VariaMos: a Tool for Product Line Driven Systems Engineering with a Constraint Based Approach

Raul Mazo, Camille Salinesi, Daniel Diaz

Mazo Raúl, Salinesi Camille, Diaz Diaz. VariaMos: a Tool for Product Line Driven Systems Engineering with a Constraint Based Approach. 24th International Conference on Advanced Information Systems Engineering (CAiSE Forum'12), Springer Press, Gdansk-Poland, 25 – 29 June 2012.

VariaMos: a Tool for Product Line Driven Systems Engineering with a Constraint Based Approach

Raúl Mazo[1,2] , Camille Salinesi[1], Daniel Diaz[1]

[1] CRI, Université Paris 1 – Sorbonne, 90, rue de Tolbiac, 75013 Paris, France
[2] Ingeniería de Sistemas, Universidad de Antioquia, Medellín, Colombia

raulmazo@gmail.com, {camille.salinesi, daniel.diaz}@univ-paris1.fr

Abstract. The creation of error-free variability models and their usage in product line analysis and product derivation is central to product line engineering (PLE). The complexity of these tasks makes tool support a success-critical factor. Tools supporting the core activities of PLE are a challenge and a real need for academics, industrial researchers, and practitioners of the PLE domain. In this paper, we present a tool for variability modeling, model integration, verification and analysis, derivation requirements specification and product derivation.

Keywords: Product line engineering, variability, product line models.

1 Introduction

Variability models are used to specify the variability of software product lines. These variability models are represented by means of a modeling formalism. In our literature research, we have found quite a number of variability modeling formalisms, such as FODA (Feature-Oriented Domain Analysis) [9], Orthogonal Variability Models (OVM) [13], UML classes [26], DOPLER [5] and Goals [6]. To represent and reason on these models, a number of approaches and tools exist in the literature. However, there is a lack of methods and tools that can support modeling, integration, reasoning and complex configuration on the Product Line (PL) domain. This lack is more accentuated when the model is composed of a collection of views representing the same product line. In this paper, we present a tool allowing represent, integrate, reason and configure product line models.

The paper is structured as follows: Section 2 gives a brief overview of our tool VariaMos. Section 3 describes some functions of VariaMos. Section 4 presents related tools supporting integration, verification, analysis and configuration of product line models. Section 5 concludes the paper and describes future works.

2 VariaMos Architecture

VariaMos (Variability Models) is an Eclipse plug-in for specification, automatic verification, analysis, configuration and integration of multi-view product line models. From a deployment point of view, VariaMos is an Eclipse plug-in that communicates with our GNU Prolog [3] by means of a socket. The VariaMos tool, its documentation and a video training are available online[1].

3 Functionalities

VariaMos allows working simultaneously on a set of models in multi-formalism mode. There are several activities that VariaMos is intended to support: domain engineering with multiple models, integrated verification of the verification criteria existing in literature [1, 14], analysis [1] and configuration [10, 16]. In additiVn, MariaMos allows creating/editing Product Line Models (PLMs) that have been imported as SPLOT XMI[2] or constraint program text files (cf. Figure 1(a)) and exporting/importing PLMs using a XMI or a constraint program file. This functionality allows communicating models from and to other applications.

3.1 Integration of Variability Models by means of Constraint Programs

In our approach, each view of the product line system is transformed into a constraint program. A constraint program is a collection of constraints without a specific order. In this way, the constraint programs, representing the different views of the PL system, can be easily integrated into a single constraint program. The resulted constraint program represents the general system and offers a richer view of the PL (than individual views). VariaMos implements the five integration strategies presented by [10]. In our approach, two models' elements referring to the same concept must have the same name; we do not deal with mismatching of names. Mazo et al. [10] offer a list of rules to transform the most popular formalisms to represent variability models into constraint programs. Once each view of the PL system is transformed into CP, they can be integrated in a single constraint program using the graphical user interface presented in Figure 1 (b).

3.2 Verification of Variability Models

VariaMos implements the typology of verification criteria presented in [10]. Using this classification we can detect if the model is void [9], if the model is not a false PLM [1, 14], if the model does not have errors (like dead variables [1, 9, 14] or variables with wrong domains [1, 14], inconsistencies (like full-mandatory features [1] requiring optional features [9]) and redundancies (like full-mandatory variables in-

[1] https://sites.google.com/site/variabilitymodels/home/downloads/PresentationVariaMos2.js
[2] http://www.splot-research.org

cluded by another variable [14] or inclusion of a relative father [14]). A snapshot of the graphical user interface of VariaMos to implement these verification operations is presented in Figure 1 (c).

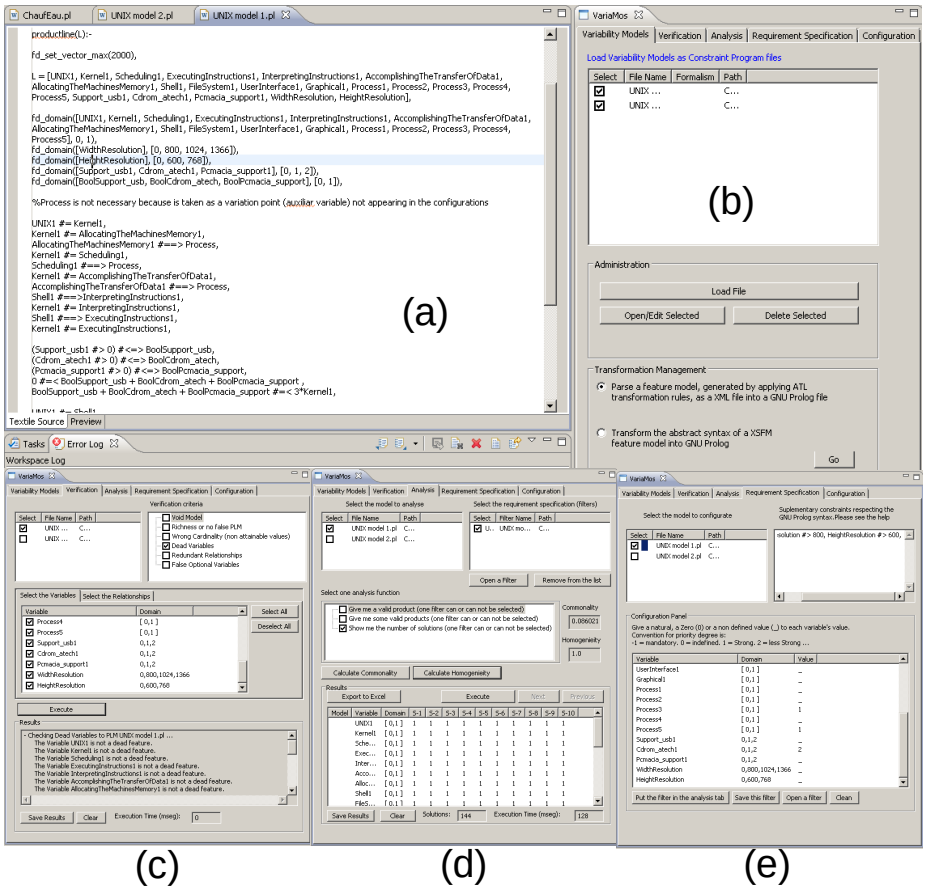

Fig. 1. GUI of VariaMos: (a) Definition/edition of Product Line Models, (b) Integration, (c) verification, (d) analysis and (e) configuration. Fig. 1 in high resolution is available at: https://sites.google.com/site/variabilitymodels/home/downloads/GUIofVariaMos.JPG

3.3 Execution of Analysis Operations

All the analysis operations implemented in VariaMos are taken from literature and from industrial projects with our partners; most of the operations are explained and referenced on the literature review of Benavides et al. [1]. A small description of each analysis operation implemented in VariaMos and how they have been implemented are presented as follows:

1. Calculating the number of valid products represented by the PLM. This operation may be useful for determining the richness of a PLM. VariaMos implements this operation with GNU Prolog in the following way: *g_assign(cpt,0), pl(_), g_inc(cpt), fail;g_read(cpt,N),* where *pl* is the fact that represents the product line model. With this operationalization we avoid the overload of the RAM with each solution generated and counted by the solver because each time a solution is found, we release the pile of solutions before the generation of a new one.
2. Obtaining the list of *all valid products* represented by the PLM, if any exist. This operation may be useful to compare two product line models. The list of valid product is obtained one by one from the solver by means of the backtracking technique. As the screenshot shows it in Figure 1(d), VariaMos provides users with the possibility to navigate in the list of products using the *Next* and *Previous* buttons.
3. Calculating commonality of a set of variables. This is the ratio between the number of products in which a given set of variables of the PLM is present and the number of products represented in the PLM. By default, this operation calculates the number of solutions in which all the variables of the PL are present and divides this number with the result obtained with operation 1.
4. Calculating Homogeneity: A more homogeneous PLM would be one with few unique variables in one product (i.e. a unique variable appears only in one product) while a less homogeneous one would be one with a lot of unique variables. By definition `Homogeneity = 1 - (#unicVariables / #products)`. This operation computes the number of variables that appear in only one product by means of a request to the solver and computes the number of products using the operation 1.
5. Calculating variability factor: This operation takes a PLM as input and returns the ratio between the number of products and 2^n where n is the number of variables considered. In particular, 2^n is the potential number of products represented by a PLM, assuming that there are not cross-tree constraints on the model and that all PLM's variables are Boolean. Variability factor = NProd / 2^ NVar. This function uses the solver to compute the number of variables and the number of products in the PLM.
6. Checking validity of a configuration. A configuration is a collection of variables and may be partial or total (e.g., the partial configuration presented in Figure 2(d)). A valid partial configuration is a collection of variables respecting the constraints of the PLM but not necessary representing a valid product. A total configuration is a collection of variables respecting the constraints of a PLM and where no more variables need to be added to form a valid product. This operation may be useful to determine if there are or not contradictions in a collection of variables or to determine whether a given product is available in a product line. To operationalize this function, the configuration to check is considered as a collection of external constraints where each constraint corresponds to the assignation of a particular value to each one of the variables of the PLM. Then, the external constrains and the constraints of the PLM are executed together in the solver to verify if the whole of constraints is consistent (i.e., there is a valid solution satisfying all these constraints).

7. Executing dependency analysis or decision propagation. It looks for all the possible solutions after assigning some fix value to a collection of values and then asking the solver for almost one solution. This operation is very similar to the operation 6, however, with this operation we can check the satisfaction of constraints by means of reification, and not only the satisfaction of variables of the PL as in operation 5.
8. Specifying external requirements specifications for configurations using constraints. This operation allows the specification of constraints that are not constraints of the domain, but configuration constraints. To operationalize this function, external constraints are defined in GNU Prolog and then added to the constraints of the PLM; once added, all the constraints are executed in the solver. See [10] for more details and Figure 1(e) for a snapshot of the implementation of this function in VariaMos.
9. Applying a filter. This operation takes a configuration (i.e., set of variables, each one with a particular value) and a collection of external requirements and returns the set of products which include the input configuration and respect the PLM's constraints and the external constraints. Figure 1(e) presents a snapshot of the GUI of this function in VariaMos.
10. Calculating the number of products after applying a filter. This operation uses the technique presented in operation 1 to compute the number of products that can be configured from a PLM in presence of a filter. A filter is presented as a collection of external constraints and particular assignation of values to the variables of the PL. To operationalize this function, the filter is added to the collection of the PLM's constraints and then executed in the solver. Figure 1(d) presents a snapshot of the GUI of this function in VariaMos.
11. Find an optimal product with respect to a given attribute like cost *(min goal)* and benefit *(max goal).* Detection of "optimal" products is very important for decision makers as presented in [10]. To operationalize this function we use the *fd_maximize* and the *fd_minimize* facts offered by the GNU Prolog solver.

3.4 Other Features

According to [8], a tool for automating reasoning on variability models should be efficient, scalable and with enough expressivity to represent different kinds of variability constraints. These characteristics are evaluated on VariaMos as follows:

Reasoning efficiency. The execution time of each reasoning operation can be calculated by the solver by means of a request for the current time (by means of the prolog function `user_time(T1)`) at the beginning and at the end (by means of the prolog function `user_time(T2)`) of each constraint program. The time spent by the solver to execute the operation at hand, is computed by means of the clause: `T is T2 - T1`. We have showed the reasoning efficiency of VariaMos in several works; for instance: [10, 12, 15] show the efficiency of VariaMos in verification of product line models and [11] shows the efficiency of VariaMos in transforming PLMs.

Scalability. VariaMos scalability has been validated using a corpus of 54 models specified in several languages, representing several domains and with sizes from 9 to 10000 variables. In all these cases, VariaMos shows a promising scalability in the

execution of the reasoning operation presented in this paper. The results have been reported in works like [10, 12, 15].

Expressivity. In VariaMos, product line models can be loaded as XMI or text files and then, labeled with it particular notation. VariaMos offers several capabilities to represent and transform different types of product line models into constraint programs. In addition, models can be edited with XML and text editors furnished by Eclipse IDE. The power of expression of VariaMos is compared with the one of constraint programming to specify PLMs [10, 15].

4 Related Works

The most of the tools for supporting product line engineering focus on one or two aspects but not in all of the aspects presented in this paper.

For instance, from the point of view of modeling, there are tools like Feature Plugin[3], XFeature[4], AHEAD Tool Suite[5], Pure::variants[6] and Requiline[7]. The most of these tools were built to graphically construct feature models and to derive products from these models, not to reason on these models.

From the point of view of analysis and verification, most of the tools found in literature are formalism-dependent and they only focus on feature models. In addition, most of them focus on verifying the consistency of a combination of features (a feature configuration) against the feature model. Tools like FAMA[8] and SPLOT[9] consider several analysis and verification operations over feature models; however, they have been targeted in the analysis and verification of models represented by a single view.

From the point of view of expressivity, modeling tools available in the literature are just starting to offer some model-to-model transformation capabilities, but these are still limited and often ad hoc. Some examples of these tools are: Andro-MDA[10], openArchitectureWare[11], Fujaba[12] (From UML to Java And Back Again), Jamda[13] (JAva Model Driven Architecture), JET[14] (Java Emitter Templates), MetaEdit+[15] and Codagen Architect[16]. There are also approaches that do combine multiple variability

[3] http://gp.uwaterloo.ca/fmp
[4] http://www.pnp-software.com/XFeature/
[5] http://www.cs.utexas.edu/~schwartz/ATS/fopdocs/
[6] http://www.software-acumen.com/purevariants/feature-models
[7] http://www-lufgi3.informatik.rwth-aachen.de/TOOLS/requiline
[8] http://www.isa.us.es/fama
[9] http://www.splot-research.org
[10] http://www.andromda.org.
[11] http://www.openarchitectureware.org/
[12] http://www.fujaba.de
[13] http://sourceforge.net/projects/jamda
[14] http://www.eclipse.org/articles/Article-ET/jet_tutorial1.html
[15] http://www.metacase.com/
[16] http://www.codagen.com/products/architect/default.htm

models, e.g., KumbangTools[17] combining the feature and component-based models. However, none of them deals whit transformation of product line models, where the semantic of the model represents not only one but an undefined collection of product models.

From the point of view of configuration, there are several tools in literature that address this topic. For instance, FAMA, SPLOT and FdConfig [16]; however these tools do not support as much reasoning operations over product line models as VariaMos do. In addition, they do not support reasoning operations over multiple PLMs.

5 Conclusions and Future Works

In this paper we introduced the first release of VariaMos which is an Eclipse plug-in for edition, integration, verification, analysis and configuration of PLMs. We introduced the functionalities of the tool and we exposed some of the most relevant design and implementation details. Finally, we showed the differences between VariaMos and other tools found in literature and we concluded that VariaMos supports more variability modeling languages, automatically verifies more criteria than the other tools, and is the first tool to implement reasoning operations over multi-views PLMs. Although VariaMos is not a mature tool yet, its promising capabilities of extensibility, interoperability, scalability, expressivity and efficiency will allow the tool to become accepted and used by the academic and industrial community in the future.

Several challenges remain for our future work. On the one hand, the implementation of more verification and analysis functions. For instance, verification against a meta model defined by users, incorporation of a guided process allowing correcting anomalies and support incorporation for incremental verification are envisaged for future releases. On the other hand, it is planned to incorporate, in our tool, a graphical representation of constraint programs, automation of PLM construction from a collection of products models, multi-stage configuration of products from complex requirements formulated as constraint programs and also connection with other kind of solvers; e.g., SAT (SATisfiability), BDDs (Binary Decision Diagrams) and SMTs (Satisfiability Modulo Theories) in order to improve the efficiency of certain reasoning operations.

Acknowledgments

Many thanks to Diego Quiroz, Sebastian Monsalve, and Jose Ignacio Lopez for their invaluable help with this tool.

[17] http://www.soberit.hut.fi/KumbangTools/

References

1. Benavides D., Segura S., Ruiz-Cortés A. “Automated Analysis of Feature Models 20 Years Later: A Literature Review”. Information Systems. Elsevier, 2010.
2. Czarnecki, K., Helsen, S., Eisenecker, U. “Formalizing cardinality-based feature models and their specialization”. Software Process Improvement and Practice, 10(1):7– 29, 2005.
3. Diaz D., Codognet P. “Design and Implementation of the GNU Prolog System”. Journal of Functional and Logic Programming (JFLP), Vol. 2001, No. 6, October 2001
4. Djebbi O., Salinesi C. “Towards an Automatic PL Requirements Configuration through Constraints Reasoning”. Int. Workshop on Variability Modelling of Software-intensive Systems (VaMoS), Essen, Germany, January 2008.
5. Dhungana D., Grünbacher P., Rabiser R. "The DOPLER Meta-Tool for Decision-Oriented Variability Modeling: A Multiple Case Study," *Automated Software Engineering*, 2010 (in press; doi: 10.1007/s10515-010-0076-6).
6. González-Baixauli B., Laguna M., Sampaio J. “Using Goal-Models to Analyze Variability”. First International Workshop VaMoS, 2007.
7. Griss, M., Favaro, J., Allesandro, M. “Integrating Feature Modeling with RSEB”. Proceedings of the 5th International Conference on Software Reuse, Vancouver, Canada, 1998.
8. Hai H. Wang, Yuan Fang Li, Jing Sun, Hongyu Zhang, Jeff Pan. “Verifying feature models using OWL”. Journal of Web-Semantics (2007) 117–129.
9. Kang K., Cohen S., Hess J., Novak W., Peterson S. “Feature-Oriented Domain Analysis (FODA) Feasibility Study”. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, November 1990.
10. Mazo R., Salinesi C, Djebbi O., Diaz D., Lora-Michiels A. “Constraints: the Heart of Domain and Application Engineering in the Product Lines Engineering Strategy”. International Journal of Information System Modeling and Design IJISMD. ISSN 1947-8186, eISSN 1947-819. April-June 2012, Vol. 3, No. 2.
11. Mazo R., Salinesi C., Diaz D., Lora-Michiels A. “Transforming Attribute and Clone-Enabled Feature Models Into Constraint Programs Over Finite Domains”. 6th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), Springer Press, Beijing–China, 8-11 June 2011.
12. Mazo R., Lopez-Herrejon R., Salinesi C., Diaz D., Egyed A. “Conformance Checking with Constraint Logic Programming: The Case of Feature Models”. In 35th Annual International Computer Software and Applications Conference (COMPSAC), IEEE Press, Munich-Germany, 18-22 July 2011. Best Paper Award.
13. Pohl K., Böckle G., van der Linden F. “Software Product Line Engineering: Foundations, Principles and Techniques”. In: Springer-Verlag New York, Inc., Secaucus, NJ, 2005.
14. Salinesi C, Mazo R. “Defects in Product Line Models and how to Identify them”. Software Product Line - Advanced Topic, edited by Abdelrahman Elfaki, InTech editions, ISBN 978-953-51-0436-0, April 2012.
15. Salinesi C., Mazo R., Diaz D., Djebbi O. “Solving Integer Constraint in Reuse Based Requirements Engineering”. 18th IEEE International Conference on Requirements Engineering (RE'10). Sydney - Australia. September-October 2010.
16. Schneeweiss D., Hofstedt P. “FdConfig: A constraint-based interactive product configurator”. International Conference on Applications of Declarative Programming and Knowledge Management (INAP) Vienna, Austria. September 28-30, 2011.