Quantools: A MDA transformation approach

Paulo Lumertz 1

Ana Paula Terra Bacelo 2

Toacy Oliveira 3

Abstract: Model driven architecture (MDA) represents a challenge for the companies to increase the benefits of code generation, creating systems by using common libraries and standards. This paper presents the Quantools that is a result of a 5-year project, introducing the benefits of MDA in a Brazilian software development company. Quantools is based on the concept of cartridges that specify the transformation rules and constraints for a particular domain. The tool may be integrated to the modeling activities of the system to check the correctness of models according to the standards previously defined. The Quantools and its Cartridges execute transformations to generate the source code in a particular domain.

1 Quantiza Systems
Av. Dos Municípios, 5510 Santa Lúcia Campo Bom – RS - Brazil ZIP 93700-000
`{lumertz@quantiza.com}`
2 Pontifical Catholic of Rio Grande do Sul – Faculty of Informatics
Av. Ipiranga, 6681 – Prédio 32 – 6 Floor - Porto Alegre – RS – Brazil ZIP 90619- 900
`{ana.bacelo@pucrs.br}`
3 University of Waterloo - David R. Cheriton School of Computer Science
200 university Ave N2L 6P2 - Waterloo, - Canadá
`{toacy@acm.org}`

1 Introduction

Software development companies have recognized the importance of reuse policies during the systems development to decrease costs, optimize resources and increase the productivity and software quality. A possible solution to this problem is proposed by OMG, through the Model Driven Architecture (MDA) framework [3]. Through MDA the developers invest more time in the requirements modeling activities and spend less time with their implementation issues. This paper presents a transformation tool based on MDA which was developed through a research project between a software company team and academic researchers.

2 Motivation and Initial Steps

In the last twenty years, the company has developed software for different domains such as footwear industry, farms, hotels, accounting etc. Although the domains are quite different, the software development team has observed many similarities in terms of human interaction infrastructure, database access and access control. Considering this scenario, the company decided to invest in Research & Development projects to build a basic infra-structure to generate applications for different domains, increasing software development productivity and improving further maintenance of such systems. Five years ago, the company's software scenario was the following:

- There were approximately 60 software products in development and maintenance;
- The majority of these projects only invested in documentation at the beginning of the process. Therefore, this documentation became obsolete in the next versions. In this context, the source code was responsible for representing business rules and for documentation of the updates done during the software life cycle;
- It was also observed an increase of time to perform impact analysis before developing new requirements of the existing products. There was no traceability among the requirements and the implemented features in a specific technology;
- A lack of a consistent use of a basic infra-structure for the development of each product consequently generated an independent infra-structure for each one;

Based on this outcome a layer generation architecture based on MDA was proposed.

3 Quantools Architecture

The Quantools Architecture is organized in layers, which receive an XMI file describing models created during the software modeling process and generate a source code in a specific technology in accordance with the cartridge chosen. Figure 1 shows the four main layers of the Quantools Architecture and the Cartridge responsible for the specification of the rules and policies regarding the specific application to be generated.

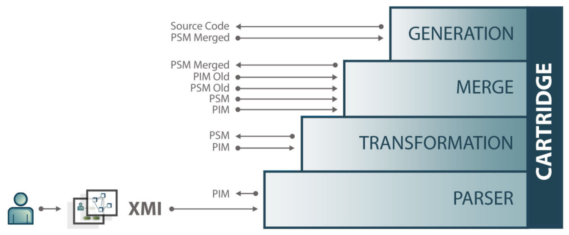

Figure 1: Quantools Architecture

The **cartridge** is the component that defines a profile (e.g. ER Profile) where the guidelines for a modeling application in a specific technology are proposed. As a complement of this profile, a set of rules and merging policies that must be adopted during the transformation is defined. For each technology there must be a cartridge which specifies its profile and its respective rules. For example, there is a cartridge that defines the rules about the transformation of OO models and merging policies among PSM-PIM-PSM models. Considering this cartridge it is possible to define some rules referring to the creation of classes, tables, windows (GUI), attributes and methods. An example of merging policies is to compare if a method or attribute of an old PSM was removed or updated in the current version of the PSM. The cartridge implements different transformation, as follows:

- **Transformation PIM to PSM**: The role of a rule is to identify patterns and elements in the PIM metamodels and to execute a transformation to create elements in the PSM model. For instance, the rule may look for some stereotypes that define screens and generate them in a PSM model.
- **Transformation PSM to PSM**: There are some rules that search for elements of PSM model to generate one or more elements of the same model, for instance, the generation of JPA mapping. To perform this transformation it is necessary the previous creation of PSM Class. This transformation does not depend on the rules of the cartridge, as the execution is done into the Quantools itself. Figure 2 shows examples of rules implemented for a cartridge.

Besides the cartridge, the Quantools presents some layers that will be explained bellow as shown is Figure 1. The first layer is called **Parser**. This layer is responsible for the input of the transformation process through an XMI file in 2.1 version. The XMI file may contain different models created during the modeling process, such as class models, requirements, use cases, interface model etc. DOM4J [2] is used in this layer to read the XML file and afterwards to create a consistent PIM Metamodels. Using the XMI file, through the DOM4J, the parser will create the PIM Metamodel which is composed by metamodels of different artifacts produced during the modeling process (e.g. the class metamodel, use case metamodel).

The second layer – **Transformation** - is responsible for the execution of transformation rules. The Quantools defines only structural classes to facilitate the creation of the rules which must be realized by the cartridge. In this context, the transformation layer presents some basic rules which may be used by all cartridges according to their needs. These generic rules can be obtained through the transformation layer. If a specific rule or policy needs to be used, we need to create a cartridge to realize them.

The third layer – **Merge** – will be executed only when there is a PSM but at same time a new PSM needs to be created due to the updating of some requirements or the creation of new ones. Thus, the Merge layer brings the merge policies established by the cartridge and generates a new PSM merged. An example of policy defined by the Cartridge is the setting up of a method that cannot be updated by the programmer. The establishment of these restrictions can be defined in any part of the metamodel as attributes, operations or in all parts of the class. This functionality is implemented through the “resource owner” concept.

The fourth layer - **Generation** – is responsible for the generation of the source code defined by a specific cartridge, base on the last PSM generated during the transformation.

4 Final Considerations

The Quantools architecture is presented in order to allow models created independently of any platform, to be transformed into specific models for a given platform. The tool is supported by the architecture of layers developed with a set of Java Technologies.

Nowadays, the Quantools architecture has been used for the construction of new software products at the company. For each application or family of applications, new cartridges have been constructed. Therefore, it has been created a set of guidelines to support the creation of the new cartridges, which the goal is to reuse them as much as possible according to the basic rules of the existent cartridges. A Quantools demo is available at http://www.quantiza.com/quantools.

References:

DOM4J. Available at http://www.dom4j.org/, 10/03/2007.

Java API. Java Application Program Interface, Available at www.java.sun.com. 07/03/2006.

Kleppe, A., Warmer, J., Bast, W. MDA Explained: The Model Driven Architecture – Practice and Promise. Boston, MA, Addison-Wesley, 2003. (2003)

XMI - XML Metadata Interchange (XMI) Specification, v1.2, OMG. (2009)