Language Engineering in Practice

Martin Große-Rhode

Dependable Systems Department, Fraunhofer ISST Berlin, Germany
Martin.Grosse-Rhode@isst.fraunhofer.de
URL: www.isst.fhg.de/~mgrosse

The Dependable Systems Department of the Fraunhofer ISST Berlin creates methods for the development, integration, and maintenance of embedded automotive systems. Based on a thourough analysis of the existing processes, domain- and enterprise-specific roles, activities, and artefacts are designed for an immediate enhancement of the technical, processual, and organsiational infrastructure that is found at the company. General principles that govern the design of the methods are continuous model-based engineering and domain engineering. Continuous model-based engineering first means to focus the development process on models whose structure is designed in such a way as to optimally support the activities and roles found in the foregoing analysis. Second, these models are interconnected via a common core, such that the transitions between the activities are also fully reflected at the model level. Thereby design decisions can be traced throughout the whole development process, consistency can be checked, and changes can be managed. Domain engineering aims at a systematic reuse by the generation of models or model templates as analysis, design, and implementation assets and techniques for their reuse within the development of new products.

As a means to define the abstract modelling language that determines the structure of the models that are to be used a two-step meta-modelling approach turned out as most adequate. In the first step class diagrams are used to introduce the modelling elements and their fundamental relationships. Since just classes, binary associations with multiplicities, and attributes are used any object-oriented modelling language or tool can be used for that. To make the meta-model complete constraints have to be added that define the relationships of the modelling elements more precisely. In principle any logic or object constraint language can be used for that purpose. We have chosen the object-oriented extension ObjectZ of the set-theoretic specification language Z, although it implied to reformulate the whole class diagram developed before within the constraint set. This decision was based essentially on the clarity of the language, the time constraints of the projects, and the previous knowledge of the team members.

Within the development and integration process the following main views are distinguished and supported by appropriate models: requirements, logical architecture, and technical architecture including hardware and software architecture. As mentioned above, also the interconnections of the views are specified by models. That means that there are realisation models that connect different requirements models (like user requirements, legal requirements, or system

Dagstuhl Seminar Proceedings 04101
http://drops.dagstuhl.de/opus/volltexte/2005/15

requirements) among each other and with the other models (logical and technical architecture), and that there are models for the partitioning of the logical components onto the technical (software and hardware) components.

Flexibility in the design of the modelling languages (and thus the models) is achieved by a layered meta-modelling approach. Thereby the modelling elements are introduced within a hierarchy, with most general elements at the top and stepwise refinements to introduce more domain-, enterprise-, and role-specific concepts. Concerning requirements models for example at the top layer the general structure of requriements is defined: a requrirement consists of a subject or stake holder (the one who requires the feature), a feature that is required, possibly a mode (the feature must or may be included), and an object or target, i.e. the system or the group of systems under development that shall have this feature. These constituents can than be refined, for example by saying which groups of stake holders are relevant, which kinds of featuers are to be distinguished (functional, non-functional, safety, etc.), which modes are there, which kinds of groups of systems are considered and how are they described. Analoguously the modelling concepts for the other views and the modelling concepts for the domain models that capture the reusable analysis, design, and implementation assets are introduced. The experience of our projects with industrial partners shows that this kind of language engineering is an adequate means for the introduction of continuous model-based software development processes in the industrial practice.

Technical reports on method developments including the corresponding metamodels will be made available soon via the web page of the author.