Product Lines, Features, and MDD[1]

Bruno González-Baixauli, Miguel A. Laguna, Yania Crespo

Department of Computer Science, University of Valladolid,
Campus M. Delibes, 47011 Valladolid, Spain
{bbaixauli, mlaguna, yania}@infor.uva.es

Abstract: One of the most important factors of success in the development of a software product line is the elicitation, management, and representation of the variability. In this context, this article explores the advantages of adopting the Model Driven Development (MDD) paradigm in the variability management, including the transformation of feature graphs into UML models. The global picture involves a sequence of models from requirements to features and from both to the architecture (a UML model). The conclusion is positive as the introduction of MDD ideas raises the abstraction level in the instantiation process of the product line. More effort is needed to further evaluate some of the ideas related to MDD transformations: in particular, traceability register is essential if we want to exploit their benefits.

1 Introduction

Product lines (PL) have become the most successful approach in the reuse field, but is a very complex concept that requires a great effort in both technical [2, 4] and organizational [1] dimensions. Our previous work intends to define a PL development process that requires less investment and presents results earlier than more traditional product line methods, by adapting a conventional process [12]. In this paper we focus on the ideas of Model Driven Development (MDD) and, in particular, Model Driven Architecture (MDA) that can help this development process. MDA was introduced by the Object Management Group (OMG) and is based on the Platform Independent Model (PIM) concept. The PIM is a specification of a system in terms of domain concepts and independently of platforms. The system can then transform the PIM into a Platform Specific Model (PSM) [14]. MDD has a wider vision and is founded on well-established practices of software reuse field. As the main strength of MDA is the transformation of different models and feature models are the basis of our process, it is worth exploring the relations of these models with UML conventional models. The next section briefly introduces *Product Line Requirements Engineering* and discusses the benefits that MDA/MDD can bring to this product line approach, including the definition and implementation of the transformation of feature graphs into UML models. Section 3 concludes the paper and proposes additional work.

[1] This work has been supported by the Spanish MEC/FEDER (TIN2004-03145).

2 MDD and Product Line Requirements Engineering

The *Product Line Requirements Engineering* discipline, as defined in our process, includes several activities. The main activity involves the specification of the domain model, which consists in the domain features. The design of a solution for these requirements constitutes the architectural asset base of the product line (typically implemented as an OO Framework). Later, in the application engineering process, an application model must be derived from the domain model. In this process alternative concepts are selected based on customer functional and non-functional requirements. This activity is essentially a transformation process where a set of decisions taken by the application engineer generates the initial feature product model and, consequently, via traceability links, the initial architecture of the product. The variation points are selected on the conceptual level on the basis of a rationale provided by functional and non-functional requirements.

One of the most critical points is the elicitation and analysis of requirements variability. In addition to the information that expresses the requirements themselves, it is important to know the variability of the requirements, and the dependencies between them. Our proposal was initially based on the work of the SEI on use cases and features [3]. To represent this kind of information, the requirements are usually structured in feature hierarchies [9, 10]. Thus, each user requirement is an identifiable functional abstraction, or feature. The features are organized by a graphical AND/OR hierarchical diagram, i.e. the feature graph, which captures the logical structural relationships between requirements. Although its effectiveness has been proven in many projects, we think that this strategy has an intrinsic weakness: it is oriented to the solution more than to the requirements. We therefore believe that specific requirements engineering techniques can help. Not only the functionality but also non-functional requirements (NFR) must be taken into account. This has led us to consider other possibilities such as Goal Oriented Requirements Engineering [13] as a way of introducing intentionality in the elicitation and analysis of the requirements variability.

It is worth analyzing the possibilities that the MDD/MDA ideas can bring to this field. Essentially we are searching for an (ideally automated) derivation of an optimal specific product in a product line, while taking into consideration functional and non-functional requirements and using the goals and feature models and their correlations as the starting point. A set of transformations between these models can actually be carried out. The general schema is presented in Figure 1. The interest of using our complementary goal/feature model is twofold: a) it allows the application engineer to deduce (if the traceability links are carefully established) what features are needed to reach the selected goals (or functional requirements), and b) which is the optimal set of goals/features in the context of a set of NFR (expressed as *soft-goal*s) of a determined priority that provides the rationale of the selection. In practice, this supposes a rise in the abstraction level of the variants selection process, making the selection in the requirements level instead of in the feature level. The novelty with MDA is the possibility of the automation of some of the transformations of Figure 1.

There are basically two kinds of transformation: horizontal and vertical. Horizontal transformations derive a subset of the main model by means of manual selection in the goal level, configuration in the feature level and instantiation in the architecture level. The first one must be done by the domain analyst from the user goals, but the others can be automatized if we have trazability from the goals to the features that implement them and from features to derived architectural classes (integrated in the vertical transformations).

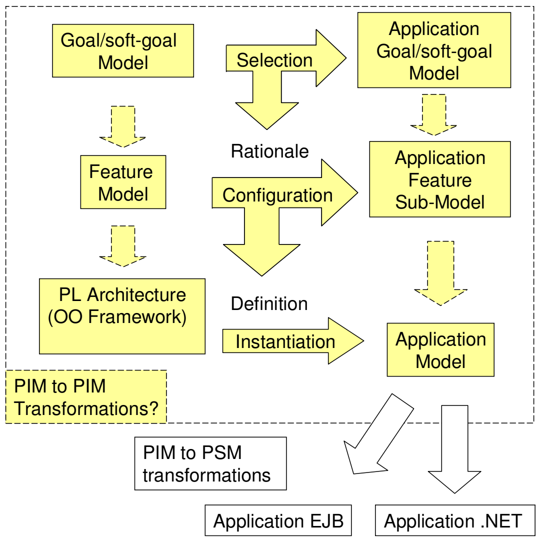
Goal/soft-goal Model
Selection
Application Goal/soft-goal Model
Rationale
Feature Model
Configuration
Application Feature Sub-Model
Definition
PL Architecture (OO Framework)
Instantiation
Application Model
PIM to PIM Transformations?
PIM to PSM transformations
Application EJB
Application .NET

Fig. 1 Product line engineering and MDA and the scope of this study. Left and right parts of the Figure refer to Product Line and Application processes respectively.

We focus in this paper on vertical transformations, where the changes are in the abstraction level. In particular, we center on the PL Feature Model to PL Architecture transformation, obtaining the architecture skeleton from the features. The method we have chosen is based on the meta-model mapping approach [6]. The work consists in defining a set of transformations between the elements of variability in the feature models and the architectural solutions (really each kind of variability in the feature model can be implemented by more than one technique [5]).

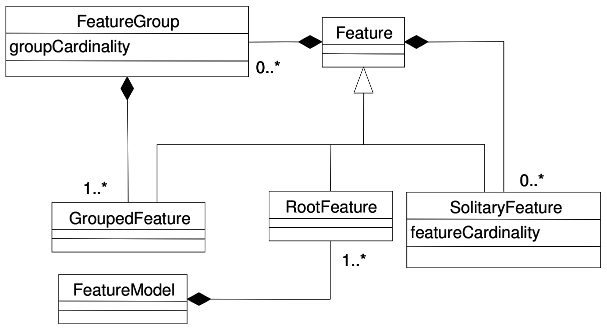

Fig. 2 Feature simplified meta-model, adapted from Czarnecki et al. [7]

The way to define a transformation is to select an element of the feature meta-model and give one or several equivalences in the UML meta-model. This implies that an annotation is needed in the feature model to select one of the possible design mechanisms. As we need a precise definition of the meta-models, the first consideration is to answer the question about the meta-model compatibility of these different models. It is clear that the PL Architecture meta-model is the UML meta-model. The feature models (and sub-models) are built using other concepts but several studies have specified different meta-models using MOF. We have explored these meta-models as the election influences greatly the transformation process. The meta-model proposed by Czarnecki et al. [7], has been selected because the simplicity of the related transformation. In this approach, three types of features are differentiated and the distinctive property of the relationships is the cardinality (Figure 2).

The strategy of transformation is based in the three subtypes of Feature. The root of every tree in a Feature model (RootFeature) is transformed in a class and a recursive transformation of Solitary Features and Feature Groups linked to every feature is carried out. The presence of a group implies a class associated to the parent feature that is specialized into several subtypes (one per alternative feature).

Figure 3 shows partially the graphically expressed transformation (using the last QVT submission syntax [15]) of a Model Feature into a UML/XMI model. In fact, the transformation is most interesting if we consider that the framework obtained (and completed by the designer but with the links saved) can be used to automatically derive the application model by selecting the desired goal/features, as mentioned above. This possibility compensates the overcharge of complexity of the goal/feature traceability management in the architectural model.

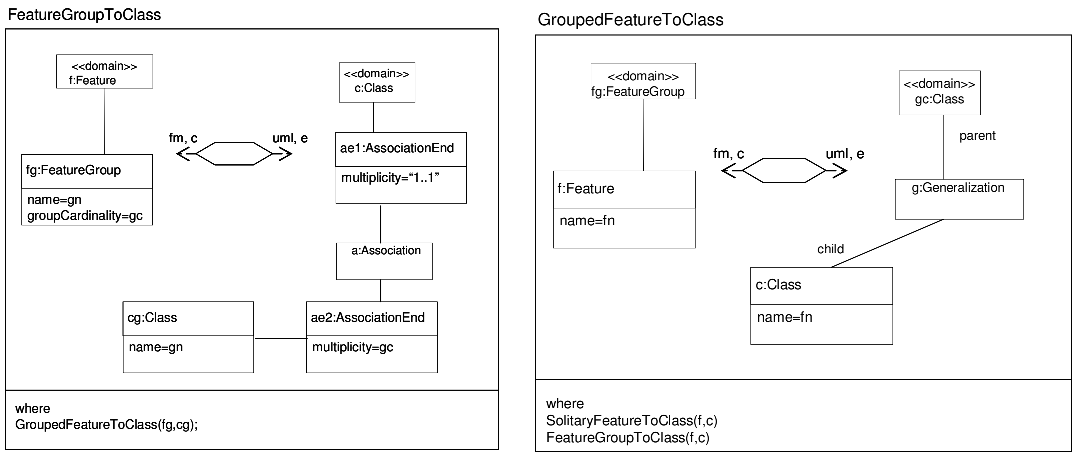

Fig. 3 Transformation definition of a Feature model into a UML/XMI model

Concerning the implementation details, the most common representation format of feature models is based in XML. In the UML side, the use of XMI is more and more frequent, in spite of compatibility issues. Therefore, XML is the basis of the representation and the transformation mechanism. The most straightforward implementations will use XLST style sheets or java tools to translate XML files directly or via DOM trees. The diversity and volatility of feature meta-models is an important issue. For this reason we have selected the XFeature (available in http://www.pnp-software.com/XFeature/) tool. It has some advantages, as the use of standard technology (XML and Eclipse) or the customizability of the feature meta-model (the tool allows users to define their own meta-models). The resulting UML model must be XMI-based; therefore any UML compliant case tool could be a valid option. A first implementation, using a XML style sheet, is given in [11]. The Figure 4 shows a feature model expressed with the XFeature tool, and the class diagram obtained from the resulting XMI file, after the application of the style sheet. The generated XMI file is imported into the Together CASE tool in a straightforward way. As XFeature, Together and the XSLT transformation engine are plug-ins of Eclipse, the integration of these tools is immediate. The image of Figure 4 shows the Together perspective. A simple change of perspective allows working with XSLT or XML/XMI files directly.

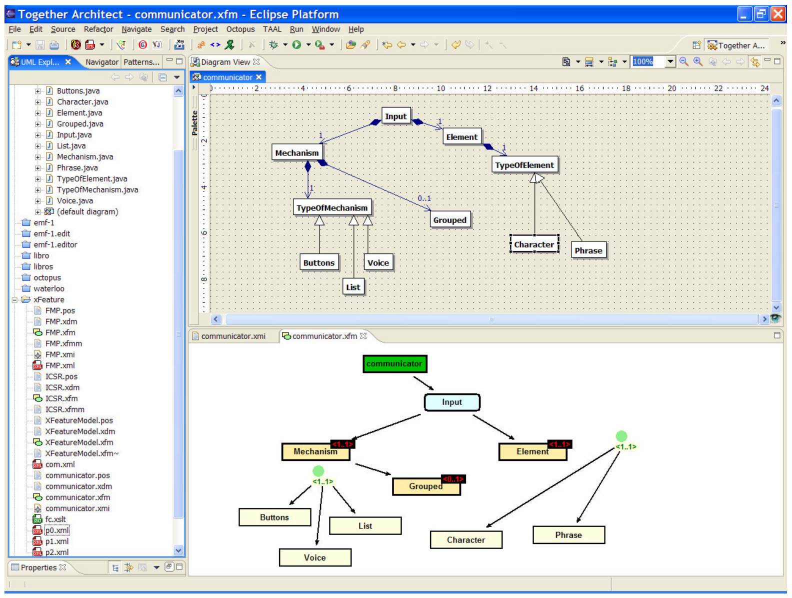

Fig. 4 A Feature model in XFeature tool and the Framework (automatically obtained using the implemented XML style sheet), integrated in the Eclipse platform

3 Conclusions and future work

In this article, the possibilities provided by new technologies such as MDA/MDD in the process of requirements elicitation and analysis are discussed in the context of product line development. The transformation of feature models in UML models has shown its possibilities.

The most immediate pending work comprises the inclusion of explicit traceability in the transformation specification and implementation. This approach implies in consequence the enhancement of the supporting meta-models.

References

1. Bass, L., Clements, P., Donohoe, P., McGregor, J. and Northrop, L. "*Fourth Product Line Practice Workshop Report*". Technical Report CMU/SEI-2000-TR-002 (ESC-TR-2000-002), Software Engineering Institute. Carnegie Mellon University, Pittsburgh, Pennsylvania 15213 (USA). 2000.
2. Bosch, J. "*Design & Use of Software Architectures. Adopting and Evolving a Product-Line Approach*". Addison-Wesley. 2000.
3. Chastek, G., Donohoe, P., Kang, K. C., Thiel, S. *"Product Line Analysis: A Practical Introduction"*. Technical Report CMU/SEI-2001-TR-001 ESC-TR-2001-001, Software Engineering Institute (Carnegie Mellon), Pittsburgh, PA 15213
4. Clements, Paul C. and Northrop, Linda. *"Software Product Lines: Practices and Patterns"*. SEI Series in Software Engineering, Addison-Wesley. 2001.
5. Krzystof Czarnecki and Ulrich W. Eisenecker, *"Generative Programming: Methods, Tools, and Applications*", Addison-Wesley, 2000
6. Krzysztof Czarnecki, Simon Helsen. *"Classification of Model Transformation Approaches"*. OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture, 2003.
7. K. Czarnecki, S. Helsen, and U. Eisenecker. *"Staged Configuration Through Specialization and Multi-Level Configuration of Feature Models"*. To appear in Software Process Improvement and Practice, 10(2), 2005.
8. García, F. J., Barras, J. A., Laguna, M.A., and Marqués, J. M. "*Product Line Variability Support by FORM and Mecano Model Integration*". In ACM Software Engineering Notes. 27(1);35-38. January 2002.
9. Kang, K. C., Kim, S., Lee, J. y Kim, K. "*FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures*". Annals of Software Engineering, 5:143-168. 1998.
10. K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson. *"Feature-Oriented Domain Analysis (FODA) Feasibility Study"*. Technical Report, CMU/SEI-90-TR-21, Software Engineering Institute (Carnegie Mellon), Pittsburgh, PA 15213
11. Miguel A. Laguna, and Bruno González-Baixauli, *"Goals and MDA in Product Line Requirements Engineering"*, Technical Report GIRO-2005-01, available in http://giro.infor.uva.es/docpub/giro-05-01.pdf
12. Miguel A. Laguna, Bruno González, Oscar López, F. J. García, *"Introducing Systematic Reuse in Mainstream Software Process",* IEEE Proocedings of EUROMICRO'2003, Antalya, Turkey, 2003.
13. J. Mylopoulos, L. Chung, and E. Yu. *"From object-oriented to goal-oriented requirements analysis"*. Communications of the ACM, 42(1):31–37, Jan. 1999.
14. Object Management Group, "*MDA Guide Version 1.0"*, 2003
15. Object Management Group and *QVT-Merge Group , "Revised submission for MOF 2.0 Query/View/Transformation version 2.0"* Object Management Group doc. *ad/2005-03-02, 2005.*