Object Oriented Feature Modeler (OOFM)

A System Family Modeling Tool

Peyman Shakari

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
MASTER OF SCIENCE

UNIVERSITETET I OSLO
Oslo, Norway-May 2nd. 2006
Summary

This thesis has presented the findings of a feature modeling notations survey and a feature modeling tools survey. I concluded that the current feature modeling tools are based upon FODA’s conceptual modeling notations, and are therefore not capable of understanding formal conceptual abstractions such as classes and objects. A new tool, OOFM, has been generated by the EMF using the UML 2.0 model of system families. We have seen that both this model (Variation model) and Java models are object oriented. The variation model uses the EMF meta-model elements such as EClass, EObject, EAttribute, and etc. These elements correspond to the Java elements IClass, IObject and IAttribute.

The fact that both the variation model and the Java models (Base Models) are object oriented makes the link between them possible. This means that the tool I have generated using EMF recognizes the Java model (Base Model) elements. Therefore it can be used to create features corresponding to the classes and objects of the Java model. The JDT technology has been used to implement the actual link between the two models. The technology has also been used to generate specific systems from the feature models.

We have seen that by combining EMF with JDT we can build a feature modeling tools that will recognize elements of a Java model. Such a tool support standard create, retrieve, update, and delete operations, and it also supports cardinality constraints, complex relationships and inheritance structures, containment definitions, and a suite of attribute descriptions. The generated code provides notification, referential integrity, and customizable persistence to XMI. All we have to do is to create an object model, from which we can generate the tool using EMF. Therefore it makes you wonder why spend so much money, time and effort on designing independent stand-alone tools based on concepts when you can use the exact tool as an Eclipse plug-in?
It is clear that the objective of these tools is to captures all elements and structures of the domain and not just objects as it is with OOFM. However, without a general and collective understanding of concepts, the world is shapeless, undefined and even more insecure. Recognizing concepts as objects will make it easy talk about them. Although it is naive to think that all concepts are objects.

ODM is a method that can be used to combine two or more conceptual abstractions (e.g., objects, functions, aspects, etc). In the future it may be an idea to focus on developing tools that support more than one conceptual abstraction, may be objects and functions at the same time.

In OOFM I have limited my work to the Ecore kernel model. There are other sides of the EMF model that can be interesting to look into, for example the behavioral features (EOperation) and the EReference. It is certainly possible to use these elements to build constraints and composition rules between features. Finally, the JFace and SWT can be used to provide powerful GUI’s that the users can use in order to implement constraints and so on. Therefore, feature modeling in Eclipse is meaningful modeling.