# Bridging BPM and MDE: On the Integration of BiZZdesigner and OptimalJ

Henk Jonkers, Maarten W.A. Steen, Lex Heerink, Diederik van Leeuwen

Telematica Instituut, P.O. Box 589, 7500 AN Enschede, The Netherlands
{Henk.Jonkers, Maarten.Steen, Lex.Heerink, Diederik.vanLeeuwen}@telin.nl

**Abstract.** Our challenge was to develop tool support for the complete design trajectory from business processes to the code of the enterprise applications supporting them. To this end we integrated two commercial modelling tools: BiZZdesigner and OptimalJ. BiZZdesigner supports the modelling and analysis of business processes and uses a dedicated graphical language. OptimalJ is an Eclipse-based implementation of the Model-Driven Architecture (MDA). In this paper we describe why and how we integrated these modelling tools through a model transformation specified in QVT.

## 1 Introduction

Enterprise applications are designed to support enterprise's business processes. Once a business process has been designed or changed, requirements are extracted for the supporting software applications and a software design trajectory is started. Unfortunately, business processes tend to change more and more frequently. So frequent, that the time between subsequent changes is rapidly becoming shorter than the time it takes to develop or update the supporting software applications. Thus the company's capability to respond to changes in the market is hindered by the IT department's inability to create or update the supporting enterprise applications in time.

In response to this challenge we integrated a dedicated business process management (BPM) tool with a tool supporting the Model-Driven Engineering (MDE) of enterprise applications. In this way we obtain an integrated tool-chain that supports and speeds up the entire design trajectory for enterprise applications from the business process design to the running code.

## 2 Tool Selection

The MDE and BPM communities are currently served by different tools from different vendors. There is a wide range of both commercial and open-source offerings for MDE available. The BPM community is mainly served by a range of domain specific commercial offerings.

Eclipse-based modelling tools are offering an increasingly attractive environment for Model-Driven Engineering (MDE) of software artefacts. The Eclipse Modelling

Project (EMP) is slowly producing open source tools that support meta-model definition, generation of model editors, transformation specification and execution, specification and execution of code generation templates, etc. However, there is still some way to go before a mature and integrated environment for MDE is realised.

Some commercial tool vendors have marched ahead of the pack and produced integrated environments that support a large part of the MDE development trajectory even though they may not support the latest meta-modelling and transformation standards yet. Compuware's OptimalJ is one such tool that is built on the Eclipse platform. It supports PIM and PSM-level modelling using variants of the UML. It also supports model-to-model transformations and code generation. The Architect Edition can also be used to define your own meta-models using MOF 1.4, your own transformations, and your own code generation templates.

Although the EMP provides some support for defining your own domain specific modelling languages (DSLs), the generated tools generally fall far short of the required stability, functionality and look-and-feel. In addition, we believe that domain specialists, such as business process architects, are unlikely to change the way they work and to change to other tools.

In order to reach our objective of including business process design into the model-driven development of enterprise applications, we decided therefore to select dedicated tools for each of the disciplines and integrate them loosely through a model transformation. More specifically we have selected Compuware's OptimalJ as a tool for MDE and BiZZdesign's BiZZdesigner for BPM. Both tools represent the state-of-the-art in their respective domains. Below we provide short introductions on these tools.

### 2.1 Model-Driven Engineering with OptimalJ

Compuware's OptimalJ is a pragmatic implementation of OMG's Model-Driven Architecture (MDA) and model-driven engineering principles. OptimalJ enables the rapid design, development, and deployment of J2EE applications. It uses a visual development approach and transformations to translate domain models automatically into working applications (Compuware, 2006).

In accordance with MDA, OptimalJ provides a clear distinction between the models of the domain (PIM) and the models of application components (PSM), as illustrated in Figure 1. The models in OptimalJ have the following abstraction levels:

- **Domain model**—defines the business domain without technology-specific detail. It is composed of the class, process and service models.
- **Application model**—defines the application, based on a certain technology (e.g., J2EE), but without any coding detail. When J2EE is selected as target platform it consists of DBMS (persistence layer), EJBs (business logic layer), and Web (presentation layer) models.
- **Code model**—defines the application as compilable and executable code.

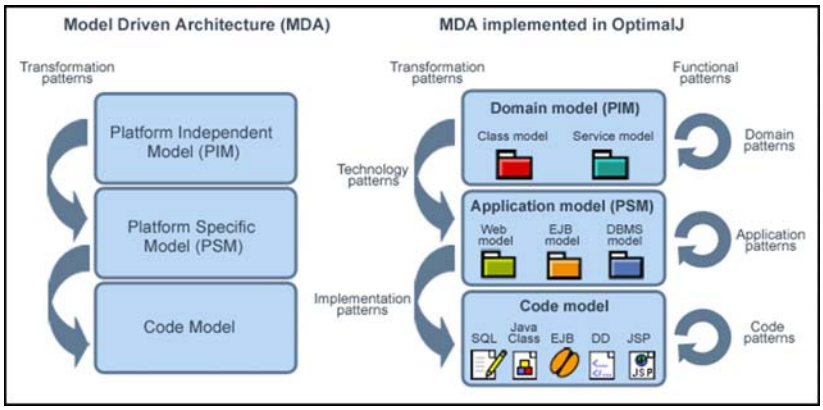


Figure 1. Model-Driven Engineering with OptimalJ

Transformations link the three modelling levels. The transformations are specified in terms of technology patterns and implementation patterns. Technology patterns map the Domain Model to the Application Model. Implementation patterns generate the code from the application models. Compuware is one of the first companies providing an implementation of the emerging OMG standard for specifying model transformations, Query/View/ Transformations (QVT; Object Management Group, 2005). QVT defines three different languages to specify transformations: core, relational and operational. An early access release of OptimalJ, made available to us by Compuware, supports the QVT core language. Although the technology patterns are currently still implemented using a proprietary transformation language, we have made use of the QVT support for mapping the business process model onto the Domain Model.

### 2.2 Business Process Management with BiZZdesigner

BiZZdesigner is a software tool for designing, analysing, and documenting business processes, work instructions, and related information. BiZZdesigner makes use of a graphical modeling language that was specifically designed for business process architects and designers, not for software engineers. For a precise definition of this language, called Amber, see (Eertink *et al.*, 1999). BiZZdesigner supports different views for modelling the structural, behavioural and information aspects of business processes, which are related to each other.

- The *structural view*, represented by *Actor Schemes*, is used to model the resources deployed for carrying out business activities, i.e., the actors, the systems and their roles and relations. Currently, we do not consider this view for the mapping to software models.
- The *behavioural view*, represented by *Behaviour Schemes*, is used to model the business activities (actions, interactions, and triggers) that make up the business process and their causal relations.
- The *information view*, represented by *Data Type Schemes* and *Item Schemes*, is used to model the data objects that are used or manipulated in the business activities.

## 3 Illustrative Example

In this section we introduce an example business process in the BiZZdesigner notation that will be used to illustrate our approach to map a business process design to a PIM for enterprise applications in Section 4.

The example represents an online product ordering process that consists of several activities. The ordering process is triggered by an event called ‘start’, after which the customer can order products by entering an online order. Subsequently, the order is processed and the required products are compared with the available products in stock. If all required products are available the order is approved, the stock is updated and the order is shipped. If none of the required products are available the order is cancelled and the customer is notified about the cancellation. If only some, but not all, of the products in the order are available, the order is modified so that it contains only the products that are available. The customer is asked to accept the (modified) order. If the customer accepts the order the stock is updated and the products are shipped. If the customer rejects the order, the order is cancelled and the customer is notified. Figure 2 depicts the corresponding BiZZdesigner Behaviour Scheme.

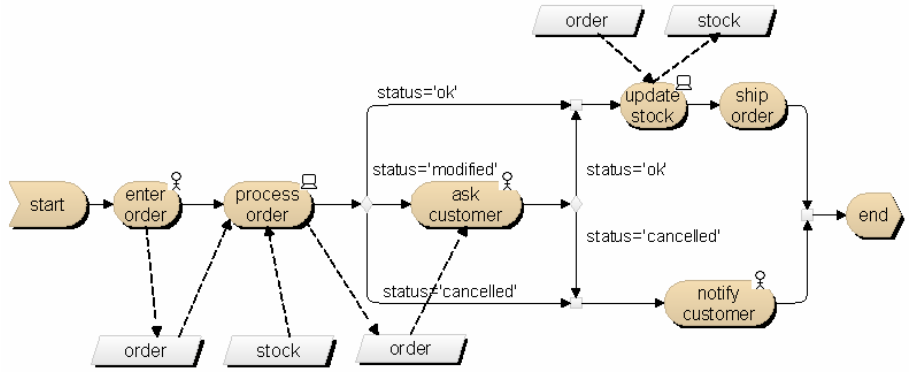


Figure 2. BiZZdesigner model of the online order handling process (Behaviour Scheme)

The Behaviour Scheme defines process activities and their relations. In Figure 2 three types of activities can be recognised: fully automated activities (‘process order’ and ‘update stock’), which are adorned with the -symbol; activities that require user interaction (‘enter order’, ‘ask customer’, and ‘notify customer’), adorned with the -symbol; and a fully manual activity (‘ship order’). The type of activity can be specified as a property in the Behaviour Scheme.

An Item Scheme defines the items that are involved in particular activities, and refers to data types that are defined in the Data Type Scheme. For example, the ‘enter order’ activity can create an ‘order’ item (of type ‘Order’) that can be used in the ‘process order’ activity. The process model of Figure 2 refers to two items from the Item Scheme: ‘order’ and ‘stock’.

The Data Type Scheme specifies the data types that are used in Figure 2. The Data Type Scheme for the online order handling example is depicted in Figure 3. An ‘Order’ has a status and an order number, and consists of one or more ‘Orderlines’. An

‘Orderline’ has a line number and a quantity, and refers to a ‘Product’. A ‘Product’ has a supplier and a product number. Furthermore, ‘Stock’ contains ‘StockItem’s and each ‘StockItem’ refers to a ‘Product’.

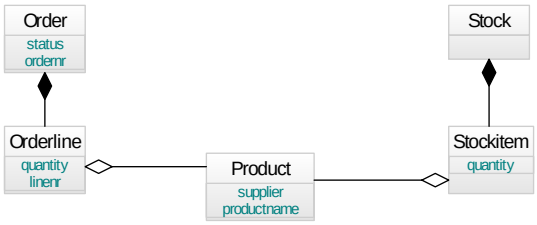


Figure 3. Data Type Scheme for the online ordering process

## 4 Approach

Our approach to incorporating business process designs into the model-driven development of enterprise applications to support them is illustrated in Figure 4. In this section we describe the steps that are required to automatically generate a running web application, starting with a business process model specified in BiZZdesigner. For this purpose, OptimalJ is extended with functionality to import business process models from BiZZdesigner. In order to make this possible, a MOF meta-model of the BiZZdesigner language has been specified in OptimalJ. Also, BiZZdesigner has been extended with a function to export models in the native XML format as XMI (XML Metadata Interchange) files, using an XSLT transformation.

Once the BiZZdesigner model is imported in OptimalJ, we transform it into an OptimalJ Domain Model, using the core language of the QVT. The required transformations are described in Section 4.1 below. Subsequently, we apply the default transformations provided by the Professional Edition of OptimalJ to generate the platform-specific Application Model and Java code.

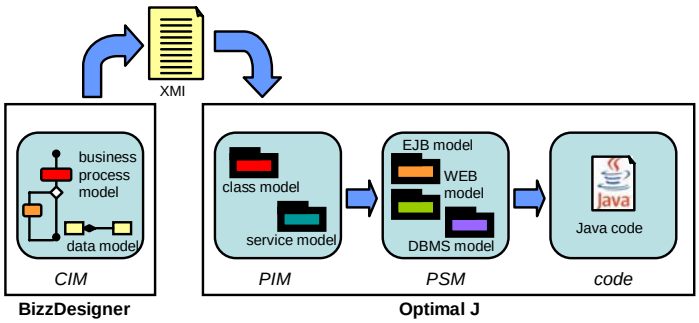


Figure 4. Extended MDE chain

### 4.1 BiZZdesigner to OptimalJ Domain Model transformation

The objective is to generate an OptimalJ Domain Model from a BiZZdesigner model through QVT transformations. Both the source model and the target model consist of a number of sub-models that are interrelated. A BiZZdesigner model consists of a Behaviour Scheme, an Item Scheme and a Data Type Scheme. These schemes are interrelated as described in Section 2.2. The generated OptimalJ Domain Model consists of a Process model, a Domain Services model and a Domain Class model. Activities in the Process model refer to services and operations in the Domain Services model; services in the Domain Services model, in turn, refer to classes in the Domain Class model. These dependencies also result in dependencies between the transformations.

In the transformation specifications, the target model is leading: for each of the three target models, a separate QVT transformation has been specified. Each of these transformations uses elements from one or more of the BiZZdesigner modelling schemes. Figure 5 shows which are the source and target models for each of these transformations.

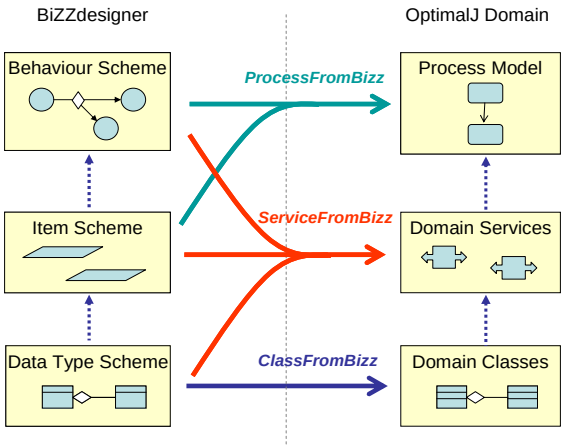


Figure 5. Overview of transformations

The transformation ‘ClassFromBizz’ derives a Domain Class model from the BiZZdesigner Data Type Scheme. The transformation ‘ProcessFromBizz’ derives a Process Model from both the Behaviour Scheme and the Item Scheme. Finally, the transformation ‘ServiceFromBizz’ derives the Domain Services Model from the BiZZdesigner Behaviour, Item and Data Type schemes. For a detailed description and definition of these transformations see (Jonkers, 2007).

### 4.2 From OptimalJ Domain Model to running code

The platform-specific Application model in OptimalJ consists of a large number of submodels, organised according to a three-tier J2EE application structure. For many of these models, graphical editors are available in OptimalJ, e.g.:

- Web Flow Diagrams in the presentation layer.
- UML Component Diagrams in the business layer.
- Relational Database Schemata in the data layer.

Typically, a diagram in the Domain model affects multiple diagrams in the Application model. E.g., a class from the Domain Class models results in a maintenance component in the presentation layer, business logic components in the business layer and a table in the relational database scheme in the data layer.

Transformations from the Domain model to the Application model are readily available in the Professional Edition of OptimalJ. These transformations, which are called technology patterns, have been implemented in a proprietary model-to-model transformation language of Compuware, not yet in QVT. Subsequently, model-to-text transformations, or implementation patterns, implemented in the proprietary language TPL (Template Pattern language), are applied to derive the J2EE Java code.

The only manual step that is still required is to write the Java code to implement the business logic for the fully automated tasks, in our example ProcessOrder and UpdateStock.

### 4.3 Scalability of the approach

Although we have illustrated our approach with a small example application, we believe that scaling up the problem to a realistic size will not cause any major problems. For example, if the BiZZdesigner model consists of multiple processes sharing the same data, the transformation will also produce multiple processes in OptimalJ's Process model sharing the same DomainClasses. The final result is a web application that also supports multiple processes.

An important prerequisite for scalability is the support for the incremental generation of target models in OptimalJ's QVT engine. This is indispensible in a realistic, iterative model-driven design process. Because of this incrementality, changes at the business process level are propagated to the PIM and PSM levels, without having to regenerate the models from scratch. Also, changes that have been made at the lower levels are preserved.

## 5 Conclusions

Model-Driven Architecture has introduced PIMs and PSMs as formalisations of the architecture and design of software applications. These formalisations enable the definition of model transformations and code generation to automate (part of) the software development process. In this paper, we have shown that the scope of the classic MDE approach can successfully be extended to include also Business Process

Management. In our approach the business process designs drive the development of enterprise applications.

The challenge here was to connect the BPM world (represented by BiZZdesigner) with the MDE world (represented by OptimalJ). The BPM world has its own concepts and tools aimed at business analysts, which are not compliant with MDA meta-modelling standards. The MDE world, in contrast, mainly focuses on software architects and developers, and uses a corresponding vocabulary (usually UML). We took a pragmatic approach to bridge these two worlds. An XSLT transformation was defined to transform the native BiZZdesigner file format to an XMI-file, which can be read by OptimalJ. Next, we defined QVT-transformations from this model to the default OptimalJ domain model (the PIM). Finally, we used the default transformations in OptimalJ to generate the application model (the PSM) and the corresponding J2EE code to obtain a running enterprise application.

In total, three transformations were written using the QVT core language, each containing approximately eight main rules. Although we spent a lot of time learning QVT and how to operate the tools, it took us in the end about four days to write the actual transformations. The main structure of the transformation specifications can be set up quite quickly; most of the effort is spent to get all the details right.

It appears that the expressiveness of QVT is sufficient to specify the mappings completely. However, to bridge the abstraction gap between business process designs and PIMs, some additional annotations and assumptions (e.g., regarding the semantics of the relations between actions and information items) were needed. Also, it may be necessary to make some slight modifications to the default result of the transformations to fine-tune the behaviour of the generated application.

The approach was successfully applied to a small example of an online order handling process. Future work has to assess to what extent this approach scales to larger applications, although we do not expect major problems here. The support that the used QVT engine offers for the incremental generation of target models will prove to be important when scaling up our approach to realistic, iterative design trajectories.

**Acknowledgements**
The work presented in this paper is part of the Freeband A-MUSE project (http://a-muse.freeband.nl), which is sponsored by the Dutch government under contract BSIK 03025. Special thanks go to Patrick Vermeij and Wim Bast from Compuware for their support on OptimalJ and QVT, and to Frans Faase of BiZZdesign for the realisation of the XMI export of BiZZdesigner models.

## References

Almeida, J.P.A., Iacob, M.-E., Jonkers, H., Lankhorst, M.M. and Leeuwen, D. van. (2006). “An integrated model-driven service engineering environment”, in *Proc. 2nd Interoperability for Enterprise Software and Applications Conferernce* (I-ESA'06), Bordeaux, France, March.

BiZZdesign, http://www.bizzdesign.nl/html/bizzdesigner_en.html

Compuware, *OptimalJ Concepts: OptimalJ 4.1* (2006), Compuware Corporation. Retrieved from: http://frontline.compuware.com/javacentral/members/doc/default.asp.

Dijkman, R. & Joosten, S. (2002). *Deriving Use Case Diagrams From Business Process Models*, CTIT Technical Report 02-08, CTIT, Enschede, The Netherlands.

Eertink, H., Janssen, W.P.M., Oude Luttighuis, P.H.W.M., Teeuw, W.B. & Vissers, C.A. (1999). "A business process design language", in J.M. Wing, J. Woodcock, & J. Davies (eds.), *FM'99 – Formal Methods. World Congress on Formal Methods in the development of computer systems*, Vol. I, LNCS 1708, Springer, pp. 76-95.

Frankel, D.S., (2003). *BPM and MDA: The Rise of Model-Driven Enterprise Systems*, Business Process Trends Whitepaper, June. Retrieved from: www.bptrends.com/publicationfiles/06-03 WP BPM and MDA Whitepaper Frankel11.pdf.

Grangel, R., Ben Salem, R., Bourey, J.-P. & Daclin, N. (2007). "Transforming GRAI Extended Actigrams into UML Activity Diagrams: A first step to model driven interoperability", in R.J. Gonçalves *et al*. (eds.), *Enterprise Interoperability II: New Challenges and Approaches* (Proc. I-ESA'07, Funchal, Portugal), pp. 447-458.

Jonkers, H., *QVT Transformation From BiZZdesigner to OptimalJ Domain Model*, deliverable Freeband/A-MUSE/D2.14b, Telematica Instituut, Enschede, the Netherlands, Feb. 2007.

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

Liew, P., Kontogiannis, K. & Tong, T. (2004). "A framework for business model driven development", *12th International Workshop on Software Technology and Engineering Practice* (STEP'04), pp. 47-56.

Object Management Group. (2005), Meta Object Facility (MOF) 2.0 Query/View/ Transformation Specification, OMG Adopted Specification ptc/05-11-01.

Mukerji, J. and Miller, J. (eds.) (2003). *MDA Guide V1.0.1*, omg/03-06-01, OMG.

OptimalJ, software tool for model driven development for Java. http://www.compuware.com/products/optimalj/

Stolfa, S. & Vondrak, I. (2003). "Using the Business Process for Use Case Model Creation", in M. Benes (Ed.), *Proceedings of the 6th International Conference Information Systems Implementation and Modelling ISIM2003*, Brno.