Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 50-1 (2017) 4336–4341

Supporting the Development of Interdisciplinary Product Lines in the Manufacturing Domain

Matthias Kowal * Sofia Ananieva ** Thomas Thüm * Ina Schaefer *

* *TU Braunschweig, Germany*
{m.kowal, t.thuem, i.schaefer}@tu-braunschweig.de
** *FZI Research Center for Information Technology, Germany*
ananieva@fzi.de

Abstract: The increasing demand for highly customizable manufacturing systems leads to an extreme number of possible machine variants. Feature models are often used to manage this system diversity. The development and maintenance of feature models are error-prone and time-consuming tasks, especially considering industrial-size models with thousands of features. In many cases, engineers might want to focus only on a few features relevant for their own domain. Additionally, each change may lead to anomalies in the feature model. In this paper, we present an approach to provide engineering support by giving user-friendly explanations for hidden dependencies and anomalies in feature models.

Keywords: Variability Modeling, Engineering, Machine Manufacturing, Variant and Version Management

1. INTRODUCTION

Customers have a rising demand for fully customizable products that can be tailored to their specific requirements (Pohl et al. (2005)). In return, manufacturers have to pay more attention to variability and its management to deal with the rising complexity introduced by the variant diversity, e.g., as in the automotive domain. Engineers tried to reduce this problem with the introduction of product lines several decades ago (Kang et al. (1990)). A product line is comprised of a set of related systems that share several commonalities and variabilities. For example, each car must have a radio making it a common feature, but a navigation system is only optional. The goal of product lines is to foster reuse potential, reduce maintenance effort and provide a better cost-efficiency (Czarnecki and Eisenecker (2000); Pohl et al. (2005)). Feature models are often used to express the variability as well as dependencies in a product line (Benavides et al. (2010)). Thousands of features and dependencies between these features are common in industrial-size feature models (Tartler et al. (2011)). Engineers often encounter two major problems while dealing with feature models.

First, product line development is an interdisciplinary process involving multiple developers from different domains, e.g., mechanical-, software-, and electrical engineering. Hence, the feature model contains information that may not be relevant for a certain domain or developer and can be hidden (Lettner et al. (2015); Feldmann et al. (2015); Ananieva et al. (2016)). It is crucial that no information is lost during such a process. Dependencies between features from different domains must still be respected and visible to the developer. In addition, hidden dependencies may occur in the partial feature model due to constraints across the complete product line. We refer to these hidden dependencies as implicit constraints and provide engineering support by giving explanations why they are present leading to more precise communication between different disciplines and help identify unintended interferences.

Second, the maintainability of a feature model decreases with its size (Mens and Demeyer (2008)). Evolution of a feature model due to changing requirements, the addition of new features or dependencies has an increasing possibility to introduce anomalies (Mens and Demeyer (2008)). Anomalies can be rather harmless such as redundancy meaning that semantic information is modeled in multiple ways which is usually not preferable (von der Maßen and Lichter (2004)). However, anomalies can also be severe such as dead features. It is not possible to select a dead feature for any variant of the system making it useless. In order to support developers in the removal of anomalies, they must be detected and explained to comprehend the cause why an anomaly has occurred in the feature model (Benavides et al. (2010); Kowal et al. (2016).

In this paper, we present an approach supporting engineers in both aspects: (1) depicting partial feature models with all implicit constraints as well as their explanation and (2) giving user-friendly explanations for anomalies, without introducing new concepts and notations for feature models or increasing the modeling workload.

2. CASE EXAMPLE: PICK AND PLACE UNIT

The running example is a product line form the automation engineering domain. The Pick and Place Unit (PPU) is a universal production demonstrator for studying evolution and variability (Legat et al. (2013)). It consists

2405-8963 © 2017, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2017.08.870

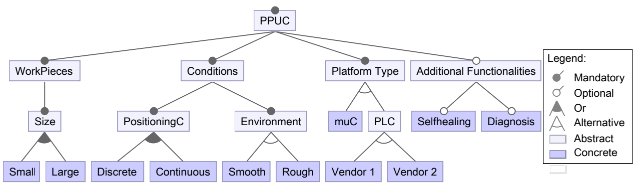


Fig. 1. Customer feature model (Legat et al. (2013))

of multiple variants and provides us with source code, UML diagrams and four feature models depicting different domains involved in the development of the PPU.

2.1 Feature Models

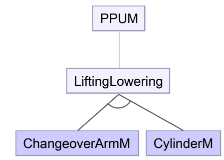
A feature model consists of a hierarchically arranged set of features and has typically a tree-like graphical representation. Fig. 1 shows a feature model of the PPU from the customer's point of view. Parent-child relationships are expressed using the following elements and semantics (see legend in Fig. 1 for the graphical representation (Kang et al. (1990); Czarnecki and Eisenecker (2000)):

- *Mandatory* – feature must be selected, if parent is,
- *Optional* – feature is optional,
- *Or* – one or more subfeatures can be selected,
- *Alternative* – only one subfeature can be selected.

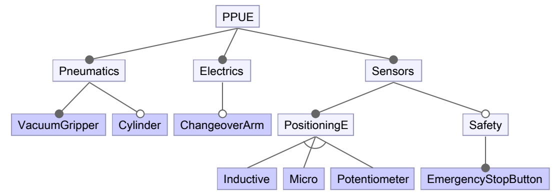
For example, the PPU can handle two types of workpieces simultaneously with *Small* and *Large*. The operating environment can either be *Rough* or *Smooth*, but not both at the same time. *Selfhealing* and *Diagnosis* are optional features. *Abstract* features are only used for structural aspects and do not contain realization artifacts, e.g., source code. Dependencies between features that are not part of a parent-child relationship are expressed with cross-tree constraints using propositional logic, $X \Rightarrow Y$. In case of the PPU feature model depicted in Fig. 1, cross-tree constraints are not present.

The development and maintenance of the PPU involves multiple disciplines with mechanical, software and electrical engineering. The customer feature model in Fig. 1 does not represent all disciplines in a sufficient manner which is why three additional engineering feature models are available (Legat et al. (2013); Feldmann et al. (2015)). Fig. 2 depicts the individual models describing the PPU in more detail for each domain. It is obvious that some similar features can be identified in multiple feature models, while other features are restricted to one model, since they are not relevant for other domains. The goal of separate feature models is to reduce the complexity for the engineers and let them focus on important parts for their domain. Several feature models in isolation are not sufficient to completely describe a product line. It is mandatory to express dependencies between the individual feature models as well. The developers of the PPU created a mapping matrix to express these global constraints connecting the customer feature model to the engineering models (Legat et al. (2013); Feldmann et al. (2015)).

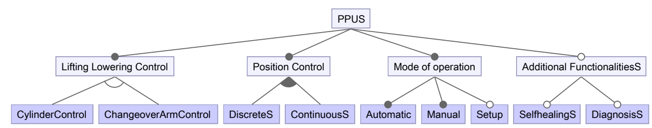
For example, the customer can select the small workpieces resulting in the selection of the features *ChangeoverArmM* in the mechanical, *ChangeoverArm* and *VacuumGripper* in the electrical and *ChangeoverArmControl* in the software model. Fig. 3 shows only an extract of the original matrix defined by Feldmann et al. (2015).



(a) Mechanical feature model



(b) Electrical feature model



(c) Software feature model

Fig. 2. Engineering feature models of the PPU (Legat et al. (2013); Feldmann et al. (2015))

	Developer's point of view					
	Mech.	Electrics/ Electronics			Software	
	Lifting/ Lowering	Pneum.	Electr./ Electron.	Sensors	Lifting/ Lowe-ring Control	Position Control
view / Work pieces / Size / Small	Change-over arm	Vacuum Gripper	Change-over arm		Changeover arm Control	
view / Work pieces / Size / Large	Change-over arm / Cylinder	(Cylinder) Vacuum Gripper			(Cylinder Control)	

Fig. 3. Extract from the mapping matrix (Feldmann et al. (2015))

2.2 Problem Statement

Engineers most likely maintain and develop only the feature model for their own domain. However, crucial information may be lost by considering just a portion of the product line, e.g. the dependencies expressed by the mapping matrix. The number of dependencies can easily add up to several thousands in industrial-size feature models making it unreasonable to present all of them, since only a small part is relevant to individual engineers. Additionally, the product line dependencies can produce implicit constraints in the considered partial feature model that are not visible at first. Regardless of the model part, each change may lead to an inconsistency. While the detection of such anomalies is well-researched, the actual explanation is often neglected or completely missing. We derive and present all relevant dependencies for an arbitrary partial feature model as well as explain the cause of an implicit constraint and all appearing anomalies. To maximize usability, we refrained from introducing new modeling concepts or notations while providing a fully functional open-source implementation in the FeatureIDE framework.

3. IMPLICIT CONSTRAINTS IN FEATURE MODELS

The definition of a mapping matrix is a successful first step to express dependencies between separate feature models (Feldmann et al. (2015)). Nevertheless, it has some drawbacks in terms of scalability and it is difficult to analyze. The connection of the individual engineering models