Ireland, September 2008, pp. 243–248.