Tobias Brückmann

Model Driven Development and Maintenance of Business Logic for Information Systems

Abstract

Since information systems become more and more important in today's society, business firms, organizations, and individuals rely on these systems to manage their daily business and social activities. The dependency of possibly critical business processes on complex IT systems requires a strategy that supports IT departments in reducing the time needed to implement changed or new domain requirements of functional departments. In this context, software models help to manage system's complexity and provide a tool for communication and documentation purposes. Moreover, software engineers tend to use automated software model processing such as code generation to improve development and maintenance processes. Particularly in the context of web-based information systems, a number of model driven approaches were developed. However, we believe that compared to the user interface layer and the persistency layer, there could be a better support of consistent approaches providing a suitable architecture for the consistent model driven development of business logic.

To ameliorate this situation, we developed an architectural blueprint consisting of meta models, tools, and a method support for model driven development and maintenance of business logic from analysis until system maintenance. This blueprint, which we call Amabulo infrastructure, consists of five layers and provides concepts and tools to set up and apply concrete infrastructures for model driven development projects. Modeling languages can be applied as needed. In this thesis we focus on business logic layers of J2EE applications. However, concrete code generation rules can be adapted easily for different target platforms.

After providing a high-level overview of our Amabulo infrastructure, we describe its layers in detail: The Visual Model Layer is responsible for all visual modeling tasks. For this purpose, we discuss requirements for visual software models for business logic, analyze several visual modeling languages concerning their usefulness, and provide an UML profile for business logic models.

The Abstract Model Layer provides an abstract view on the business logic model in the form of a domain specific model, which we call Amabulo model. An Amabulo model is reduced to pure logical information concerning business logic aspects. It focuses on information that is relevant for the code generation. For this purpose, an Amabulo model integrates model elements for process modeling, state modeling, and structural modeling. It is used as a common interface between visual modeling languages and code generators. Visual models of the Visual Model Layer are automatically transformed into an Amabulo model.

The Abstract System Layer provides a formal view onto the system in the form of a Coloured Petri Net (CPN). A Coloured Petri Net representation of the modeled business logic is a formal structure and independent of the actual business logic implementation. After an Amabulo model is automatically transformed into a CPN, it can be analyzed and simulated before any line of code is generated.

The Code Generation Layer is responsible for code generation. To support the design and implementation of project-specific code generators, we discuss several aspects of code integration issues and provide object-oriented design approaches to tackle the issues. Then, we provide a conceptual mapping of Amabulo model elements into architectural elements of a J2EE infrastructure. This mapping explicitly considers robustness features, which support a later manual integration of generated critical code artifacts and external systems. The Application Layer is the target layer of an Amabulo infrastructure and comprises generated code artifacts. These artifacts are instances of a specific target platform specification, and they can be modified for integration purposes with development tools.

Through the contributions in this thesis, we aim to provide an integrated set of solutions to support an efficient model driven development and maintenance process for the business logic of information systems. Therefore, we provide a consistent infrastructure blueprint that considers modeling tasks, model analysis tasks, and code generation tasks. As a result, we see potential for reducing the development and maintenance efforts for changed domain requirements and simultaneously guaranteeing robustness and maintainability even after several changes.