Configuration of Mechatronic Multi Product Lines

Christopher Brink
Software Engineering Group
Heinz Nixdorf Institute
University of Paderborn, Germany
Christopher.Brink@uni-paderborn.de

Martin Peters Sabine Sachweh
University of Applied Sciences and Arts
Dortmund, Germany
{Martin.Peters ||
Sabine.Sachweh}@fh-dortmund.de

Abstract

For the development of variable systems, software product lines (SPL) are an established way to handle the variability by using feature models. Nevertheless, the configuration of an SPL can be complex, especially if a product line consists of a large number of features. The problem of handling the complexity becomes even more sophisticated if not only software, but also mechatronic systems containing software and hardware components are configured. Besides modeling the software, within a mechatronic system dependencies and associations between software and hardware features need to be considered which further increases the complexity.
To handle this complexity in product lines for mechatronic systems, we propose a multi product line (MPL) approach which allows to distinguish between software and hardware by using different feature models for each. In addition we introduce a level of abstraction to complex product lines consisting of multiple feature models by establishing a feature model mapping. In this paper we present details to the mapping to provide an abstract configuration view as well as the introduced associations for our MPL approach.

Categories and Subject Descriptors D.2.9 [*Software Engineering*]: Management—Software configuration management, Productivity; D.2.10 [*Software Engineering*]: Design—Methodologies

General Terms Design, Management

Keywords Mechatronic Multi Product Lines, Feature Models, Mapping

1. Introduction

The development of mechatronic systems combines the engineering disciplines of mechanical engineering, electrical engineering and software engineering. Accordingly the development of mechatronic systems, composed of among other things, controllers for continues behavior and real time protocols for the coordination of connected systems [15], is a complex task. Due to the increasing complexity of these systems and the variety of products, resulting both from the combination of hardware and software components, the development time and costs of mechatronic systems are increasing, too.
One opportunity to handle the complexity is the use of feature models introduced by [11] to describe the variable and common parts of software product lines [5]. A feature model consists of a hierarchically arranged set of features connected through different types of associations. In different approaches [1, 8] feature models are used for product configuration. Therefore features are connected to software development artefacts. Figure 1 depicts an example of a feature model describing a navigation system in a car. While the application of feature models for describing product lines is well known and integrated in different approaches [3, 7, 9] most of the applications take place in the field of software product line engineering. Hardware engineering in contrast is usually performed by the use of large spreadsheets and configuration files.

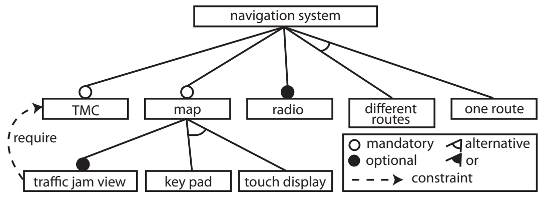

Figure 1. Example of a feature model

For a better integration of different disciplines of engineering and easier handling of the complexity of variable mechatronic systems, we propose a multi product line approach which allows us to use different feature models for software and hardware features by introducing dependencies between multiple feature models. Each feature may contain development artefacts derived by one of the aforementioned engineering discipline. We also establish a mapping of different

VariComp'12, March 26, 2012, Potsdam, Germany.
Copyright © 2012 ACM 978-1-4503-1101-4/12/03. . . $10.00

feature models to one abstract model to ease the handling of the complexity and to provide an user specific view with a restricted choice of features.This abstract model and the corresponding mapping allows a simple product configuration by the user without knowledge of the different development artefacts.
In the following we first indicate the use of product lines for mechatronic systems before we focus on the abstraction of product lines and the underlying mapping for ease of complexity in section 4. Section 5 will give an overview of related work and the delimitation to the presented work. The last section will conclude the paper and point out future activities in our research.

2. MPLs for mechatronic systems

The modeling of software and hardware features of a mechatronic system within one feature model may not only lead to a high complexity, but also has some disadvantages and limitations in further processing, resulting from the combination of software and hardware features and the respective deployment. On the one hand it is not possible to distinguish between software and hardware features during processing of a configuration. On the other hand a feature model does not allow to describe the distribution of a software on different hardware components. Contrariwise a feature model does not provide an opportunity to model the deployment of multiple software artefacts on one hardware component. For example, a car manufacturer may use a feature model to handle the configuration of a car navigation system including the electronic control unit (ECU) and the software for different types of navigation systems. In this scenario it would not be possible to deploy another piece of software, for example to control the electric windows, on the same ECU. Also further dependencies between the different system parts and their properties can not be modeled adequate in a combined feature model. Therefore we propose an approach where different system parts can be managed within different feature models and additionally associations between those feature models exist to specify dependencies and further information for a product configuration. A fragmentation of a mechatronic system into different fea-

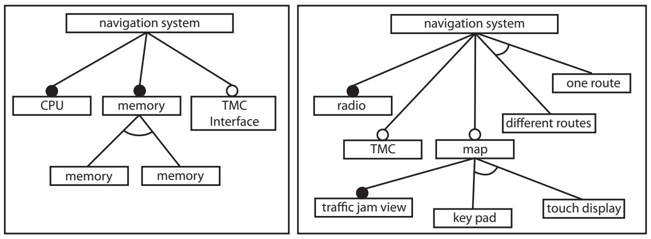

Figure 2. Fragmentation into multi feature models

ture models by differentiating between software and hardware parts is depicted in figure 2, while the dependencies of the multiple feature models are introduced in section 3. With the breakdown into multiple feature models it is possible to treat software and hardware features in a different way and connect them with different artefacts. For example a software feature may be associated with a software artefact of the Mechatronic UML [2], which allows modeling of continuous and discrete behavior of mechatronic systems, while a hardware feature is associated with a parts list and a circuit diagram. In addition it is possible to not only generate software from a product configuration, but also to produce a hardware list and a deployment diagram. The hardware list may also include further information about the technical details or the ordering of specific parts, while the deployment diagram connects the software and hardware parts and may be used as a blueprint for the configured product. The deployment diagram may also include different software platforms which are needed.

3. Multi product line constraints

To connect software and hardware artefacts distributed over multiple feature models in an adequate way, a set of associations is necessary. Therefore the well known *require* and *exclude* dependencies for features within one feature model described amongst others in [6] are not adequate. Hence we extend this concept and apply it to features in different feature models as well as to feature models itself. For example the selection of the *Traffic Message Channel* (TMC) feature in the software feature diagram in figure 2 is only valid, if the hardware for the *TMC Interface* is selected, too. Accordingly there must be a *require* dependency between the software feature *TMC* and the hardware feature *TMC Interface*. In addition we introduce one more type of dependency which can be applied to one feature and a feature model. The *deploy* dependency describes deployment information of an artefact and specifies the execution environment. For example the software of an navigation system, which may be variable due to different maps offered by the manufacturer, may have dependencies to different kinds of hardware for the navigation system which differ in the available memory.
The resulting dependencies are summarized in table 1. For

Table 1. Different types of dependencies

Dependency	**Description**
require	Requires the target feature to be selected in a configuration, too.
exclude	Excludes the selection of the target feature in the current configuration.
deploy	Describes the hardware or software platform used for deployment.

each type of dependency, we additionally differentiate between constraints (CT, applied between features within one feature model), cross feature constrains (FCT, applied between features of multiple feature models) and feature model constrains (MFCT, applied to multiple feature models and so to root-features) so that we finally have the following set of contraints:

- $CT = \{require, exclude\}$
- $FCT = \{require, exclude, deploy\}$
- $MFCT = \{require, exclude, deploy\}$

The aforementioned dependencies allow us to describe different system parts, tailored into software and hardware features, within multiple feature models and so to handle complex mechatronic systems. Another advance is the possibility to specify deployment information and requirements between features. Figure 3 represents the scenario described in figure 2 completed with the introduced dependencies. Nevertheless we found out, that the use of simple dependencies is not always satisfying. That is the case, if one feature has multiple features as precondition and otherwise would not lead to a valid configuration. To be able to handle such conditions we plan do develop a notation to map complex dependencies in the future, too.

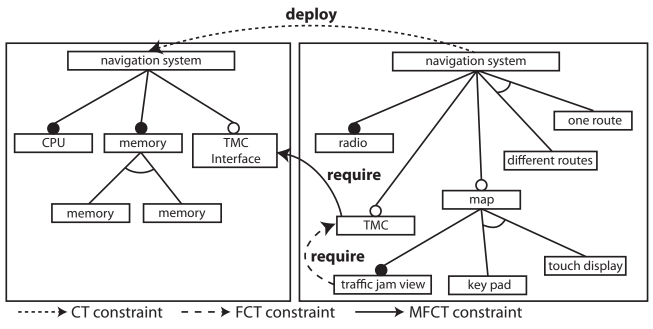

Figure 3. Dependencies between feature models

4. Abstraction for the configuration of MPLs

The introduction of the cross model dependencies makes it easier to handle different system components and to differentiate between hardware and software artefacts. Contrariwise it can also make it harder to handle a configuration because of additional dependencies and references which on the other hand raise the complexity of a diagram. To simplify the process of configuration in a multi product line we propose a feature model mapping, which maps multiple feature models and associated references to one simple feature model to hide the complexity for a user and to be able to provide a user specific view.

Before we present the mapping of a multi product line to an abstract feature model, we first introduce the formal definition of a feature model as well as of a multi feature model. In doing so we use the definition made by Trigaux et. al in [18] and adopt it for our needs. The graph type of a feature model in this paper is defined as a tree, while NT (Node Type) is a set of Boolean functions which describe the type of node NT = {mandatory, optional, alternative, or}. The Constraint Type (CT) in turn is defined as a binary boolean operator and is used to describe dependencies (require, exclude) which may exists between features of one feature model. As described in section 3 for a multi feature model we also need constraint types for constraints between features of different feature models (FCT) as well as constraints between different feature models (MFCT). The aforementioned description leads to the following definition:

Definition 1. *A feature model $FM = (\mathcal{N}, r, \lambda, \mathcal{E}, \mathcal{C})$ where:*

- *$\mathcal{N}$ set of nodes (nodes are features)*
- *$r \in \mathcal{N}$ is the root feature of the feature model FM*
- *$\lambda : \mathcal{N} \to NT$ labels each node with an operator from NT*
- *$\mathcal{E} \subseteq \mathcal{N} \times N$ set of edges*
- *$\mathcal{C} \subseteq \mathcal{N} \times CT \times \mathcal{N}$ is the set of constraint edges*
- *$NT = \{mandatory, optional, alternative, or\}$*
- *$CT = \{require, exclude\}$*

Definition 2. *A Multi Feature Model $MFM = (\mathcal{FM}, \mathcal{FC}, \mathcal{FMC})$ where:*

- *$\mathcal{FM}$ set of feature models*
- *$\mathcal{FC} \subseteq \mathcal{N}' \times FCT \times \mathcal{N}''$ set of constraint edges between features of different feature models*
- *$\mathcal{FMC} \subseteq \mathcal{FM}' \times MFCT \times \mathcal{FM}''$ set of contraint edges between different feature models*
- *$FCT = \{require, exclude, deploy\}$*
- *$MFCT = \{require, exclude, deploy\}$*

The mapping between the multi feature model $MFM = (\mathcal{FM}, \mathcal{FC}, \mathcal{FMC})$ and an abstract model, which is also a feature model, $FM_a = (\mathcal{N}_a, r_a, \lambda_a, \mathcal{E}_a, \mathcal{C}_a)$ is defined as follows:

$$\begin{aligned} R \subseteq \mathcal{N}_a \times \mathcal{N}_{MFM} \\ with: \\ \mathcal{N}_{MFM} = \bigcup_i \mathcal{N}_i, \\ \mathcal{FM}_i = (\mathcal{N}_i, r_i, \lambda_i, \mathcal{E}_i, \mathcal{C}_i) \in \mathcal{FM} \end{aligned} \tag{1}$$

That means that one feature of the abstract model maps multiple features in an arbitrary number of underlying feature models and each abstract feature must be mapped to at least one feature. The introduced mapping can be used to create a clearly and user defined view of a complex multi product line of a mechatronic system. In contrast to the direct presentation of a multi product line including all features, the abstracted view can be used by a customer to configure a product without knowledge of technical details or knowledge about dependencies. The abstraction is achieved by mapping one abstract feature to a variety of features in underlying feature models. Moreover abstract feature models may be enhanced with additional user information like introduced by Streitferdt in [16], without rising the complexity of the underlying feature models.

Within this view, all mandatory features may be hidden to

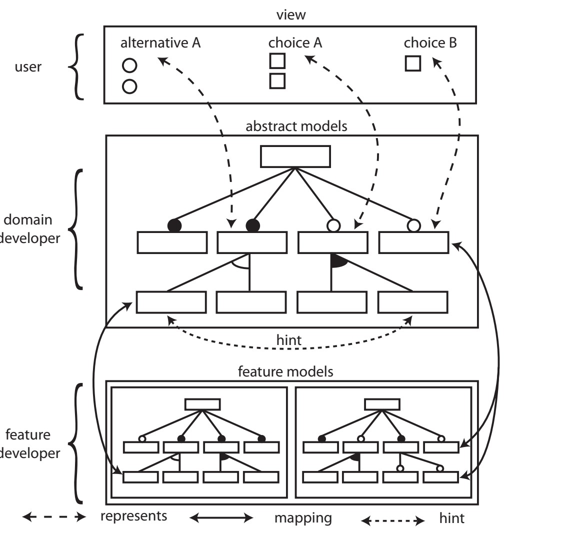

Figure 4. feature model abstraction

provide a clearly arranged user interface containing only selectable options while the aforementioned user information provide additional assistance during the configuration process. In addition the abstract model can be used to make a pre-selection of features or to restrict the available features for a specific customer, domain or type of product. In the car industry for example, this might be the case with the configuration of a BMW 5 and a BMW 7, which each on its own is a variable product. Although both cars share some components like control gears and other electronic equipment, they do not allow an arbitrarily combination. With the introduced abstract model it is possible to provide a single feature model for each kind of car even if the underlying feature models may be associated through different kinds of dependencies and thereby belong to one complex multi product line. These concepts are depicted in figure 4.

5. Related work

Recently many approaches were proposed to extend the concept of product lines to integrate software development artefacts like requirements, classes or components [1, 4, 16]. Apel et. al. introduced a way to connect classes as well es refinements of classes with features. Depending on the selected configuration of a product line, those classes were modified based on the previous mentioned refinements. Finally, the software is provided by the generated artefacts. Although this approach is promising for software product lines it is not adequate for mechatronic systems because neither hardware artefacts nor dependencies between software and hardware artefacts can be considered. Streitferdt extends the concept of product lines in his PhD-thesis and consideres an association between features and requirements. Therefore he identifies different types of dependencies which mainly differ from the dependencies introduced in this paper that they only can be applied to features within one feature model. In addition deployment dependencies are not taken into account.

Another refinement of the concept of product lines is introduced in [5], where a process for the development of software product lines is presented. This concept is used in our approach and accordingly applied to multi product lines for mechatronic systems.

In [12] Apel et. al. outlines a process for a combined hardware and software product line which would meet the scenario of a product line for mechatronic systems. Nevertheless they do not consider the necessary associations between features so that dependencies are not taken into account.

One approach where the configuration of software and hardware features is considered within one product line was suggested in [17]. However, only requirements are mapped and only one single feature model is used, which limits the differentiation between software and hardware features. Furthermore no information about the deployment and execution environment can be integrated

The handling of multi product lines was focused in [14] where Rosenmüller et. al. introduced composition models to associate multiple feature models within one multi product line to generate a configurator based on the composition models. While this approach may be at first sight quite similar to the introduced abstract feature models, our approach provides much more flexibility through the mapping of one abstract feature to many features of different feature models. In addition our approach enables the generation of a user specific view with for example hidden options, regardless of whether these are mandatory or just not available for a specific customer. Reiser et. al. introduced in [13] an approach for modeling multi level feature trees. In contrast to our approach, Reiser et. al. organize feature models hierarchically and do not consider different development artifacts. In [10] an approach is presented which allows to combine different modelling methods for variability by the use of web services. In contrast to our approach they do not consider hardware parts, a deployment constraint or an abstract configuration view, which is necessary for modelling mechatronic systems. To simplify the process of configuration Czarnecki et. al [8] introduced an approach where the configuration is tiled into different steps based on different levels of abstraction. A previously selected configuration directly affects further variation points and limits the available options. This concept should allow the configuration through various stakeholders where an expert starts with the configuration and finally the customer gets a restricted view with a highly limited choice. Our approach in contrast provides a direct configurable view for customers through the use of abstract feature models. A preselection of features could be realized by the use of OEM. Furthermore our approach enables the configuration of multiple product lines without any expertise about underlying models.

Benavides et. al. also describe a progressively approach for the configuration of product lines in [19]. Therefore they use a formal model and map it to a constraint satisfaction problem which than can automatically be resolved by a constraint resolver. While this is an interesting way to handle the complexity of a product line, it is not a satisfying concept for multi product lines.

6. Conclusion and further work

Within this paper we proposed a multi product line approach for mechatronic systems which allows to model the variability of hardware as well as of software in conjunction with necessary software platforms used for deployment. Single system parts are tailored into different feature models to be able to describe software and hardware features differentiated from each other. To model the associations and dependencies between different system parts we extended the concept of dependencies within one feature model to facilitate the employment to features of different feature models. With the redefined characteristics it is possible to configure a mechatronic system based on feature models, containing features associated with development artefacts for the hardware in the same way like software artefacts such as components of the Mechatronic UML.

To simplify the process of configuration an abstract model was introduced which allows the aggregation of multiple features of different feature models and so to hide the complexity resulting from additional associations and the variety of features. Through the use of this concept a complex multi product line may be configured by a customer without knowledge of development artefacts associated with features. The proposed concept was prototypically implemented in FoCuS, a java based tool providing a web interface for configuration.

Further work will include the extension of the proposed concept towards a language to describe complex dependencies. Furthermore we plan the integration of the Mechatronic UML to be able to describe the reconfiguration of mechatronic systems. Therefore we are going to investigate which parts of the model need to be integrated in features and how these parts need to be considered during configuration.

Acknowledgments

This work was funded partly by the Federal Ministry of Education and Research (BMBF), project number, 17N3909.

References

[1] S. Apel and C. Kästner. An overview of feature-oriented software development. *Journal of Object Technology*, 2009.

[2] S. Becker, S. Dziwok, T. Gewering, C. Heinzemann, U. Pohlmann, C. Priesterjahn, W. Schäfer, O. Sudmann, and M. Tichy. MechatronicUML - syntax and semantics. Technical Report tr-ri-11-325, Software Engineering Group, Heinz Nixdorf Institute, 2011.

[3] D. Benavides and S. Segura. Automated analysis of feature models 20 years later: A literature review. *Information Systems*, 2010.

[4] T. Buchmann. *Modelle und Werkzeuge für modellgetriebene Softwareproduktlinien am Beispiel von verwaltungssystemen (in German)*. PhD Thesis, University of Bayreuth, 2010.

[5] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley Professional, 2001.

[6] K. Czarnecki and U. Eisenecker. *Generative Programming Methods, Tools, Applications*. Addison-Wesley, Boston, 2000.

[7] K. Czarnecki, T. Bednasch, P. Unger, and U. Eisenecker. Generative Programming for Embedded Software : An Industrial Experience Report. Technical report, 2002.

[8] K. Czarnecki, S. Helsen, and U. Eisenecker. Staged Configuration Through Specialization and Multi-Level Configuration of Feature Models. In *Third International Software Product Line Conference*, 2004.

[9] K. Czarnecki, C. Hwan, and P. Kim. Cardinality-Based Feature Modeling and Constraints : A Progress Report. Technical report, 2005.

[10] 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*, 2011.

[11] K. C. Kang. Feature-oriented domain analysis, feasibility study. Technical Report November, DTIC Document, 1990.

[12] J. Liebig, S. Apel, C. Lengauer, and T. Leich. RobbyDBMS: a case study on hardware/software product line engineering. In *First International Workshop on Feature-Oriented Software Development*. ACM, 2009.

[13] M.-O. Reiser and M. Weber. Managing Highly Complex Product Families with Multi-Level Feature Trees. *14th IEEE International Requirements Engineering Conference*, 2006.

[14] M. Rosenmüller and N. Siegmund. Automating the configuration of multi software product lines. In *Fourth International Workshop on Variability Modelling of Software-intensive Systems*, 2010.

[15] W. Schäfer and H. Wehrheim. The challenges of building advanced mechatronic systems. In *Future of Software Engineering*, 2007.

[16] D. Streitferdt. *Family-oriented requirements engineering*. PhD - Thesis, Technical University of Ilmenau, 2004.

[17] D. Streitferdt, P. Sochos, and C. Heller. Configuring Embedded System Families Using Feature Models. In *Proc. of Net. ObjectDays*, Erfurt, 2005.

[18] J. Trigaux, P. Heymans, P. Schobbens, and A. Classen. Comparative semantics of feature diagrams: Ffd vs. vdfd. In *Fourth International Workshop on Comparative Evolution in Requirements Engineering*, 2006.

[19] J. White, B. Dougherty, D. Schmidt, and D. Benavides. Automated reasoning for multi-step feature model configuration problems. In *13th International Software Product Line Conference*. Carnegie Mellon University, 2009.