Questionnaire based configuration of product-lines in FeatureIDE

Jens Wiemann, Otto-von-Guericke-Universität Magdeburg,
Stephan Dörfler, Otto-von-Guericke-Universität Magdeburg,
{jens.wiemann, stephan.doerfler}@st.ovgu.de

***Abstract*—Variability management is an essential part of working on product lines. As an established way to simplify the process of product configuration out of software product-lines, feature models are used to describe the set of features and constraints contained in a given software product-line. This paper proposes a method for automatically generating feature models out of descriptive files and naming conventions. Furthermore existing methods of configuration are considered in this work and to develop an alternative based on questionnaires to enable users or customers to configure a product on their own and to allow experts to design the questionnaires according to their domain knowledge.**

***Index Terms*—FeatureIDE, Feature Model, Extraction, Configuration, Questionnaire.**

I. Introduction

SOFTWARE product lines. foo. There are several approaches trying to control the vast amount of product variants though configuration. This allows experts to apply their domain knowledge in order to a resulting product conforming to a users needs.

Feature Models are essential tools for the configuration of product lines in such a way that they give a complete and easily understandable overview of the given features and constraints of a product-line. This work aims at automatically generating feature models out of descriptive files and naming conventions, to simplify a big part of the configuration.

This paper considers existing methods of configuration and tries to come up with a better alternative based on questionnaires to enable users or customers to configure a product on their own and to allow experts to design the questionnaires according to their domain knowledge.

Problem statement

Very high complexity of configuration due to many features and constraints. Domain knowledge is highly required to understand the given software product-line and being able to combine it's features to a valid configuration which satisfies the users needs.

Contribution

Simplifying the process of configuration of product lines through extracting a feature model out of naming conventions and configuration files from an existing product line in FeatureIDE and applying a configuration wizard which guides the user though the creation of a specific product (variant) in the style of a questionnaire, thus applying the domain knowledge of an expert.

This[1] is[2] a[3] dummy[4] to create citations.

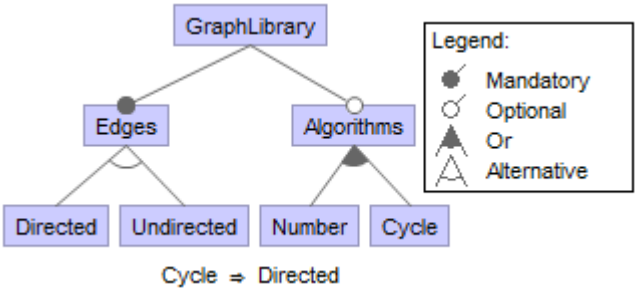

Fig. 1. A simple example of a feature model

II. Basics for simplification of variant configuration

In the scope of simplifying the configuration of a given product line some specific tools and techniques were used. This section will give an overview of what was used in this work to archive the simplification.

A. Feature Models

Feature models are multi-purpose tool for product lines. Figure 1 shows a simplified example of a feature model. Amongst their benefits are:

- Visualization of the possible features and their hierarchy
- Classification of features and their dependencies (alternative/or; optional/mandatory/abstract)
- Formal Representation of the whole product line ⇒ computationally processable
- Assistance/foundation for configuration and variant validation

They are basically structured like trees: There is a root node, an arbitrary number of levels of nodes and finally leaf nodes without child nodes of their own. In that manner feature models map the hierarchy of features. In addition they mark each feature as either mandatory or optional as well as either abstract or concrete. The possible relationships of multiple features with a mutual parent feature are *or* (at least one feature has to be selected), *alternative* (exactly one feature has to be selected) and *and* (any number of features can be selected). As features' relations may be of higher complexity

than just parent-child relations additional constraints can be noted within a feature model. Constraints are boolean expressions, the example in figure 1 shows an implication.

B. Product configuration

Through configuration a concrete product can be derived from a product line. Each valid configuration represents a specific *variant* of the possible products.

During configuration a user selects or unselects features to his needs. This process requires a lot of domain knowledge on the one hand and detailed information about each single feature on the other. With growing numbers of possible features (and thus growing numbers of possible variants) configuration get more and more complex and turns out to be not trivial.

One of the purposes of the feature-oriented approach is saving the effort of creating a whole new product and deriving that product from a product line instead. The sheer overhead of configuration might even negate that benefit if configuration grows too complex.

C. Constraints, contradictions, SAT-solver

As stated above a feature model may contain constraints in the form of boolean expressions. Also, the feature-tree can be expressed as a boolean statement. The model shown in figure 1 can be expressed as follows:

$$\begin{aligned} & GraphLibrary \ \wedge \ Edges \\ & \wedge \ ((Directed \ \wedge \neg Undirected) \\ & \vee (\neg Directed \ \wedge \ Undirected)) \\ & \wedge \ (Algorithms \ \Rightarrow \ (Number \ \vee \ Cycle)) \\ & \wedge \ (Cycle \ \Rightarrow \ Directed) \end{aligned} \tag{1}$$

This formalism allows a configuration to be checked for validity. To do so each selected Feature is appended with a logical *AND* and each specifically unselected feature is also appended with a logical *AND* but gets negated. The resulting expression is then evaluated by a SAT-solver to check for satisfiability. Even during configuration this process can be applied to check for invalid partial configurations after each decision.

III. Workflow based on an unstructured product line

General workflow Diagram

unstructured product line $\rightarrow$ generated feature model $\rightarrow$ optimizing (via SAT-solver?) $\rightarrow$ domain knowledge of an expert $\Rightarrow$ Questionnaire $\rightarrow$ Feature model + Questionnaire $\Rightarrow$ Product (variant)

A. Extraction of a feature model

The (Naming-)conventions and rules that are applied to generate a feature model out of it.
State the problems? Error-Handling on failure to comply the conventions?

B. Questionnaire Approach

Use questions to guide the user through a conditional configuration process. The questions, their order and the influences of the answers are pre-defined by a developer/an expert who uses his domain knowledge to design the questionnaire. Therefore, a data structure is introduced and mapped to a specific set of XML-tags.

IV. Examplary scenarios

FeatureIDE
Explain the used environment

State a few usage scenarios and show why the questionnaire is better or as good as the existing solutions in the given situations. Also point out in which scenario this might be the wrong tool.

V. Conclusion and Future Work

This is where the work is concluded. In this section there will be a description of the way we did things and the experiences we made during it. An emphasis will be on the insights and the findings from the scenarios will get outlined.

Here will be a summary of the new questions that were raised in this work. Also there will be topics for further research. Particularly the problems we encountered and couldn't solve with our concept and why will be pointed out and first approaches will be suggested.

References

[1] M. Antkiewicz, "Featureplugin: Feature modeling plug-in for eclipse," *OOPSLA04 Eclipse Technology eXchange (ETX) Workshop, Oct. 24-28, Vancouver*, 2004.

[2] M. La Rosa, "Questionnaire-driven configuration of reference process models," *BPM Group, Queensland University of Technology, Australia*, 2006.

[3] M. La Rosa, "Questionnaire-based variability modeling for system configuration," *BPM Group, Queensland University of Technology, Australia*, 2008.

[4] D. Batory, "Feature models, grammars, and propositional formulas," *H. Obbink and K. Pohl (Eds.): SPLC 2005, LNCS 3714, pp. 7 20*, 2005.