VariaMos: an extensible tool for engineering (dynamic) product lines

Raul Mazo, Juan C Muñoz-Fernández, Luisa Rincón-Perez, Camille Salinesi, Gabriel Tamura

VariaMos: an extensible tool for engineering (dynamic) product lines

Raúl Mazo
CRI, Université Paris 1 Panthéon Sorbonne, Paris, France
raul.mazo@univ-paris1.fr

Juan C. Muñoz-Fernández
Universidad Icesi, Facultad de Ingeniería
Departamento de TICs, Cali, Colombia
CRI, Université Paris 1 Panthéon Sorbonne - Paris, France
jcmunoz@icesi.edu.co

Luisa Rincón
Fac. de Ingeniería - Pontificia Universidad Javeriana – Cali
Cali, Colombia
lfrincon@javerianacali.edu.co

Camille Salinesi
CRI, Université Paris 1 Panthéon Sorbonne, Paris, France
camille.salinesi@univ-paris1.fr

Gabriel Tamura
Universidad Icesi, Facultad de Ingeniería
Departamento de TICs, Cali, Colombia
gtamura@icesi.edu.co

ABSTRACT
This paper presents the new release of VariaMos, a Java-based tool for defining variability modeling languages, modeling (dynamic) product lines and cyber-physical self-adaptive systems, and supporting automated verification, analysis, configuration and simulation of these models. In particular, we describe the characteristics of this new version regarding its first release: (1) the capability to create languages for modeling systems with variability, even with different views; (2) the capability to use the created language to model (dynamic) product lines; (3) the capability to analyze and configure these models according to the changing context and requirements; and (4) the capability to execute them over several simulation scenarios. Finally, we show how to use VariaMos with an example, and we compare it with other tools found in the literature.

Keywords
Variability, product line engineering, dynamic product line models, constraints, tool, simulation

1. INTRODUCTION
VariaMos is an acronym for Variability Models; these models are referred to the specification of the variability of (dynamic) product lines. Product lines where the products configured from the associated variability models can be re-configured or adapted (their architecture can be changed) at runtime are known as dynamic product lines. These variability models are usually represented by means of modeling languages such as FODA (Feature-Oriented Domain Analysis) [1], Orthogonal Variability Models (OVM) [2], DOPLER [3], Goals [4] and constraint networks [5]. To represent and reason on these models, several approaches and tools exist in the literature. However, there is yet a lack of methods and tools for both representing and simulating (dynamic) product lines. This lack is more accentuated when the model is composed of a collection of views representing different facets of the same product line. This paper presents a whole new version of the precedent VariaMos tool. In particular, the new VariaMos allows defining variability modeling languages, modeling (dynamic) product lines and cyber-physical self-adaptive systems, and supporting automated verification, analysis, configuration and simulation of these models. The modification of the modeling language is partially supported at runtime but we are working to full support this capability in the near future. This runtime support makes the modification of the languages directly available to use in the models represented in that language.

In this paper, we use a simplified case of an Online Shopping Store. In particular, we considered 12 functionalities, related with payment and shipping requirements, represented with a feature model.

The paper is structured as follows: Section 2 gives an overview of VariaMos and its functionalities. Section 3 compares VariaMos with related tools. Section 4 presents the work in progress, and finally, Section 5 concludes the paper and describes future works.

SPLC 2015, July 20-24, 2015, Nashville, TN, USA
ACM 978-1-4503-3613-0/15/07.
http://dx.doi.org/10.1145/2791060.2791103

2. VARIAMOS TOOL
VariaMos supports different types of models and views by itself, but also offers the possibility of extending them. This possibility provides extensive generality and flexibility for the designer to accommodate the tool to her needs. From a point of view of interoperability, VariaMos allows to export XLS and JSON configuration files, to import JSON configuration files and save/load models to/from XML files. The VariaMos tool, its documentation, and a video tutorial are available online[1].

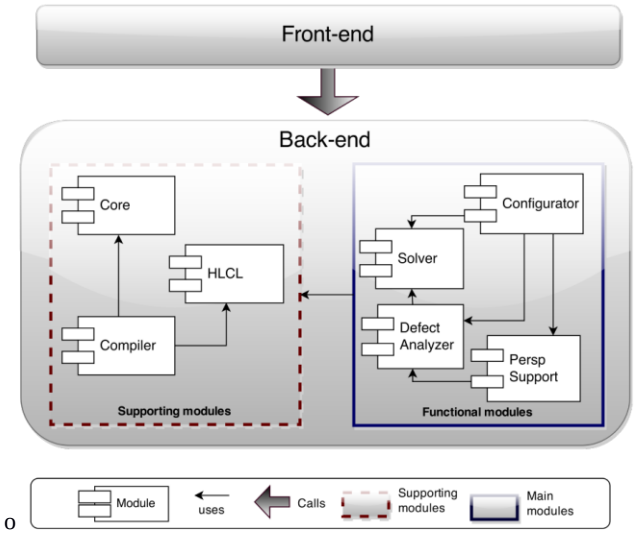
General Architecture
VariaMos can be used both as a standalone graphical tool and as a Java library that can be executed on different operating systems (i.e., Windows, Mac OS and Linux). It was developed in Java and

[1] www.variamos.com

it is an open source tool. Figure 1 shows its high-level architecture.

VariaMos has two main layers, the front-end with the Graphical User Interface (GUI) and the back-end that actually implements all the required functionalities. Inside the back-end layer there are two types of modules: supporting modules and functional modules. Supporting modules (i.e., *core, HLCL,* and *compiler*) provide basic operations, whereas functional modules (i.e., *PerspSupport, Configuration* and *DefectAnalyzer)* implement the main functionalities.

In detail, the *Core* module provides exceptions handling and utilities. The *HLCL* module implements a high-level constraint language, by means of Java objects, used to express models in an agnostic level [8], and the *Compiler* module has rules to convert high-level constraints into constraints expressed in a particular language.



o

Figure 1. High-level architecture of VariaMos

The *Solver* module implements the operations for reasoning with each particular constraint language according with the solver at hand. Currently, VariaMos supports two solvers: SWI prolog and GNU prolog. Other solvers will be incorporated in the near future. *PerspSupport* module supports simulation operations, and adaptations according to external contexts and modeling using different views. This module will also provide extensibility hooks for defining new variability meta-models. *Configurator* module supports configuration operations such as partial configurations, complete configurations and propagation of decisions. *Defect Analyzer* module supports semantic verifications such as to identifiy if a model is void and to identify dead and false optional elements, redundancies, false product line, and (in)valid configurations [6]. In the near future, this module will also support explanations and corrections for identified defects in the (dynamic) product line model and configurations.

Modeling Variability

Our approach is language-independent and relies on meta-modeling. In VariaMos, a meta-model defines the possible views of the systems to be modeled and defines the concepts of each view and the relations among them. For example, a view can represent the variability in terms of a feature model. Figure **2** presents a screenshot of a simplified feature model represented in VariaMos. This feature model is about an Online Shopping Store, where features are represented with double ellipse ovals; however, in the current release of VariaMos, the symbols used to represent model elements can be personalized and changed in a configuration file.

VariaMos supports expressions to define the constraints required by concepts and relations. VariaMos transforms, at runtime, the expressions into a constraint program. A constraint program is a collection of constraints without a particular order. The constraint program represents the system model and offers a richer view of the product line than individual views.

Currently, VariaMos supports two predefined meta-models. The meta-model for self-adaptive systems [7] and the meta-model for feature models (FM). Also, VariaMos partially supports the creation of new modeling languages and edition of the existing ones. The graphical interface is like the modeling interface presented in Fig 2. Main options in the graphical interface are:

Simulation buttons: they control simulation operations like get first solution, get next solution and reset simulation.

Views: they support the modeling of different concerns. According to the selected meta-model different views are available in the tool. For instance, with the REFAS [7] metamodel available views are: variability model, soft goals model, context model, soft goal satisficing model and assets model.

Perspectives: they define the set of views and available options in VariaMos. For instance, in the meta-modeling perspective users can change the shapes to represent concepts, whereas in the modeling perspective users must use those predefined shapes for modeling variability.

Status bar: it shows the time required by the last operation.

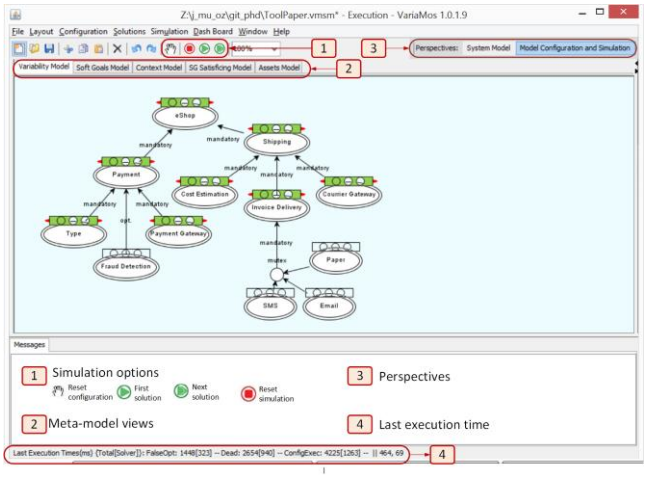


Figure 2. The simulation perspective of the Online Shopping Store feature model. VariaMos screenshot

Functionalities

Some of the analysis, verification and configuration operations implemented in VariaMos are the typical ones found in literature, and other ones are inspired by the industrial projects with our partners. In particular, VariaMos implements functions inherited from the previous release (i.e., list of all valid products, checking validity of configurations, finding the elements that should always be used in any product and the ones that can never be used and configuring products), but considering inherent issues of dynamic product lines such as the context changes.

Moreover, in this new version VariaMos shows the results in an interactive way. For instance, a configuration of the feature model presented in Figure 2 can have many steps and Figure 3 presents

an example of one of those steps in which the designer has not selected any feature (cf. Figure 3(a)), the designer provisionally selects the SMS feature (cf. Figure 3(b)), and the designer accepts the selection of the SMS feature (cf. Figure 3(c)).

In Figure 3, features have in the top of the oval a rectangle (henceforth named selection indicator). If the selection indicator is green for selected features, non-colored for selectable features or red for non-selectable features. The first circle is green for full-mandatory features, the second circle is green for selected features during the configuration and the third circle is green for features chosen in the simulation. In addition, the first circle is red for dead features and the second circle is red for features non-selectable during the configuration.

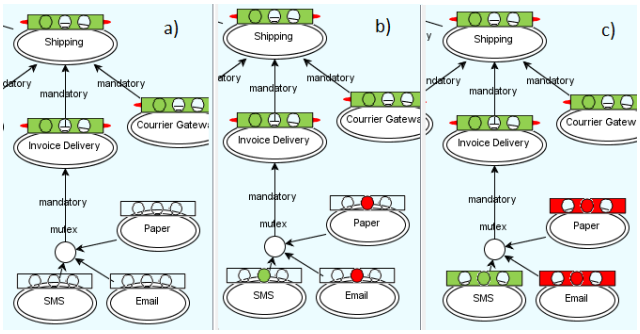


Figure 3. Configuration step for the SMS feature

VariaMos also provides simulation operations like iterate over all solutions of a partial configuration (cf. Figure 2), visualize possible adaptations of the system, and evaluate the solution of a configuration and propose alternatives (cf. Figure 4). VariaMos allows the definition of external context, simulation of context change and configuration of target systems. The idea in these simulation operations is to support the designer in the testing of the models definition before the implementation of the system or for maintenance purposes. The simulation operation relies on a MAPE-K loop implementation [8] and have four objectives. First, to monitor changes in models, external variables and the configuration of the system. Second, to analyze if the system maintains a valid solution after those changes. Third, to plan the adaptation when a not valid solution is detected, and finally, to execute the adaptation, formatting the outputs and updating the user interface. A valid solution in VariaMos in the one that satisfies the constraints of the different views of the variability model, the context conditions and the constraints expressed by who is configuring new products. Some of these constraints are "soft" (e.g., claims and configuration constraints) in the sense that it may prove impossible to satisfy them for all possible situations and their satisfaction is maximized according to preference levels.

To execute these operations, VariaMos represents the (dynamic) product lines models as a collection of constraints. These constraints are represented in a high-level constraint language [9] and then translated into the particular language of the solver in which these operations will be executed.

3. COMPARISON WITH OTHER TOOLS

There are several characteristics that differentiate VariaMos from the existing tools for managing (dynamic) product lines and self-adaptive systems. For instance, from the point of view of modeling, there are tools like Feature Plugin (http://gp.uwaterloo.ca/fmp), XFeature (http://www.pnp-software.com/XFeature), FeatureIDE (http://wwwiti.cs.uni-magdeburg.de/iti_db/research/featureide/), Pure::variants (http://www.software-acumen.com/purevariants/) and Requiline [10]. Most of these tools were built to graphically construct feature models and to derive products from these models, not to create the own variability language and simulate the models built on that language.

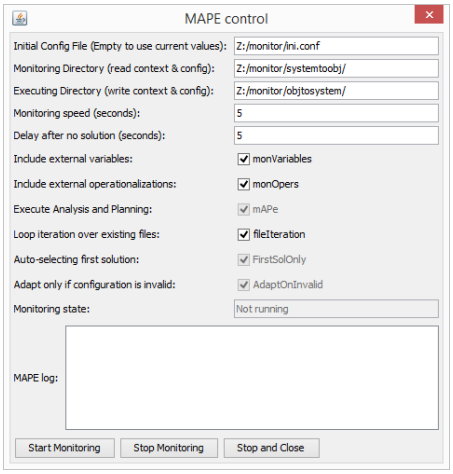


Figure 4. VariaMos simulation control dialog

From the point of view of analysis and verification, most of the tools found in the literature are formalism-dependent, and they only focus on feature models. In addition, most of them concentrate on verifying the consistency of a combination of features (a feature configuration) against the feature model. Tools like FAMA (http://www.isa.us.es/fama) and SPLOT (http://www.splot-research.org) consider several verification operations for feature models. VariaMos supports the same verification operations but not only over feature models, but also over other models based in the meta-models defined by engineers in order to represent variability-based systems.

There are also approaches that combine multiple variability models. For instance, the suite KumbangTools (http://www.soberit.hut.fi/KumbangTools/) combines feature and component-based models, and the tool Invar [11] provides the integration of heterogeneous variability models approaches such as DOPLER, feature model, and OVM. VariaMos is a language-independent tool in the sense that it allows developing our own variability modeling language. VariaMos is also extensible because different solvers can be incorporated as components of the VariaMos architecture and used to execute the (dynamic) product line models.

Regarding the relation between domain and application engineering, ISMT4SPL [12] offers traceability among the artifacts created from domain engineering and application engineering and provides automatic generation of variability models and source code. In the same line, LISA[13] toolkit presents an approach for integrating the variability management in architecture design and implementation to provide traceability and synchronization between models, architectures and implementations. VULCAN[14] is a CASE tool that provides verification of specifications, parameterization of product line architecture specifications, and source code generation: for generate products from assets. VariaMos focus mainly on domain engineering and therefore it primarily provides functionalities related to modeling, reasoning, and simulation on variability models written in any notation. These tools were compared to identify in them the main characteristics of VariaMos. These

characteristics are: simulation and adaptation (Sim), verification operations (Verif) and configuration operations (Conf). Moreover, we also analyze the modeling languages (ML) supported by each tool. Table 1 shows the comparison results. In general, most of the analyzed tools have configuration operations while none of them provide simulation operations. Furthermore, the analyzed tools are language-dependent because they support particular notations instead of different variability modeling languages as VariaMos does by allowing engineers define their own variability languages.

4. WORK IN PROGRESS

In the current version, VariaMos creates the meta-model instances from an Object model. We are working to support functionalities for importing/exporting from/to external files of meta-models in a similar way as already implemented for the models. In addition, VariaMos will visually support the complete creation and edition of meta-models. In the current version, VariaMos creates the instances of semantic models from a set of fixed Java classes. We are exploring other alternatives for defining the semantic models dynamically. One alternative that we already implemented is the dynamical definition of conditional expressions for some concepts. We are working to extend this idea to link expressions to models, views, concepts and relations in the meta-model, and then, we will associate those expressions to the operations supported by VariaMos.

Having models that correctly represent the domain of the product line is of paramount importance for product line engineering success. For this reason, we are working on a method that will point out the causes and possible corrections of various kinds of defects in product line models specified in different notations. Moreover, we are also interested in proposing some criteria to help the designer to make the best correction choice.

Table 1. Characteristics supported by some modeling tools

Tool	Sim	ML	Verif	Config
VariaMos	●	Language-independent	●	●
Feature Plugin		FM		●
XFeature		FM		●
FeatureIDE		FM	●	●
Pure::Variants		FM	●	●
Requiline		FM	●	●
FAMA		FM	●	●
SPLOT		FM	●	●
KumbangTools		FM		
INVAR		OVM,FM,DOPLER		
LISA toolkit		OVM		
ISMT4SPL		OVM		●
VULCAN		FM		

5. CONCLUSIONS

In this paper, we presented the second release of VariaMos. 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 has its place among existing applications. Although VariaMos is not a mature tool yet, its promising capabilities of extensibility, interoperability, expressiveness, scalability and efficiency (the last two, inherited from the first release) 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 reasoning functions according to the modeling language at hand. For instance, verification against a meta-model defined by users, incorporation of a guided process allowing correcting anomalies, support incorporation for incremental verification and the implementation of connections with other solvers; e.g., SAT (SATisfiability), BDDs (Binary Decision Diagrams) and SMTs (Satisfiability Modulo Theories) are envisaged for future releases in order to improve the efficiency of certain reasoning operations.

6. ACKNOWLEDGMENTS

Special thanks to Diego Quiroz, Sebastian Monsalve, Jose Ignacio Lopez, León Jaramillo, Andrés Posada and Beatriz Melo for their collaboration in technical and operative aspects related to the tool.

7. REFERENCES

[1] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and S. P. Peterson, "Feasibility Study Feature-Oriented Domain Analysis (FODA). Technical Report," 1990.

[2] K. Pohl, G. Böckle, and F. J. van Der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York, Inc., 2005.

[3] D. Dhungana, P. Grünbacher, and R. Rabiser, "The DOPLER Meta-tool for Decision-oriented Variability Modeling: A Multiple Case Study," *Autom. Softw. Engg.*, vol. 18, no. 1, 2011.

[4] B. Gonzales-Baixauli, J. C. S. Prado Leite, and J. Mylopoulos, "Visual variability analysis for goal models," in *Requirements Engineering(RE) Conference*, 2004.

[5] R. Mazo, C. Salinesi, D. Diaz, O. Djebbi, and A. Michiels, "Constraints: the Heart of Domain and Application Engineering in the Product Lines Engineering Strategy," *Int. Journal on Information System Modeling and Design (IJISMD)*, vol. 3, no. 2, 2011.

[6] C. Salinesi and R. Mazo, "Defects in Product Line Models and how to identify them," in *Software Product Line - Advanced Topic*, A. Elfaki, Ed. InTech, 2012, pp. 97–122.

[7] J. C. Munoz-Fernandez, G. Tamura, R. Mazo, and C. Salinesi, "Towards a requirements specification multi-view framework for self-adaptive systems," in *XL CLEI Conference, 2014* .

[8] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer (Long. Beach. Calif).*, vol. 36, no. 1, 2003.

[9] R. Mazo, C. Salinesi, and D. Diaz, "Abstract Constraints: A General Framework for Solver-Independent Reasoning on Product-Line Models," *J. Int. Counc. Syst. Eng.*, vol. 14, no. 4, 2011.

[10] T. von der Massen and H. Lichter, "RequiLine: A Requirements Engineering Tool for Software Product Lines," in *Proceedings of the Fifth Int. Workshop on Product Family Engineering*, 2003.

[11] K. Park, D. Ryu, and J. Baik, "An Integrated Software Management Tool for Adopting Software Product Lines," *Comput. Inf. Sci. (ICIS), 2012 IEEE/ACIS 11th Int. Conf.*, 2012.

[12] D. Dhungana, R. Rabiser, P. Grunbacher, D. Seichter, G. Botterweck, D. Benavides, J. Galindo, "Integrating heterogeneous variability modeling approaches with Invar," *VaMos*, 2013.

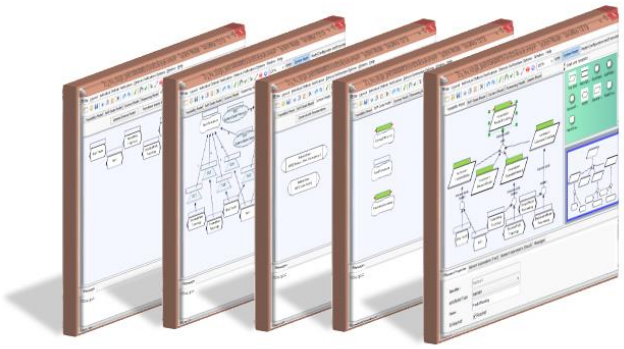
[13] I. Groher and R. Weinreich, "Supporting Variability Management in Architecture Design and Implementation," *2013 46th Hawaii Int. Conf. Syst. Sci.*, 2013.

[14] H. Lee, J. Yang, and K. C. Kang, "VULCAN: architecture-model-based workbench for product line engineering.," *SPLC*, vol. II, 2012.

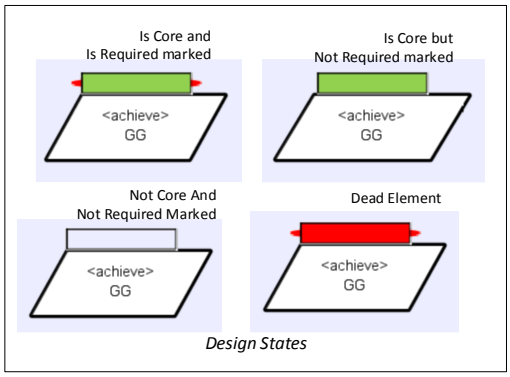
APPENDIX A: **Presentation of VariaMos**

VariaMos Key Functionalities

1. Support for Model creation, edition and verification:
   - Multi-Model support – REFAS language support and the possibility to use other languages. Variability defined using goals or features models

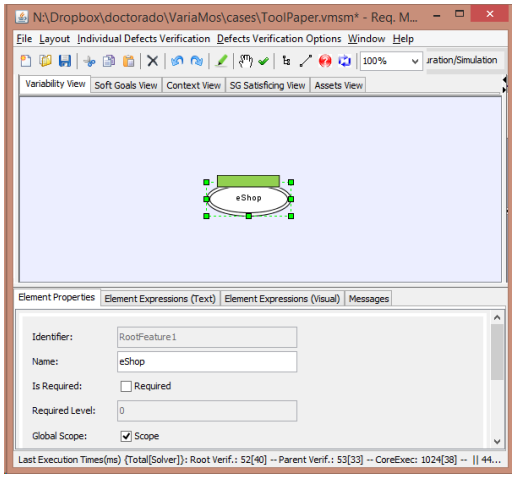


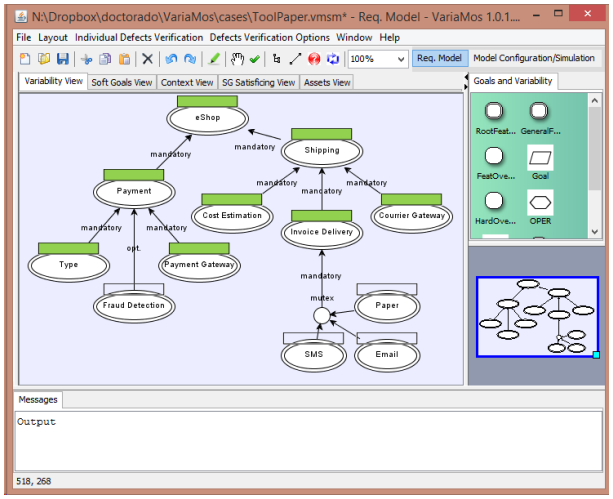
   - Explain the states of concepts at design time according to the figure:



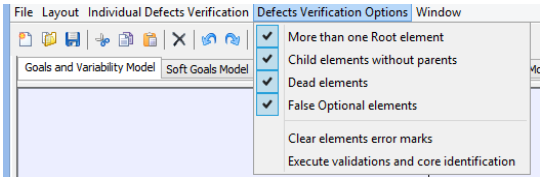
Design States

2. *OSS modeling demo:*
   - Present the definition of concepts and relations (instantiation and edition) in VariaMos for an Online Shopping Store (OSS).

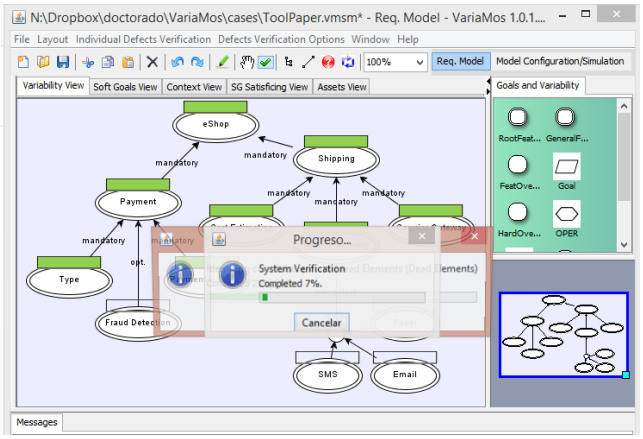




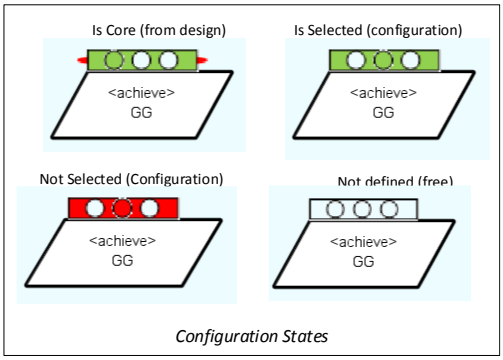
3. *Introduce the operations supported in VariaMos.*
   - Explain calculation of core concepts operation.
   - Explain verification of feature models operation:
     - Single Root
     - All features have parents
     - Dead features
     - False optional features



4. *OSS verification operations demo:*
   - Show the core calculation and verification operations.
   - Show the Meta-Model visualization (abstract and concrete syntax).
   - Show the Semantic Model visualization (Model supporting the Meta-Model).

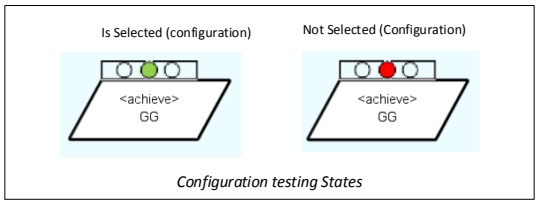


5. Model Configuration:
   - Selected and excluded concepts from a configuration
   - Evaluation of a configuration implications
   - Propagation of configuration implications
   - Explain the states at configuration and simulation time:



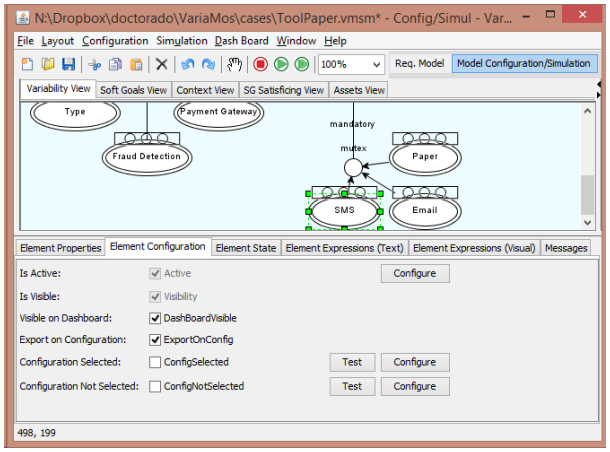
Configuration States

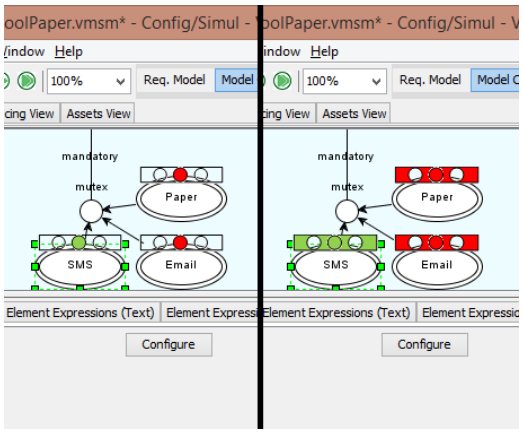
Explain the states during a configuration step:



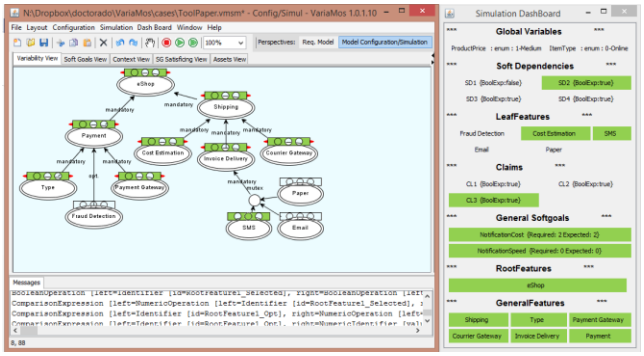
Configuration testing States

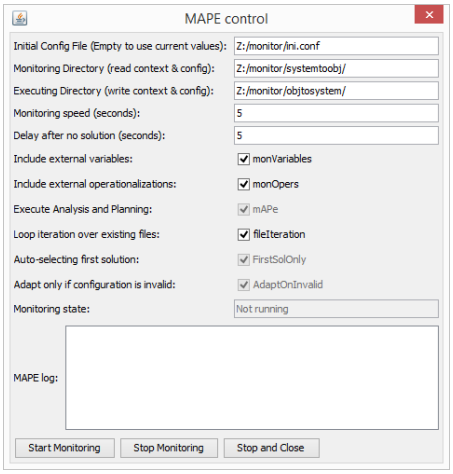
6. *OSS configuration functionality demo:*
   - The configuration of concepts as selected with implications.





7. Model basic simulation with a Dashboard:
   - Visualization a solution based on model definition and current configuration.
   - Iteration over available solutions of a configuration.
   - Iteration from different context and system configurations, evaluating solutions and alternatives.
8. *OSS simulation functionality:*
   - Show simulation of solutions for a context.
   - Show iteration of solutions for a context.
   - Show the scenarios for an optimal solution of the system.





Work in Progress

- Full support of meta-models load/save from/to external files.
- Full support of meta-models creation/edition. Define the semantic models dynamically.
- Point out the causes and possible corrections of various kinds of defects in product line models specified in different notations.
- Propose some criteria to help the designer to make the best correction choice.

Conclusions

- VariaMos has its place among existing applications.
- Although VariaMos is not a mature tool yet, its promising capabilities of extensibility, interoperability, expressiveness, scalability and efficiency will allow the tool to become accepted and used by the academic and industrial community in the future.