2010 18th IEEE International Requirements Engineering Conference

Feature Unweaving: Refactoring Software Requirements Specifications into Software Product Lines

Reinhard Stoiber*, Samuel Fricker*†, Michael Jehle* and Martin Glinz*
*University of Zurich, Switzerland
†FUCHS-INFORMATIK AG, Switzerland
Email: {stoiber, fricker, glinz}@ifi.uzh.ch, mjehle@access.uzh.ch

***Abstract*—The design of the variability of a software product line is crucial to its success and evolution. Meaningful variable features need to be elicited, analyzed, documented and validated when an existing software or reference system evolves into a software product line. These variable features are the main discriminators between individual products and they need to reflect the needs of a large variety of stakeholders adequately.**

In this paper we present a novel approach, called *feature unweaving*, that supports the identification and extraction of variable features from a given graphical software requirements model. We have extended our aspect-oriented software product line modeling tool [9] [10] such that it supports feature unweaving: it takes a set of model elements that a domain requirements engineer considers to constitute a variable feature and automatically refactors the model into a semantically equivalent one in which the model elements belonging to this feature are grouped into an aspect. This allows the identification and modeling of variable features in an incremental style. It also substantially reduces both the intellectual and clerical effort required for constructing the variable parts of a software product line requirements model.

I. Introduction

Building a software product line with its full scope from scratch is considerably more expensive than extracting a product line from already existing products or building one with a smaller scope and evolving it incrementally [7]. Recent work has introduced tool support for improving the evolution of an existing portfolio of products into a software product line by mining the requirements specifications of existing valid product configurations and automatically creating a feature model that represents the common and variable features [12] [13]. If multiple consistent product specifications do not yet exist, however, such a method can not be applied.

In this paper we introduce *feature unweaving*, a novel approach for tool-supported, semi-automatic extraction of requirements that constitute a variable feature from a single product or reference requirements model (one that contains all requirements of existing engineering artifacts). Feature unweaving supports domain requirements engineers in incrementally evolving the given product or reference model into a software product line model. Variable features often affect several modules and facets of a system. Aspects allow an explicit specification of such (heterogeneously) scattered concerns. After a domain requirements engineer has identified a selection of model elements that she considers constituting a variable feature, she can call feature unweaving to perform an automated refactoring. Feature unweaving [4] performs four steps automatically: (i) it creates a feature aspect and builds all the necessary internal aspect structure, (ii) it removes all selected elements from the requirements model, (iii) it inserts them into their respective aspect, and (iv) it builds all the necessary weaving semantics. The resulting refactored model is semantically equivalent to the original model. This equivalence can easily be verified by executing a weaving operation of the newly unwoven aspect and comparing the resulting model with the original one. Mistaken or not ideal feature extractions can be undone, too, by simply weaving and removing them again.

The work presented here builds on two core assumptions: First, all facets of a requirements specification are represented in a single, integrated and coherent graphical model, e.g. one written in ADORA [3]. This eliminates information scattering of the impact of variable features over multiple diagrams. View generation is used to leverage its scalability [8] [11]. Second, every variable feature is modularized with one dedicated base aspect. Mandatory features are not modularized and remain part of the core model (the commonality) [11] [9]. If a feature manifests heterogeneously in several modules of a system, then nested sub-aspects are created to provide the necessary internal aspect structure [4].

Feature unweaving supports domain requirements engineers when reasoning about meaningful variable features and specifying the product line variability space. While an engineer still needs to provide the selection of elements she considers a variable feature, the feature unweaving function automates most of the remaining clerical and intellectual effort for feature extraction and aspect-oriented modeling.

II. Feature Unweaving: An Example

Fig. 1 shows a high-level reference requirements model of a real-world industrial product line exemplar of industrial automation devices, modeled in the ADORA language. For a detailed description of the exemplar, its background and the ADORA notation please refer to [9]. This model includes all major requirements of the product line, but no information about the variability, yet.

A domain requirements engineer, who aims at evolving this reference specification into a product line specification,

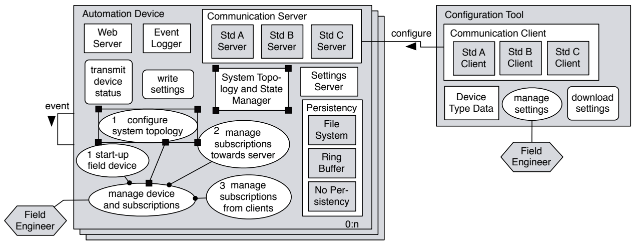

Figure 1. A reference requirements model of a real-world industrial automation devices product line, modeled in ADORA.

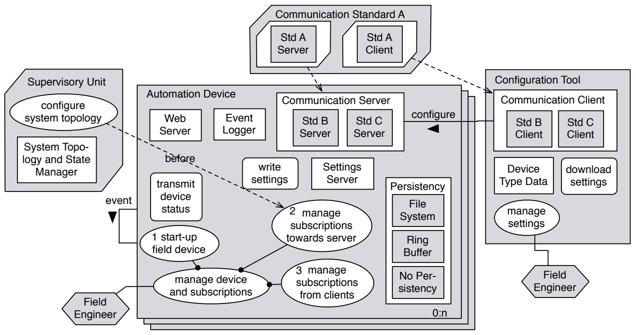

Figure 2. A semantically equivalent, partially refactored software product line requirements model where two variable features are unwoven.

will start her variability analysis with identifying the core variable features. She may recognize that an *Automation Device* can either implement a supervisory unit or an intelligent field device, but not both at the same time. Therefore, these two properties constitute variable features and all related requirements shall be unwoven. In Fig. 1, the model elements that constitute a supervisory unit have already been selected by the domain engineer: the sub-scenario *configure system topology*, its scenario connection, and the component *System Topology and State Manager*. She then calls the feature unweaving function to extract them as a first refactoring.

Fig. 2 shows a partially refactored specification in which two variable features have been unwoven. One of them is the *Supervisory Unit* feature mentioned above – in this case a single base aspect was sufficient to extract all selected model elements. Further, Fig. 2 also shows a second unwoven variable feature - the *Communication Standard A*. The requirements constituting this variable feature were distributed over different modules of the specification, therefore dedicated nested sub-aspects had to be created for every target module [4]. Again, one base aspect modularizes the variable feature as a whole, which is crucial to maintain an easily comprehensible basic structure of the product line.

This process of unweaving variable features continues until all variability is identified. If changes in variability design are required, a variable feature can be woven and re-unwoven with an adapted selection of elements, or be edited manually. ADORA further adds a boolean decision item for every variable feature and allows defining constraints to restrict the allowable configurations of variable features [9]. Valid products can then be derived on this basis, using constraint propagation and selective aspect weaving [10].

III. Related Work

Existing approaches to product line modeling build on a dedicated variability model, typically a feature model [5] [2], and mappings between variable features and their corresponding engineering artifacts. For requirements engineering, these include [2] [1] [13] [14]. The need to define mappings for specifying variable features introduces a significant engineering overhead when systematic software product line engineering gets introduced: codified configuration knowledge as in [1] or detailed actions and pointcut expressions as in [14] need to be defined manually by engineers. Our approach neither requires a dedicated variability model (e.g., a feature model) nor any additional mappings between variable features and requirements. Instead, variable requirements are directly modularized with aspects in the graphic requirements model, using feature unweaving. In contrast to existing work on aspect model unweaving [6], where aspects are considered given and unweaving essentially reverts a previous weaving, feature unweaving supports the creation of aspects in the first place.

IV. Conclusion

The paper has presented a novel approach, called feature unweaving, that supports a domain requirements engineer in refactoring a product or reference model into a software product line model based on aspect-oriented modeling. Future research will address graphical layout adaptions, a complete formalization of the unweaving semantics and a validation of the approach in real organizational settings.

References

[1] R. Bonifácio and P. Borba. Modeling scenario variability as crosscutting mechanisms. In *Proc. of AOSD '09*, pp 125-136, 2009.
[2] K. Czarnecki et.al. Fmp and fmp2rsm: Eclipse plug-ins for modeling features using model templates. In *Proc. OOPSLA Companion*, 2005.
[3] M. Glinz, S. Berner, and S. Joos. Object-oriented modeling with ADORA. *Inf. Syst.*, 27(6): pp 425-444, 2002.
[4] M. Jehle. Feature unweaving: semi-automated aspect extraction in product line requirements engg. Master thesis, Univ. Zurich, 2010.
[5] K. C. Kang, et.al. Feature-oriented domain analysis (FODA) feasibility study. Technical report, CMU/SEI-90-TR-021, Nov 1990.
[6] J. Klein, et.al. Aspect model unweaving. In *Proc. of MoDELS '09*, pp 514-530, 2009.
[7] C. W. Krueger. Easing the transition to software mass customization. In *Proc. of PFE '01*, pp 282-293, 2002.
[8] T. Reinhard, et.al. Tool support for the navigation in graphical models. In *Proc. of ICSE '08*, pp 823-826, 2008.
[9] R. Stoiber and M. Glinz. Modeling and managing tacit product line requirements knowledge. In *Proc. of MaRK'09*, pp 60-64, 2009.
[10] R. Stoiber and M. Glinz. Supporting stepwise, incremental product derivation in product line requirements engineering. In *Proc. of VaMoS'10*, pp 77-84, 2010.
[11] R. Stoiber, T. Reinhard, and M. Glinz. Visualization support for software product line modeling. In *Proc. ViSPLE'08*, 313-322, 2008.
[12] B. Wang, et.al. A use case based approach to feature models' construction. *Proc. of RE'09*, pp 121-130, 2009.
[13] N. Weston, R. Chitchyan, and A. Rashid. A framework for constructing semantically composable feature models from natural language requirements. In *Proc. of SPLC '09.*, pp 211-220, 2009.
[14] S. Zschaler, et.al. Vml* - a family of languages for variability management in software product lines. In *SLE'09*, pp 82-102, 2009.