Model Driven Software Engineering in the Large

Experiences at the Dutch Tax and Customs Service (Industry Talk)

Betsy Pepels, Gert Veldhuijzen van Zanten
Dutch Tax and Customs Administration (DTCA), the Netherlands
{ejh.pepels, ge.veldhuijzen.van.zanten}@belastingdienst.nl

Abstract

Model Driven Software Engineering is a discipline that only recently has begun to be used in mainstream practise. At the DTCA, we use this approach successfully for a major part of the Dutch Social Benefits system, a nationwide online system serving 6 million citizens. We report on our experiences with bringing forth and maintaining this system, and on our future plans for extending this approach to other key systems of our organisation. The DTCA is responsible for collecting taxes and paying social benefits. The supporting software systems typically have a lot of functionality and process huge volumes of data. Furthermore, the systems should be easy to adapt within short notice: laws do change often, and once a change to a law has been approved by the parliament, there is only little time to implement the changes. To adapt complex systems quickly, we use Functional Model Driven Development (FMDD), a variant of Model Driven Software Engineering. The FMDD approach separates functionality from other aspects of an application, hence its name. We outline the basic way of working by describing the main artefacts and roles.

Categories and Subject Descriptors D.2.1 [**Software Engineering**]: Requirements/Specifications - languages, methodologies, tools; D.2.9 [**Management**]: Life cycle; D.2.13 [**Reusable Software**]: Domain engineering;

Keywords industrial experience; software language engineering, model-driven software engineering

1. Language Workbench

A key feature of our approach is *separation of concerns*. We organise the specifications and transformations such that different concerns are separated from each other. We separate the concerns by offering a separate language for each separate concern. This way we can separate the maintenance cycle of the concerns.

Separating functionality of an application from its technical implementation Our approach separates functionality from technical implementation. The Domain Engineers focus on the functionality by making and maintaining the specifications which define the functionality of the application. The Transformation Engineers focus on the technical aspects of the application.

Separating time aspects from specifications An important feature of administrative systems is the ability to deal with retroactive (and other time dependent) computations. The specifications in our BSL don't have time aspects: the Domain Engineers focus on functionality solely. When executed, a richer meaning is assigned to the specifications by adding time information, a strategy that in the functional programming world is known as "lifting".

Business Specific Language Business concerns are expressed in a Business Specific Language. It is largely declarative, and abstracts from technical issues like persistence The BSL is not defined once, but evolves according to the changing domain.

Functional specifications Using the BSL, the Domain Engineers define functionality by creating functional specifications. These functional specifications are the foundation of many artefacts to be generated.

Transformations On basis of the BSL, the Transformation Engineers design and build the software that transforms the functional specifications into all kinds of artefacts, in particular into executable source code and documentation. We furthermore generate artefacts like a test suite, database creation and upgrade scripts, the event catalogue, and interfaces.

Business orientation The focus of FMDD is on the elucidation, specification and maintenance of *business knowledge*, in a language that is close to the business itself. This approach increases flexibility and maintainability of the system and business.

Installing an automated software product line The aim of our approach is not to build and maintain the application, but to install an automated software product line, a "factory", in which as much as possible of the development process is automated with generators, scripting, automated tests etc.

Separating legislation from administration and communication We are currently designing languages in which we aim to separate the specification of the legislation from concerns that have to do with administration and communication (data)

Language Workbench As language workbench we use the Jet-brains Meta Programming System (MPS), which is open source and based on IntelliJ and Java.

ITSLE'16, October 31, 2016, Amsterdam, Netherlands
ACM. 978-1-4503-4646-7/16/10...$15.00
http://dx.doi.org/10.1145/2998407.3001933