Introducing Product Lines through Open Source Tools

Øystein Haugen

SINTEF and University of Oslo
oystein.haugen@sintef.no

Abstract. We present an approach to introducing product lines to companies that lower their initial risk by applying open source tools and a smooth learning curve into the use and creation of domain specific modeling combined with standardized variability modeling.

Assumptions

Typically companies see that they have potential gains by being able to come to grips with the variability of their software. They want to define their variability, but they are not ready up front to buy expensive tools or to decide on a drastic change in how their software development is conducted. Furthermore, the company wants to continue to build on software in their legacy.

Thus we have the following elements that we need to cope with

1. Money. Reluctance to take decisions on buying expensive tools.
2. Process. Reluctance to make a drastic change in development process without certainty of the results
3. Transparency. A need at every point in the development to understand in adequate detail what is going on.

Language Engineering Workbench

The approach can be characterized as a language engineering workbench. The included tools are all open source tools and one focus is to create a domain specific language dedicated to express a small domain at the core of the are where there is the need to control the variability.

Our explanatory and real case is that of defining the control structures of train stations. Train stations are quite similar, but they are not identical. Train stations are equipped with a number of train signals and manipulators that together should ensure a safe and efficient journey through the train station.

Fig. 1 gives an overview of our language engineering workbench approach to handling the challenges of train stations.

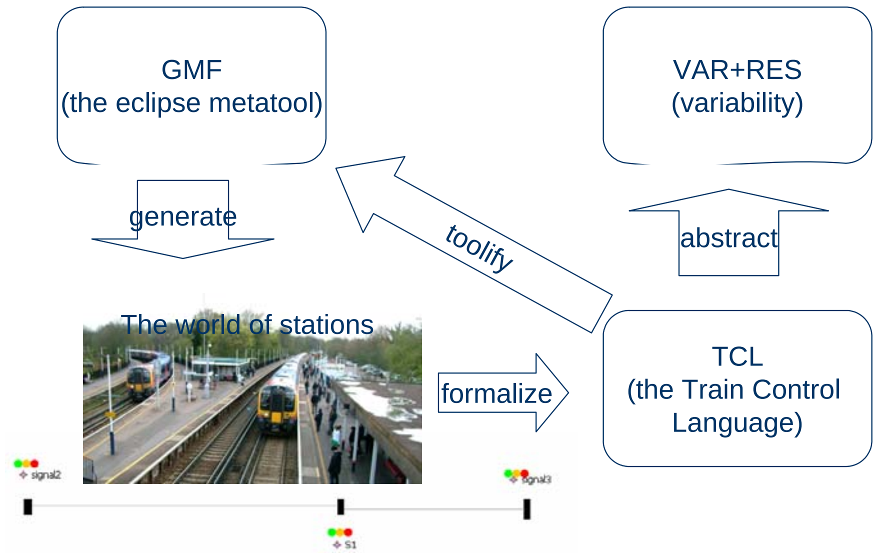

Fig. 1. The Language Engineering Workbench for train stations

Step 1: Define a domain specific language

In our case we started off to define a domain specific language for describing train stations. We call this the Train Control Language (TCL). This part of the process has several advantages:

- Defining a language increases the understanding of those who own the domain. It makes their knowledge more precise and ambiguities are eliminated.
- Assisting to define the language makes the language design experts have a steep learning curve relative to the domain.
- The defined language represents a well-defined common knowledge of the domain for the domain experts as well as the language experts.

Regardless of the success of the upcoming points, the process of defining a domain specific language is useful in itself since adequate knowledge has been achieved.

Step 2: Create a tool for the DSL

Having reached a reasonable initial definition of the domain specific language, we apply open source tools to formalize the definition in a metamodel. In our TCL case we are exploring the use of GMF (Eclipse Graphical Modeling Framework) where we in addition to the metamodel defines the graphical symbols of TCL and how these combine with the metamodel.

Our domain experts are then able to play with the generated TCL editor and may react to the definition of TCL on the basis of their experimentation.

Furthermore the domain experts are then invited to become language engineers themselves as we give courses in GMF modeling. For the domain experts of the company this is a threshold, but a rather small one. There is no big sums of money involved since the tools are free and our course is given under the umbrella of a joint research project. The resources our domain experts are obliged to offer are their time and dedicated effort.

Following the course the cooperation between the domain and language experts have become even tighter. The language experts now know more about the domain and the domain experts know more about language design. The domain specific language in turn improves, and this is when steps 3 and 4 may take place in parallel.

Step 3: Creating code generators

Having a reasonably stable language definition formalized in GMF such that an editor has been generated, it is time to formalize the semantics of the language. For domain specific languages the semantics is normally defined through their code generators. In our TCL case the first code generator will define tables of control gear setup. This will effectively validate the language and tools against already existing train station definitions.

Code generators are also made by means of open source software. In the TCL case we apply MOFscript, a tool for model to text transformation developed by SINTEF that is almost compliant with the OMG standard for model-to-text transformation.

Step 4: Defining the variability

Even without this step, the company has gained a lot if the other three steps have been successful. They may even be said to have a product line approach as the domain specific language defines the product line, and every model in the domain specific language defines the individual products, which in our case are the individual train station control setups.

But adding the variability increases the potentials even more. Our approach to variability modeling is to apply a standard approach to this. The MoSiS project has as its goal to define a domain specific language for variability that can be standardized through OMG.

From the definition of the variability language we may produce common tools for variability through GMF such that the same variability tool can be applied to define variability on several different domains. In our case the variability model will result in families of models in TCL each representing a variant of a train station.

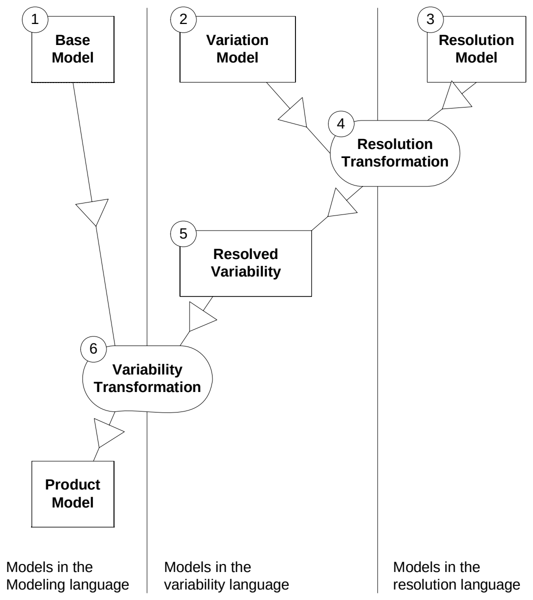
1
Base Model
2
Variation Model
3
Resolution Model
4
Resolution Transformation
5
Resolved Variability
6
Variability Transformation
Product Model
Models in the Modeling language
Models in the variability language
Models in the resolution language

Fig. 2. The process of resolving a variation model

In **Fig. 2** we describe how the variability model is bound by the information in a resolution model and that this resolved variability model is combined with the base model to yield a product model. In our case the base model consists of samples of train stations and the variability models define variants of those samples.

The code generation from the variability language is a generic transformation into models of the domain specific language, in our case the TCL. This has the positive effect that there is no need to augment the TCL code generators when introducing variability since the resolution of the variability will yield a plain TCL model.

Summary

Our Language Engineering Workbench approach to product lines is based on open source tools. The approach combines language design with applying a general and potentially standardized language for variability.

The advantages for our companies that apply this approach are the following:

1. Low risk and high transparency throughout the process
2. Open source tools with low cost makes experimentation easy
3. Open source tools that are controlled by SINTEF such as MOFscript makes the use even simpler since expertise is readily available
4. Variability modeling is kept separate from the creation of the domain specific language itself, but of course the variability models must take into account details of the underlying base models.