fmp and *fmp2rsm*: Eclipse Plug-Ins for Modeling Features Using Model Templates

Krzysztof Czarnecki, Michał Antkiewicz,
Chang Hwan Peter Kim, Sean Lau, Krzysztof Pietroszek
University of Waterloo
{kczarnec,mantkiew,chpkim,sqlau,kmpietro}@swen.uwaterloo.ca

ABSTRACT

Feature-based model templates have been proposed as a technique for modeling software product lines. We describe a set of tools supporting the technique, namely a feature model editor and feature configurator, and a model-template editor, processor, and verifier.

Categories and Subject Descriptors: D.2.1 [Software Engineering]: Requirements / Specifications—*Tools* D.2.2 [Software Engineering]: Design Tools and Techniques—*Computer-aided software engineering (CASE)* D.2.4 [Software Engineering]: Software/Program Verification

General Terms: Design, Documentation, Verification

Keywords: Feature modeling, model-driven development, product configuration, software-product lines, variability management

1. INTRODUCTION

In our other contribution, which is located in the poster section of this volume, we gave the motivation for model-driven product lines and explained the concepts underlying *feature-based model templates* [5], which is a particular technique for model-based development of software product lines.

In this short paper, we present a set of tools supporting feature-based model templates.

2. PLUG-IN FOR FEATURE MODELING

fmp is an Eclipse plug-in for feature modeling and configuration [1]. It supports a particular form of feature modeling, which is referred to as *cardinality-based* [7]. A summary of the main capabilities of fmp follows.

Feature model editor. Feature models can be edited in an explorer-style view as shown in the top part of Figure 1.

Feature-based configurator. Feature configurations can be created in a check-box view or using a wizard. Both partial and full configurations are supported. Partial configurations can be rendered as feature models and used as a basis for creating further configurations (this idea is referred to as *staged configuration* [6]).

Support for constraints. Additional constraints among features and feature attributes can be defined using XPath and/or propositional formulas.

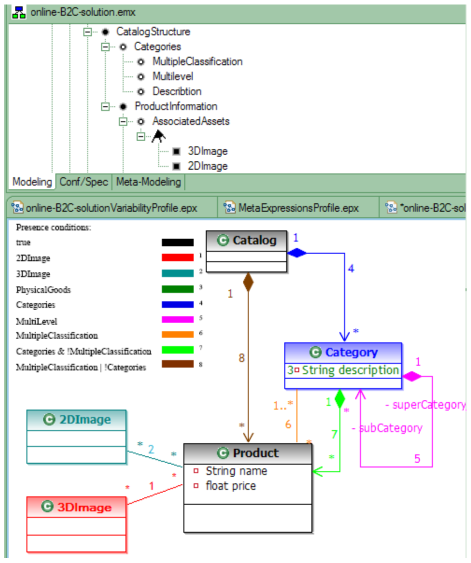

Figure 1: Sample feature model and a UML class model template

Constraint checking and propagation. A feature model can be checked for consistency (i.e., whether it has at least one valid configuration). Furthermore, concrete configurations can be checked to see whether they satisfy all the constraints from their corresponding feature models. The exact number of full configurations represented by a partial configuration is shown. Finally, constraint propagation is used to guide the user during configuration, e.g., selecting a feature that requires another feature will automatically select the latter. Distinction is made between choices that are undecided, made by the user, and automatically made. The guidance is optional and there are several levels of assistance. The support for constraint-based configuration is provided by the configuration toolkit Configit [4], which is based on Binary Decision Diagrams.

Synchronization between feature models and configurations. Changes to a feature model can be automatically propagated to its subsequent stages of partial and full configurations by treating models and configurations in a uniform way. The approach used for this propagation is described elsewhere [9].

Model and configuration exchange. Feature models and configurations can be exported and imported using an

XML format. The XML representing configurations can be fed into other tools, such as code generators. A special XML format for configurations is provided that allows template code to access configurations easily and in a readable manner using XPath.

User-extensible metamodel. The user can easily define the format for additional information to be associated with features, such as priorities, binding times, implementation status, etc., by extending the metamodel of the feature modeling notation, which is a feature model, too.

The plug-in was implemented using the Eclipse Modeling Framework (EMF).

3. PLUG-IN FOR FEATURE-BASED MODEL TEMPLATES

fmp2rsm is a plug-in that adds support for model templates to IBM Rational Software Modeler (RSM), an Eclipse-based UML2 modeling environment. The support for feature models is given by *fmp*, which can be run within RSM. In fact, Figure 1 shows a screen shot of RSM with fmp and fmp2rsm running inside it.

Currently, fmp2rsm only supports presence conditions and meta-expressions; support for iteration directives is planned. Annotations are represented as stereotypes, which are collected in a profile. A summary of the main capabilities of the plug-in follows.

Support for all sub-notations of UML. Any kind of UML models (e.g., class, activity, interaction, and state models) can be templetized.

Profile generation. A profile containing stereotypes with presence conditions for all or selected features can be generated from a feature model.

Automatic coloring. The stereotypes assigned to model elements can be hidden and the assignment of the annotations can be visualized through colors, where each color represents a particular presence condition.

Template instantiation with patching transformations. Given a feature configuration, the corresponding template instance is created automatically. The instantiation process can apply the so-called *patching transformations*, which allow a more concise representation of templates. An example of such transformation is the closure of an incoming and outgoing flow of an action in an activity model template, when the action is removed in a particular instance.

Automatic template verification. An automatic verifier makes sure that no ill-structured template instance can be created for a valid feature configuration. The desired structural well-formedness constraints can be expressed in OCL. Any violations are reported to the template designer by highlighting the involved model elements and giving sample feature configurations for which the violation occurs. The implemented verification procedure is described elsewhere [8].

4. DISCUSSION AND FUTURE WORK

We have gained some early experience with the tools described in this paper by creating feature models and business model template of a configurable e-commerce platform. The business model template consists of UML class model templates for business entities and activity model templates for business processes. The feature models contain several hundred features and the model templates contain dozens of classes and activities. The early experience has been very encouraging, but more work is needed. In particular, guidelines on using templates with other mechanisms for representing variability in models, such as various analysis and design patterns, are needed.

Several tools for feature modeling exist, e.g., XFeature [11], pure::variants [3], GEARS [10], ReqiLine [12], and the configuration wizard in the AHEAD toolsuite [2]. The main unique capabilities of our tool include support for staged and multilevel configuration, feature model synchronization, and advanced support for constraints. We are not aware of any model template tools comparable to *fmp2rsm*.

In future, we would like to gain more experience in applying the described approach and tools in different application domains.

fmp and *fmp2rsm* are available at `http:\\gp.uwaterloo.ca\fmp` and `http:\\gp.uwaterloo.ca\fmp2rsm`, respectively.

5. REFERENCES

[1] M. Antkiewicz and K. Czarnecki. FeaturePlugin: Feature modeling plug-in for Eclipse. In *OOPSLA'04 Eclipse Technology eXchange (ETX) Workshop*, 2004. Paper available from `http://www.swen.uwaterloo.ca/~kczarnec/etx04.pdf`. Software available from `gp.uwaterloo.ca/fmp`.

[2] D. Batory. Feature Models, Grammars, and Propositional Formulas. Technical Report TR-05-14, University of Texas at Austin, Texas, Mar. 2005.

[3] D. Beuche. pure::variants Eclipse Plugin. User Guide. pure-systems GmbH. Available from `http://web.pure-systems.com/fileadmin/downloads/pv_userguide.pdf`, 2004.

[4] Configit Software. *Configit—Product Configuration Engine*, 2005. `http://www.configit-software.com/`.

[5] K. Czarnecki and M. Antkiewicz. Mapping features to models: A template approach based on superimposed variants. In R. Glück and M. Lowry, editors, *GPCE 2005 - Generative Programming and Component Enginering. 4th International Conference, Tallinn, Estonia, Sept. 29 – Oct. 1, 2005, Proceedings*, volume 3676 of *LNCS*, pages 422–437. Springer, 2005.

[6] K. Czarnecki, S. Helsen, and U. Eisenecker. Staged configuration using feature models. In R. L. Nord, editor, *Software Product Lines: Third International Conference, SPLC 2004, Boston, MA, USA, August 30-September 2, 2004. Proceedings*, volume 3154 of *LNCS*, pages 266–283, Heidelberg, Germany, 2004. Springer-Verlag.

[7] K. Czarnecki, S. Helsen, and U. Eisenecker. Formalizing cardinality-based feature models and their specialization. *Software Process Improvement and Practice*, 10(1):7–29, 2005.

[8] K. Czarnecki and K. Pietroszek. Verifying feature-based model templates against well-formedness OCL constriants. Submitted for publication, 2005.

[9] C. H. P. Kim and K. Czarnecki. Synchronizing cardinality-based feature models and their specializations. In *Proceedings of ECMDA'05*, 2005. `swen.uwaterloo.ca/~kczarnec/ecmda05.pdf`.

[10] C. W. Krueger. Software mass customization. White paper. Available from `http://www.biglever.com/papers/BigLeverMassCustomization.pdf`, Oct. 2001.

[11] O. Rohlik and A. Pasetti. *XFeature Modeling Tool*. Automatic Control Laboratory, ETH Zürich, 2005. `http://www.pnp-software.com/XFeature/`.

[12] T. von der Maßen and H. Lichter. *RequiLine*. RWTH Aachen, 2005. `http://www-lufgi3.informatik.rwth-aachen.de/TOOLS/requiline/`.