Model Driven Development of Distributed Business Applications

Wolfgang Goerigk

b+m Informatik AG
Rotenhofer Weg 20, D-24109 Melsdorf, Germany
wolfgang.goerigk@bmiag.de

Abstract

The present paper presents a model driven generative approach to the design and implementation of destributed business applications, which consequently and systematically implements many years of MDSD experience for the software engineering of large application development projects in an industrial context.

Keywords and phrases MDSD, Software Architecture, Modelling, Code Generation

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.211

1 Introduction

MDSD (Model-Driven Software Development) is a generic term covering methods and techniques used to automatically generate executable software or other software related artefacts from formal models [4].

The MDSD platform b+m gear Java (gear) is the result of many years of experience with enterprise application development using various MDSD tool chains. gear supports the classical 3 tier architectural layers, *i.e.* entity, service and front end, and also a workflow partition similar to the BPMN (Business Process Modeling Notation). gear focusses on business software and in particular on client/server applications, and it is designed to support several non functional and technical requirements like *e.g.* safe distributed transaction handling or business driven historization. b+m gear Java is based on the Eclipse Modeling Project [2] and the well known generator framework openArchitectureware [3], which has originally been initiated and founded by the b+m Informatik AG. Since 2003 it is an open source framework and it is now part of the Eclipse Modeling Project.

2 Model Driven Development Using b+m gear Java

A comprehensive description of MDSD can be found in [4]. In the talk we will discuss the model driven software engineering approch and its maturity and state of the art in more detail. In the paper we want to focus on modeling and some technical advantages of the architecture centric generative software development process. Figure 1 depicts the so far supported DSL partitions and underlying platforms of gear:

Entity is used to model the persistence layer. OR mappers (like *e.g.* Hibernate) are used to generate database schemata and access code. The entity partition declaratively supports historization (cf. below).

Service is used to model business services and entity access. The service partition guarantees safe distributed transactions and uses the Java Transaction API (cf. below).

Frontend is used to model screen flows, which are coupled synchronously or asynchronously. From screen flow models, JavaServer Faces (JSF) and Spring WebFlows are generated.

Workflow is used for business service orchestration. It also organizes persistent tasks and their user and rôle assignments. For Workflow, the underlying jBPM platform is supplemented by task management, distribution strategies and enhanced by concepts like delegation, escalation, resubmission, task forward.

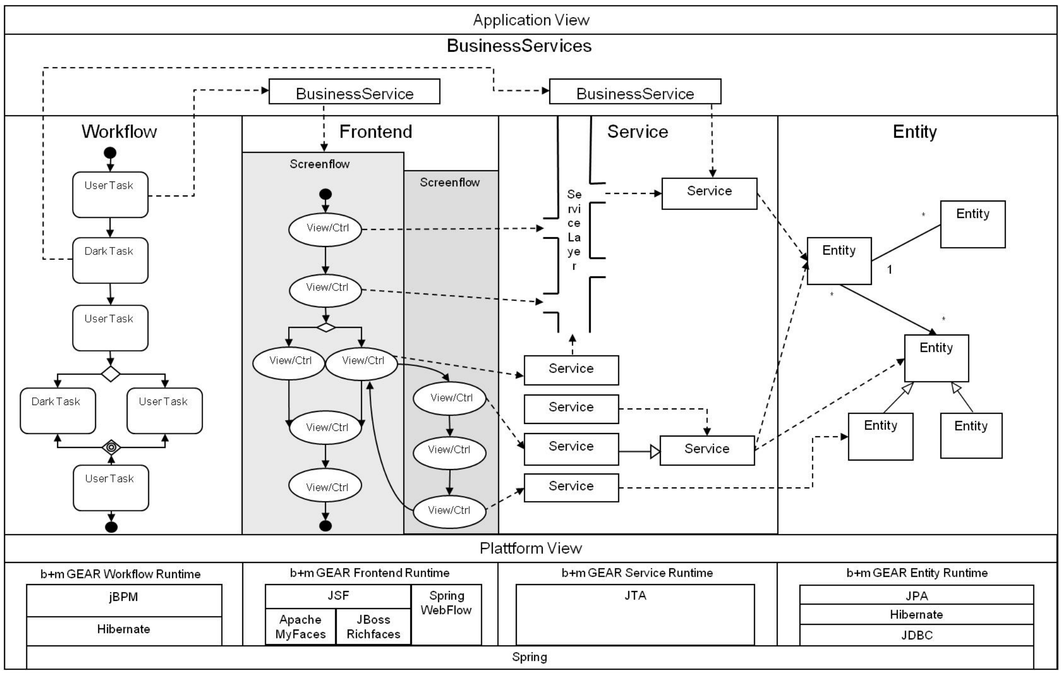

Figure 1 DSL partition map and underlying platforms in gear

Workflow and its formal and orthogonal modeling and generative support, as well as of business rules and underlying rule engines, become increasingly important. gear supports workflow. The present development for b+m gear Java aims at the BPMN 2 standard [1], and the integration of a rule engine is planned in the future as well. Modeling is supported by grafical editors within the Eclipse IDE; the syntax resembles UML (Unified Modelking Language). However, for practical and pragmatical reasons we prefer a more light weight approach using specifically tailored domain specific languages.

We want to illustrate technical advantages of the MDSD approch using some examples: **Historization** is often used in order to guarantee, that software systems *e.g.* in the finance or insurance domain are audit-proof – often required by law. Every transaction has to be plotted, which requires additional history tables for entities, and complicated write access has to be weaved horizontally into the entire code. In gear, historization is a simple annotation of entity definitions, and weaved generatively. An equally prominent example is **transaction safety**, necessary for distributed client/server applications with persistent data. In gear, the service partition uses the Java Transaction API. In both cases, sophisticated and concise architectural enhancements for DSL and generator support the mastery of potentially complex horizontally weaved code aspects. Quality and implementation efficiency are improved.

References

1 Business Process Model and Notation (BPMN), Version 2.0. `http://www.omg.org/spec/`

`BPMN/2.0`, 2010.

2 Eclipse Modeling Project. `http://www.eclipse.org/modeling/`, 2010.

3 openarchitectureware.org. `http://www.openarchitectureware.org/`, 2010.

4 T. Stahl, M. Völter, S. Efftinge, and A. Haase. *Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management.* dpunkt.verlag, 2 edition, 2007.