Integrating Heterogeneous Variability Modeling Approaches with Invar

[Tool Demonstration]

Deepak Dhungana
Siemens AG Österreich
Corporate Technology
Vienna, Austria
deepak.dhungana@siemens.com

Dominik Seichter and Goetz Botterweck
Lero–The Irish Software Engineering Research Center
Limerick, Ireland
goetz.botterweck@lero.ie

Rick Rabiser and Paul Grünbacher
Christian Doppler Lab for Automated Software Engineering
JKU Linz, Austria
rick.rabiser@jku.at

David Benavides and José A. Galindo
Department of Computer Languages and Systems
University of Seville
Seville, Spain
benavides@us.es

ABSTRACT

There have been several proposals to describe the variability of software product lines by using modeling languages. In larger organizations or projects (e.g., multi product line environments) this can lead to a situation where multiple variability modeling techniques are used simultaneously. Rather than enforcing a single modeling language, we present an integrative infrastructure that provides a unified perspective for users configuring products in such multi product line environments, regardless of the different modeling methods and tools used internally. In this tool demonstration paper, we present a prototypical implementation of our framework based on Web services. So far, the prototype has been used with a feature-based, an OVM-style and a decision-oriented variability modeling approach.

Keywords

Variability Modeling, Product Configuration, Integrated Tool, Web Service

1. INTRODUCTION

The existing diversity in software product lines has led to the development of different tools, techniques, and languages to describe the variability of software product lines (e.g., [11, 3, 8, 5]). To reflect the complexity of real-world systems it is often appropriate to use different modeling approaches to describe different system parts. Different modeling languages have benefits and drawbacks depending on the system to be described and the personal preferences of the product line engineers.

However, the use of different "island solutions" for variability modeling and product configuration restricts communication and hinders collaboration between distributed product line engineers. Hence, there is a need for an integrative infrastructure enabling the collaboration between different organizations developing product lines. Such an infrastructure has to support different variability modeling languages, notations, and tools. The specific tools or data formats used when creating the variability models are not relevant for end users who only care about the available choices and their implications for the product.

Consider the example of a multi product line[1] of an Enterprise Resource Planning (ERP) application. The main vendor of the ERP application integrates several suppliers providing specific encapsulated functionality. In the sample case, the vendor might for instance use a feature model to present the set of available choices and to communicate extension and integration possibilities for other systems. The suppliers might use different approaches and tools for their particular parts of the system. For example, one supplier could use decision modeling tools [6], while others apply orthogonal variability modeling [9] or feature modeling [5].

For the configuration of an ERP solution in such a context an integrative infrastructure is needed, which is able to represent and implement constraints across model boundaries. In doing so, the needs of both modelers and end users (of configuration tools) must be considered.

In an earlier paper [7], we introduced the I$nvar$ approach sup-

[1]The concept of a "multi product line" refers to product lines, where the different sub systems are also product lines themselves.

porting the integration of heterogeneous variability modeling approaches and presented an initial evaluation. In this tool demonstration paper, we present the *Invar* infrastructure that facilitates the integration of heterogeneous variability models. The *Invar* prototype currently supports integration of three different variability modeling approaches, i.e., a feature modeling [12], a decision modeling [6], and an orthogonal variability modeling approach [10]. *Invar* enables the communication between different languages and tools for variability management. It eliminates the need to stick to one concrete variability modeling approach when designing multi product lines. A configuration front-end for end users transparently presents models created in different notations.

We first explain the key assumptions underlying *Invar* and then present the *Invar* infrastructure and its implementation.

2. UNDERLYING ASSUMPTIONS

In the current state of practice multiple heterogeneous variability modeling approaches are used by different organizations. Different reasoning and analysis engines – for instance, SAT, BDD, or CSP solvers [1] or rule engines [6] – are adopted for interpreting and implementing the models' semantics. This is problematic, since there is no integration of these diverse tools supporting different notations. In some organizations variability is managed "manually" using textual descriptions or spreadsheets; these "tools" are typically not integrated with other variability modeling tools due to their lack of formality or simply because of a lack of dedicated configuration tools.

Despite the differences in modeling notations, data formats and reasoning technologies, variability modeling approaches do share characteristics and can be described by common concepts: A Variability Model consists of Variables and Constraints over these variables. Each variable has a Type, for instance, Boolean, Integer, or String. Depending on the particular approach there are different Types of Constraints, e.g., "Optional Sub-features", "Alternative Groups", or "requires/excludes". A modeler creates a set of variables and constraints. An Assignment of values to the variables corresponds to a Configuration of the model.

For a given configuration, we can decide whether it satisfies the constraints defined in the model. Hence, a model defines a set of compliant configurations. Users add more constraints as they make configuration decisions. Based on the current set of constraints the remaining possible values for each variable can be determined. Adding more constraints eventually leads to a model that has exactly one valid assignment in which each variable has exactly one value. This represents the configured product, for instance in terms of selected and deselected capabilities. If no valid assignment is left then the model is unsatisfiable.

When presenting remaining configuration options in a configuration tool, we can ensure that the model remains satisfiable, i.e., only those options are offered for interaction, which, when they would be chosen, leave at least one valid assignment.

Our approach relies on the assumption that variability models can (from the perspective of the end user) be broken down into options that are either available, selected, constrained or not available during configuration and that are related in different ways, e.g., defining an order for selecting options.

This can be compared to a pivot format for all kinds of variability models, which is general enough such that concepts of existing approaches can be translated to this simplified view. The formal semantics of the pivot format has yet to be defined, but this paper is a step towards this direction in a pragmatic way through illustrative implementation.

3. THE INVAR INFRASTRUCTURE

The *Invar* infrastructure allows to "plug-and-play" variability models. "Plugging" refers to adding new variability models to a shared repository. "Playing" refers to presenting the options provided by variability models to end users when configuring a product. For this purpose, a variability model is treated as an modular entity, which can be plugged into the configuration space to provide configuration options. However, being modular does not necessarily mean the models are independent of each other. In *Invar* this is reflected by the fact that variability models can be related to each other by interdependencies [7] enabling the use of cross-model constraints such as, "if feature A is selected in model X then feature B in model Y is required".

Our approach allows using variability models distributed across multiple repositories by accessing them through Web services, which are providing configuration choices. An end user then works with a front-end for product configuration and can use the services without knowing details about the concrete variability models underlying the services. Figure 1 depicts the architecture of the *Invar* infrastructure comprising five main components:

Vendor model repositories (see (1) in Figure 1) – Product vendors or suppliers add their variability models to model repositories. The models may or may not contribute to the same product and are not necessarily dependent on each other.

Invar **repository** (2) – This repository defines aggregations of different models from the vendor repositories by logically grouping them. For instance, one particular model may be part of multiple product lines, as it may contribute to more than one product.

Configuration Web services (3) – The different models residing in (possibly distributed) repositories are accessed by configuration Web services. A Web service provides a standard interface for different configuration front-ends such as websites, mobile devices, or stand-alone applications. For each type of model, designated configuration services are developed (by implementing an interface) that can read the data formats, interpret the content, and perform operations on the models.

Configuration broker (4) – The configuration broker enables the communication between the Web services. It reads the inter-model dependency information to determine which Web services are affected when products are configured. The configuration broker also translates events from the end user performing the configuration and passes them on to the Web services that need to react to this input.

End-user product configuration front-ends (5) – The configuration choices defined in the variability models are presented to the end user in a product configuration front-end. This can be a website or a stand-alone application. We provide an example user interface through the *Invar* framework website at http://invar.lero.ie also shown in Figure 2.

By using *Invar* stakeholders can create variability models using an approach of their choice. *Invar* defines key operations and queries (*configuration primitives*) on variability models to allow the integration of heterogeneous approaches (for details please refer to our earlier work [7]). The configuration primitives are implemented as operations of a Web service to allow uniform access to the models. This allows the involved organizational units and teams to use their preferred modeling approaches while reusing variability models from other units. *Invar* provides a single and transparent configuration tool to end users to ensure interoperability and reuse of variability models in different contexts. For example, one model may be shared between several companies and each one may use it to create different products. The participating organizational units could also create their own configuration tools and access diverse

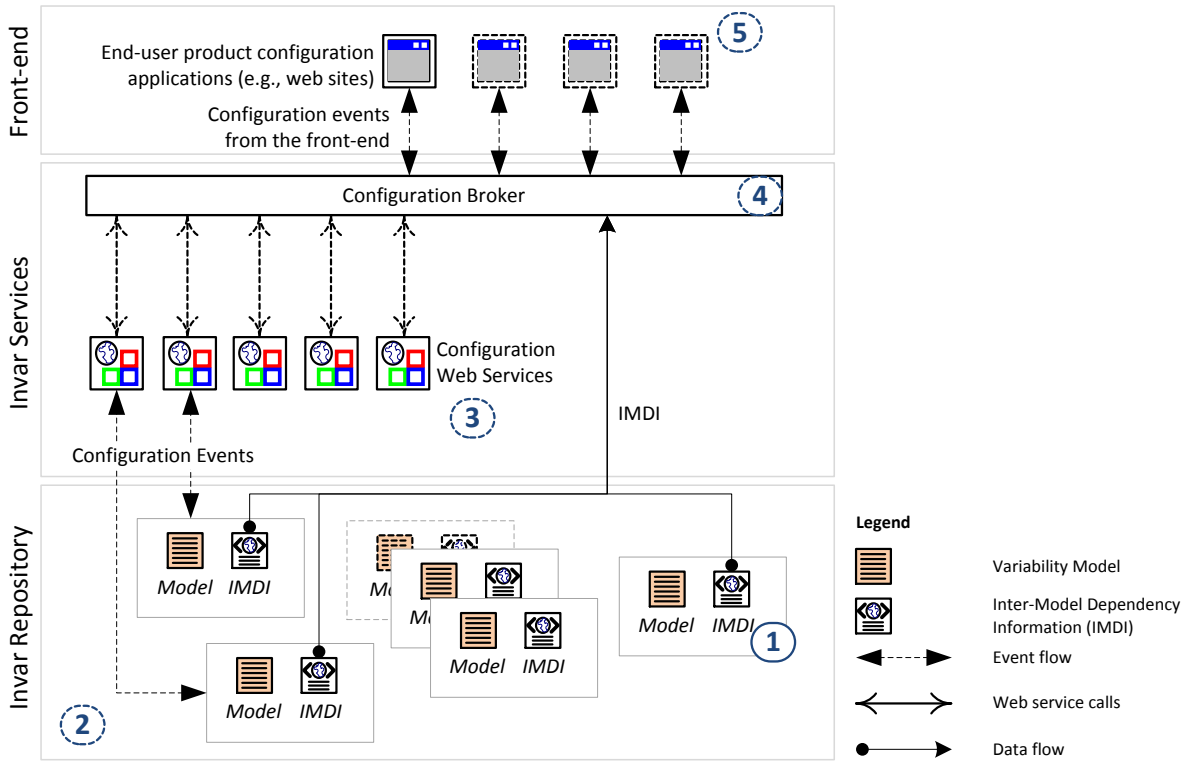

Figure 1: Architecture of the Invar infrastructure (adapted from [7]).

models via the Invar Web services without having to know all the details about the actual modeling approaches underneath.

Whenever a variability model is plugged into the configuration environment, it needs to explicitly define its relationships to the other models. This is done by adding an *inter-model dependency information (IMDI) packet* together with the model. Dependencies are defined using if *condition* then *action* clauses which can be compared to conventional cross-tree constraints used in single models. IMDI packets do not affect the internal semantics of the models in use. An IMDI action is executed when its condition evaluates to TRUE. Details on the IMDI packets and the conditions and actions currently supported by Invar can be found in [7].

4. IMPLEMENTATION

The Invar implementation allows creating and maintaining repositories for sharing variability models and supports end-user configuration based on these models. The infrastructure relies on *Web services* for accessing variability models and on the configuration front-end to provide these models to the user. Any Web service API can be used to generate Web services, which can be plugged into the Invar framework based on a provided WSDL description. For the front-end, a layered Java/J2EE application platform based on the Spring Framework was used to provide centralized, automated configuration and integrative wiring of the application objects.

Web services were chosen as a base technology for the Invar prototype since they allow for an easy, standardized integration of potentially distributed and heterogeneous software components. In Invar Web services are used to enable the different variability modeling tools to communicate with the core component. The flexible composition of Web services allows to configure products dynamically and to include different variability models from different Web services into the configuration environment when they are needed. Similar to many typical architectures for serviced-based applications Invar also uses the concepts of *providers*, *consumers*, and a *registry*. The different Web services are providers of variability models and the Invar configuration site simultaneously represents both the service registry and the consumer of the models.

Central to the implementation of Invar is the *generic configuration interface* defined for programmatically accessing the diverse variability models. The configuration service definition has to be implemented *once* for each modeling notation. The Web service is designed such that the configuration options are presented to the end user as questions. Questions are only a means to render the variation points/options/features to present it to the user. This means the user is asked questions about a certain "feature" (in the wider sense) or a property of the system she configures. The possible answers to the question (the available alternatives) are presented to the user such that she may choose one or many of them depending on the type of variability. The notion of "questions" and possible answers as options is therefore key to the Invar configuration service.

The interface consists of two parts: A *variability model query part* providing basic information about models (e.g., the set of available questions and the possible answers) and an *operational part* directly interacting with models to assign answers to specific questions (e.g., when selecting a particular feature).

The configuration service also defines a set of predefined question *types*. The types have been defined based on how the end user is supposed to answer them. For example, the question type Alternative refers to questions where the user can select exactly one option (rendered using radio buttons or combo-boxes in the UI); for Optional the user can pick multiple items (rendered using checkboxes in the UI) and MoreThanOne refers to cardinality (1..*) (ren-

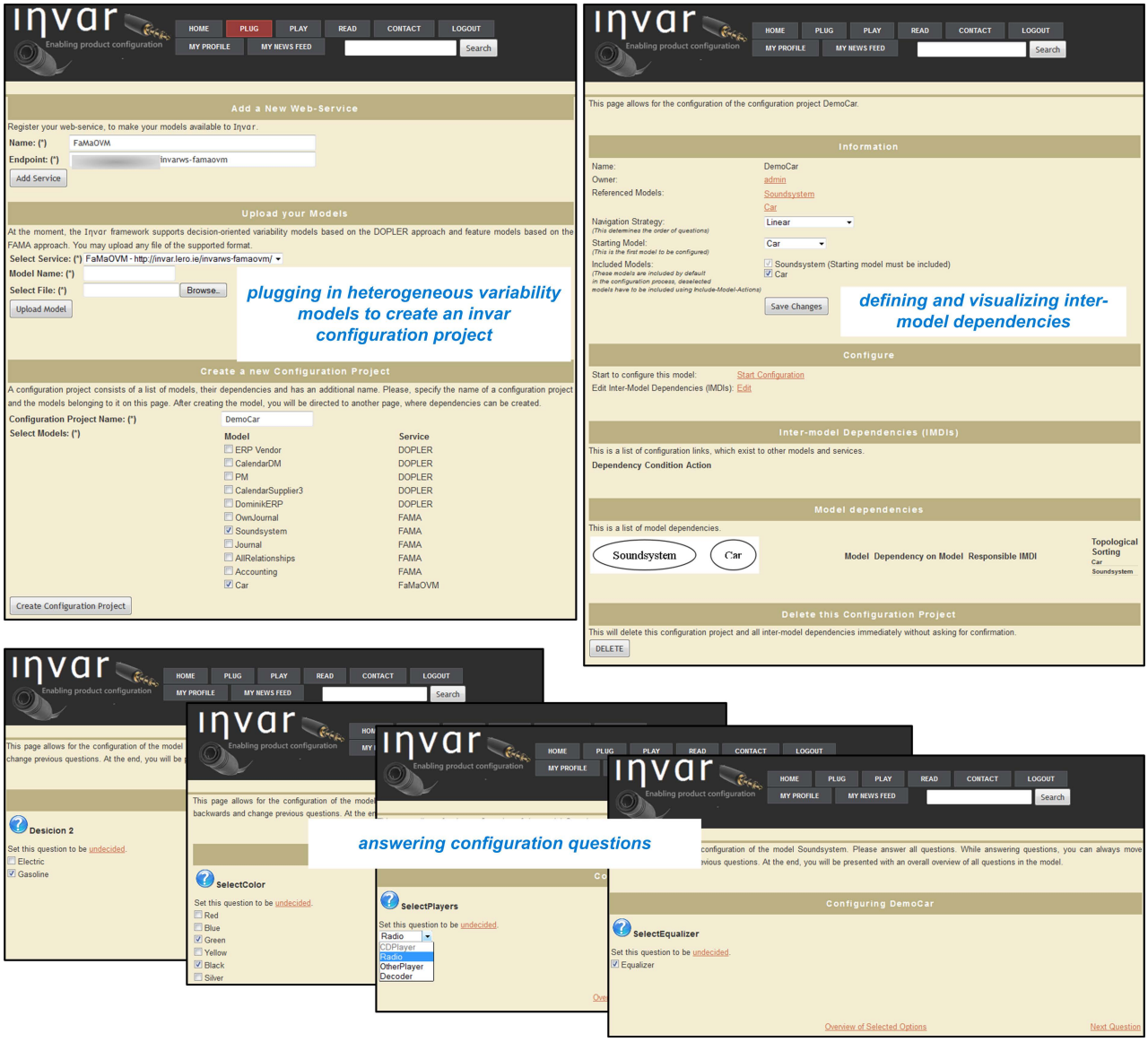

Figure 2: The *Invar* prototype: plugging in heterogeneous variability models to create configuration projects (top left), defining and visualizing inter-model dependencies (top right), and answering configuration questions (bottom). The *Invar* prototype can be tested at http://invar.lero.ie (username: a; password: a).

dered using multi-selection checkboxes in the UI).

The *Invar* service configuration interface is offered out of the box for the feature-oriented FaMa tool suite [12], the OVM-oriented FaMa tool [10], and the decision-based DOPLER [6] tool suite. The selection to implement these particular approaches in *Invar* was based on the experience gained by several years of experience of applying them in academic and industrial settings including large-scale product lines [1, 2, 6, 10, 12]. Furthermore, these three approaches represent three distinct classes of modeling techniques in software product line engineering, i.e., feature modeling, orthogonal variability modeling, and decision modeling [5, 9]. Refer to [7] for details on the implementation of *Invar* for DOPLER and FaMa. Since that earlier work, *Invar* has been extended by the support for OVM models.

5. SUMMARY AND OUTLOOK

In this tool demonstration paper, we presented the *Invar* infrastructure facilitating the integration of variability models for product configuration regardless of the modeling techniques, notations and tools used. We have presented the key assumptions underlying the approach, discussed the *Invar* architecture, and showed the web-based configuration front-end for end users. So far *Invar* has been used with and implemented for one feature modeling, one decision modeling, and one orthogonal variability modeling approach.

The key contributions of *Invar* are that (i) it enables the communication between different languages and tools for variability management; (ii) it eliminates the need to stick to one concrete variability modeling approach when designing multi product lines; and (iii) its configuration front-end for end users transparently presents models created in different notations.

We are currently working on the integration of more modeling approaches to *Invar* like TVL [4] or COVAMOF [11]. We are also working on improving the underlying reasoning mechanisms. Furthermore, we plan (i) to formalize the integration concepts, including the semantics of the IMDI links, so that verification and validation across models is possible, (ii) to visualize the models and the relations between them beyond modeling notations, so that the modelers are aware of dependencies and the impact of changes to models, and (iii) to evaluate the framework in an industrial context, so that the practical value and usefulness can be demonstrated.

Acknowledgements

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero; by the Christian Doppler Forschungsgesellschaft, Austria; by the European Commission (FEDER) and Spanish Government under project SETI (TIN2009-07366); and by the Andalusian Government under project THEOS (TIC-5906) and the Talentia program.

6. REFERENCES

[1] D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated analysis of feature models 20 years later. *Information Systems*, 35(6):615–636, 2010.

[2] G. Botterweck, M. Janota, and D. Schneeweiss. A design of a configurable feature model configurator. In *3rd International Workshop on Variability Modelling of Software-intensive Systems (VaMoS 2009)*, pages 165–168, Sevilla, Spain, 2009. ICB Research Report vol. 29.

[3] L. Chen, M. Babar, and N. Ali. Variability management in software product lines: A systematic review. In *13th International Software Product Line Conference (SPLC 2009)*, pages 81–90, San Francisco, CA, USA, 2009. ACM.

[4] A. Classen, Q. Boucher, and P. Heymans. A text-based approach to feature modelling: Syntax and semantics of TVL. *Science of Computer Programming*, 76(12):1130–1143, 2011.

[5] K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid, and A. Wasowski. Cool features and tough decisions: a comparison of variability modeling approaches. In *6th International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS 2012)*, pages 173–182, Leipzig, Germany, 2012. ACM.

[6] D. Dhungana, P. Grünbacher, and R. Rabiser. The DOPLER meta-tool for decision-oriented variability modeling: A multiple case study. *Automated Software Engineering*, 18(1):77–114, 2011.

[7] D. Dhungana, D. Seichter, G. Botterweck, R. Rabiser, P. Grünbacher, D. Benavides, and J. Galindo. Configuration of multi product lines by bridging heterogeneous variability modeling approaches. In *15th International Software Product Line Conference (SPLC 2011)*, pages 120–129, Munich, Germany, 2011. IEEE.

[8] L. B. Lisboa, V. C. Garcia, D. L. dio, E. S. de Almeida, S. R. de Lemos Meira, and R. P. de Mattos Fortes. A systematic review of domain analysis tools. *Information and Software Technology*, 52(1):1–13, 2010.

[9] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, 2005.

[10] F. Roos-Frantz, D. Benavides, A. Ruiz-Cortés, A. Heuer, and K. Lauenroth. Quality-aware analysis in product line engineering with the orthogonal variability model. *Software Quality Journal*, 20(3-4):519–565, 2012.

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

[12] P. Trinidad, D. Benavides, A. Ruiz-Cortés, and S. S. A. Jimenez. FaMa framework. In *12th International Software Product Line Conference (SPLC 2008), vol. 2*, page 359, Limerick, Ireland, 2008. Lero TR `http://www.isa.us.es/fama`.