MODEL TRANSFORMATION FROM BPMN FORMAT TO OPENERP WORKFLOW

Zhe Liu [(a)], Gregory Zacharewicz[(b)]

University of Bordeaux, Lab. IMS, UMR 5218, 33405 Talence, France.

[(a)]liuzhe.ch@gmail.com, [(b)]gregory.zacharewicz@ims-bordeaux.fr

ABSTRACT

This article presents a model transformation from a conceptual modeling language to a platform-specific model. Business people do not have IT background when creating views of their process nevertheless at the end they look at implementing their design. The idea is to support the model transformation from concept to implementation. The source model is BPMN (Business Process Model and Notation) that is a graphical representation used for conceptual modeling in business process management. The target is OpenERP that is an open-source enterprise resource planning (ERP) and customer relationship management (CRM) software which consist in two functional modules, end-user processes and workflow engine. In this research, using model-driven architecture, a mapping relation is built up to bridge the two different models. And with the help of XSLT, the transformation can be achieved, tested and validated.

Keywords: model transformation, BPMN, workflow, OpenERP

1. INTRODUCTION

Business process management and workflow management are both common and important approaches in business process. They help people intuitively understand what to do in business process.

Among the many business process management methods, BPMN is a symbolic language used for specifying business processes using a process models.

In the meantime, OpenERP is the most popular ERP software for European small and medium enterprises. Today it has a number of collaborators and participants all over the world (Pinckaers et al. 2013).

As mentioned above, OpenERP is widely used in European SMEs, which means it has a large amount of users. However, considering about business process modeling, most people are still accustomed to use BPMN. Converting BPMN files to OpenERP workflow will extend a new function in business process management of OpenERP and help people have a better understanding of their business process, have a better communication and cooperation, improve efficiency, and reduce mistakes and omission.

2. BACKGROUND

2.1. Workflow

There are many definitions of workflow given by different persons or organizations. To define standards for the interoperability of workflow management systems, an association, the Workflow Management Coalition (WfMC), founded in 1993.

According to the definition given by WfMC (1996):

Workflow is "*the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant (also called resource, which means a human or machine) to another for action, according to a set of procedural rules.*"

Its synonyms are workflow management, workflow computing and case management.

2.2. Process

As for process, it has different meanings in different domains. To find an accepted definition, we can see the one defined by WfMC (1996):

A process is "*the representation of a business process in a form which supports automated manipulation, such as modelling, or enactment by a workflow management system.*"

And as for business process, defined by WfMC, it is "*a set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships.*"

2.3. BPMN

BPMN is flow-chart based notation for defining business processes. BPMN is an agreement between multiple modeling tools vendors, who had their own notations, to use a single notation for the benefit of end-user understand and training. (Stephen 2004)

BPMN consists of a set of graphical elements. The four basic kinds of elements are flow objects (events, activities and gateways), connecting objects (sequence flow, message flow and association), swimlanes (pool and lane) and artifacts (data object, group and annotation).

2.4. Signavio

With the widespread use of BPMN, many BPMN drawing software and plugins are released. Apart from Visio as we all know, EdrawMax, intalio designer, bonitaBPM, Signavio are all BPM drawing tools.

In this project, we choose Signavio web as the source tool from the BPMN drawing tools mentioned above. The Signavio Process Editor is a network-based business process modeling tool.

User can export drawing in different formats, Signavio archive (SGX), BPMN 2.0 XML, PNG, SVG and PDF in Signavio, which provides many choices. Considering the OpenERP workflow format, the output format of Signavio is more suitable and easy to use.

2.5. OpenERP

OpenERP system provides a flexible modular architecture, including financial management, purchasing/sales management, inventory management, MPR, CRM, human resource management, point-of-sale management, project management, etc. (Xiao 2010).

Among all the functional modules of OpenERP, business process management has two parts, one is end-user processes, and another is workflow engine. In this topic, we also consider these two aspects.

2.5.1. OpenERP workflow

Workflows represent the company's different document flows. They are used to define the behavior of a given file. Developers and system implementers use workflows to determine which object perform the actions, which actions should be performed and at which moments the actions performed.

A workflow is expressed by a directed graph, in which the nodes are called "activities" and the arrowed lines are called "transitions".

Workflow activities represent the nodes of workflows. These nodes are the actions to be executed. They state the work that needs to be performed in the OpenERP server, such as modifying the content of some records, or sending emails.

Workflow transitions are the conditions to be satisfied to go from one activity to the next one; they control how the workflow progresses from activity to activity. Transitions are represented by one-way arrows joining two activities.

In OpenERP workflow format, three main elements exists which are workflow, workflow activity and workflow transition.

2.5.2. OpenERP process

In OpenERP, enterprise process is a view, that is to say, it is a display interface, rather than a real flow. The actual flow is workflow. So, enterprise process is a display view in a different form, like the form view. While workflow is a function that changes the object's state, not a view which is used to display. However, the difference between enterprise process and other views is that each node in enterprise process can be associated with different objects. As a consequence, it can present action menu of multiple objects at the same time in enterprise process, unlike the form view which only can show one object and its related functions (report, wizard, etc.) (Xiao 2010).

Processes form a structure for all activities that enable the company to function effectively. Processes represent workflows across all of a company and the documents associated with the workflows. For this reason, user processes are associated to workflows. Processes are used by end-users to locate an action for more complete handling, and to help them understand the problems which have not been handled in OpenERP.

Corresponding to workflow activity and workflow transition of workflow, process has process node and process transition to present activities and transitions.

3. GENERAL DESIGN

3.1. MDA

The Model-Driven Architecture (MDA) is a framework based on a set of OMG standards that uses models to describe their transformation into other models or complete systems (Mellor et al. 2003).

The OMG's standards include UML (Unified Modeling Language) modeling notation, MOF (Meta Object Facility), XMI (XML Metadata Interchange), CWM (Common Warehouse Metamodel), CORBA (Common Object Request Broker Architecture) middleware, etc. These standards define the key concepts of the MDA.

3.1.1. Models in MDA

According to the different levels of abstraction, MDA identifies three model types, computation independent model (CIM), platform independent model (PIM) and platform specific model (PSM), which are also mentioned above. To make it more clearly, next, each level will be introduced them one by one.

A computation independent model doesn't show any details of the system, since it is defined by the business requirements. CIM is also called "domain model". It focuses on the requirements of the systems.

A platform-independent model is defined by Mellor et al. (2003) as *a model that contains no reference to the platforms on which it depends*. PIM is used to present the aspects of system characters that are unlikely to change with the change of the platform which the model depends on (Robert and Bernhard 2007).

The OMG defines a platform as "*a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns*".

A platform-specific model is defined by Mellor et al. (2003) as *the result of weaving a PIM with the platforms on which it depends*. PSM describes a system in which platform specific details are integrated with the elements in a PIM.

In general, when developing an MDA-based application, the first step is to build a platform independent model (PIM), conveyed by UML on the basis of the proper core model. Platform architects then convert this common application model into one target platform which is a specific platform such as CCM, EJB, or MTS. Standard mappings allow tools to perform automatically some of the transformation.

The work in the following stage is to generate application code. The system needs to produce several types of code and configuration files. (Soley 2000).

The essence of MDA is to distinguish platform-independent model and platform-specific model.

3.1.2. Models in this case

In this topic, we focus on the process from PIM to PSM without considering CIM.

Generally, the model mapping process goes from PIM to PSM, finally to coding.

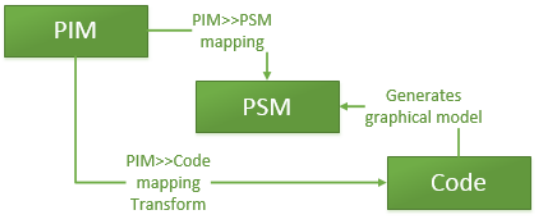

Figure 1 Model mapping in MDA of this project

However, in this transformation, it begins from PIM, after building the mapping relationship between PIM (BPMN in this case) and PSM (OpenERP workflow and business process in this case), it firstly goes to code (workflow and business process XML files) according to the mapping and then generates PSM from XML code with the help of OpenERP platform. See figure 1.

3.2. Model mapping

In the process of the development based on MDA, the transformation between the models is a very important part.

Before transforming PIM, it is necessary to identify the target platform. On the basis of the platform, then designers identify the metamodels and the mapping techniques which will be used with the metamodels (Mellor et al. 2004).

The following part explains the related concepts and design in the mapping process.

3.2.1. Model and metamodel

Defined by Mellor et al. (2003), a model is *a coherent set of formal elements describing something (for example, a system, bank, phone, or train) built for some purpose that is amenable to a particular form of analysis*, such as communication of ideas between people and machines. In other words, a model is an abstraction of the real objects or processes.

Each model, both the source model and the target model, is expressed in a sort of language. The target model's language may define different objects or processes from the source model's language. To describe the models better, we need to define the two languages in one way or another by building a model of the modeling language — a so-called metamodel. A metamodel defines the structure and well-formed rules of the model confirms to it (Tom and Pieter 2006). Which means, as a model is an abstraction of the real world, a metamodel is an abstraction of the model.

Figure 2 Model, metamodel, and platform.

See figure 2. The OMG's Meta Object Facility (MOF) defines a model as an instance of a metamodel. A metamodel can describe a particular platform.

3.2.2. Mapping and model transformation

A model transformation *defines a relation between two sets of models.* (Robert and Bernhard 2007) If one set of models is called as a source model and the remaining one as a target model, then an automated process will take the source models as input and produce the target models as output, according to the transformation rules. We call this process as model transformation (Kleppe et al. 2003; Sendall and Kozaczynski 2003).

To realize the transformation, we build a "bridge" from source models to target models. All model transformations performed are based on to the generic transformation architecture (see figure 3) (Jouault et al. 2008). The "bridge" is a mapping technique (Bazoun et al. 2013). A mapping between models takes one or more models as its input and produces one output model. The mapping technique describes rules for the transformation. These rules are described at the metamodel level.

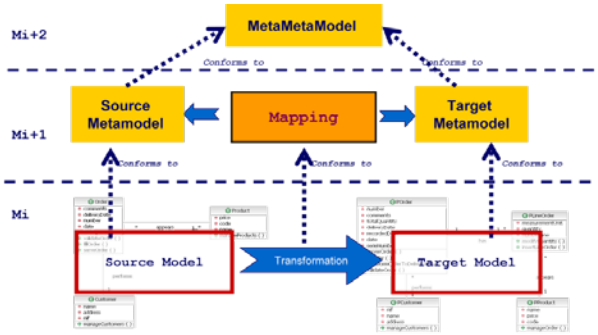

Figure 3 Transformation architecture

In this project, we make a new transformation from BPMN to OpenERP workflow.

In consideration of this case, figure 3 also shows instances of PIM and PSM models and metamodels, and how these instances relate to one another. The source model here refers to the PIM model instance, BPMN. The target models refer to the PSM model instances,

OpenERP workflow and process, which contains the semantic information of the original PIM model instance, as well as the information added as a result of the mapping technique. Because the PSM metamodel describes a platform (here is OpenERP), the PSM also includes the elements that workflow and process depend on in OpenERP platform.

As mentioned above, the model mapping in this case is a little different from the general model mapping in MDA. See figure 4.

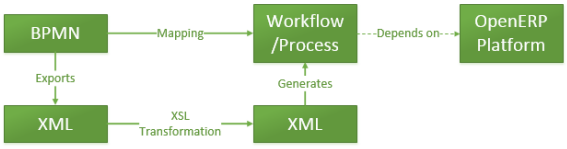

Figure 4 Model transformation in this project

To realize the transformation, first, their mapping relationship needs to be established. In accordance with the mapping, with the help of coding, the original BPMN files can be transformed to the target model files. Then, depending on the OpenERP platform, workflow graphs and process graphs can be generated.

3.2.3. Mapping relation in this case

Core elements of BPMN associated with workflow are flow objects (event, activity and gateway) and sequence flow.

To describe workflow, OpenERP workflow has three elements which are workflow, workflow activity and workflow transition. In general, pool matches workflow. Event and activity (task and sub-process) correspond to workflow activity. Sequence flow and gateway equals to workflow transition.

To describe process, OpenERP process also has three elements which are process, process node and process transition. Process is similar with workflow; the paper doesn't focus too much on describing it.

Even though BPMN and workflow are both used to model the business process, they are still two different modeling language and have totally different file formats. It is not closely coincident between their elements and attributes. The node activities and the transmission method to the next node of the flow that the two modeling methods describe in the business process are not exactly in the same way. Many concepts of them are not overlapping each other.

Figure 5 describes the corresponding relation between BPMN and OpenERP workflow and process in concept, and builds the mapping between source metamodel and target metamodel.

From the workflow perspective, it only has two major elements (namely, workflow activity and workflow transition) to describe business process, there are yet many sub-elements or attributes to make further explanation or give limiting conditions.

For workflow, in more details, "id" or "name" attributes of elements in BPMN format correspond to "id" or "name" attributes of related elements in workflow format. Start event means the value of workflow activity's "flow_start" attribute is "True", and end event means the value of workflow activity's "flow_stop" attribute is "True".

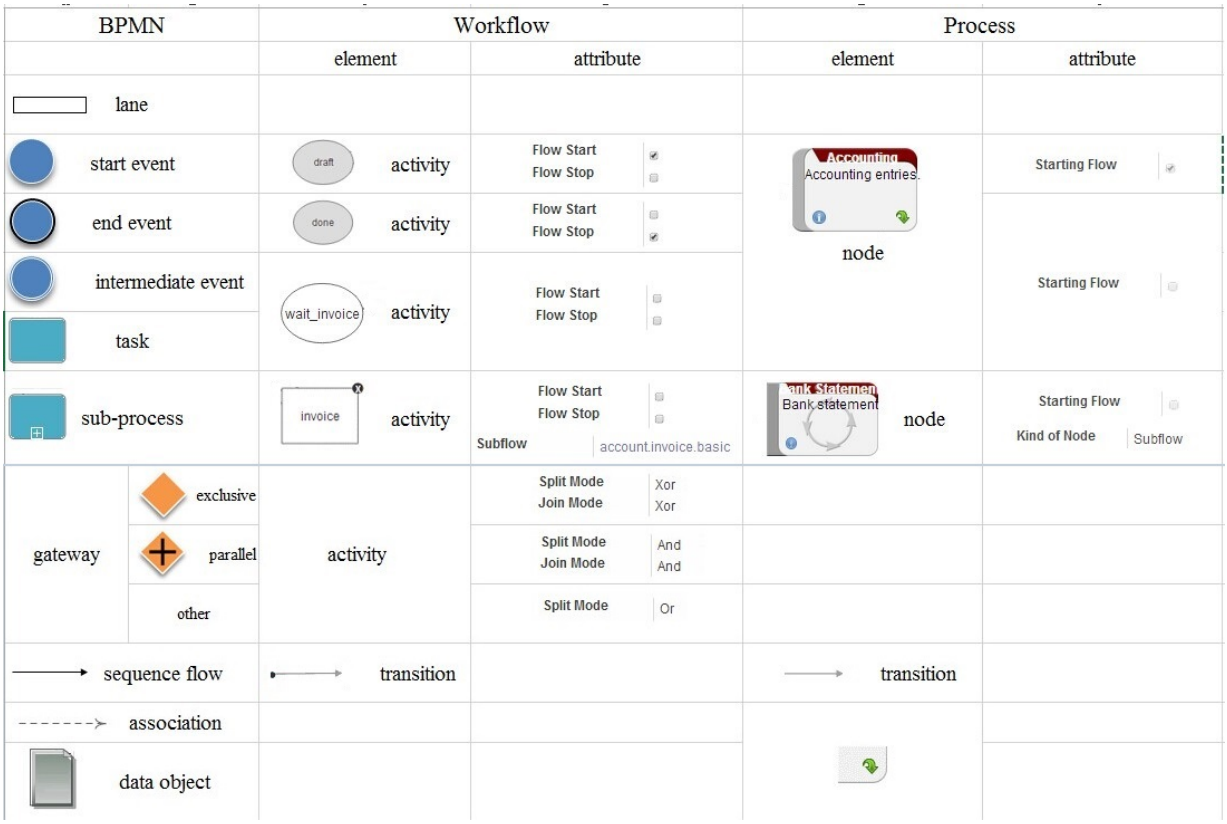

Figure 5 Mapping in concept

Gateway's attribute "gatewayDirection" is also used to describe workflow activity's attributes, which are split_mode (gatewayDirection= "Diverging") and join_mode (gatewayDirection="Converging"). Sequence flow's "sourceRef" attribute corresponds to "act_from" attribute of workflow transition, and sequence flow's "targetRef" attribute corresponds to "act_to" attribute of workflow transition.

For some elements in BPMN, such as data object and text annotation, there is no corresponding content in workflow. And for some other elements in BPMN, we can see some contents which describe the same concepts in workflow. However, the corresponding contents in workflow are not node elements just like in BPMN, but some sub-elements or attributes describe the node elements. For instance, about how to describe the trend of divergence and convergence in the business process flow. In BPMN format, we use "gatewayDirection" attribute of gateway element to describe the trend of divergence and convergence. But in workflow, we use "join_mode" and "split_mode" attributes of workflow activity to indicate whether the activity node is going to diverge or converge the flow. This makes the node describes flow divergence and convergence become a child node of the activity node from a sibling node.

Some of these attributes are hard, even impossible to map. For example, "kind", an attribute of workflow activity which shows the action type the system should execute when flow gets to this node, has four values in total, "dummy", "function", "subflow" and "stopall". Among the four values, "function" is not able to be simply mapped from BPMN. If the value of "kind" equals to "function", it indicates to execute Python code defined in "action", and execute "server action" defined in "action_id". Since the two attributes, "action" and "action_id", are being used only when the value of "kind" equals to "function", they are also difficult to map. The common case is, define a "write" method in "action", and modify the state of the related object. For the "function" type nodes, Python code defined in "action" will return "False" or a client action id.

3.3. XSLT

Since Signavio web has been choosen as the source tool, only the output format of Signavio web is considered as source model.

In OpenERP, the workflow input and the process input are both in a XML format, in this case, we choose BPMN 2.0 XML as Signavio exported format for simplicity.

As the source format and the target format are both XML formats, XSLT is used to achieve the transformation.

XSL (Extensible Stylesheet Language) is a language which is used to present XML data in a readable format. XSLT is a language for transforming XML documents into other forms, such as XML documents in another format, text documents, HTML documents, XHTML, or into XSL Formatting Objects which can then be converted to PDF, PostScript and PNG. The original document is not changed; rather, a new document is created based on the content of an existing one (Clark 1999).

To perform the XSL transformation, the programmer firstly needs to provide one or more XSLT stylesheets, which are also XML documents and describe the transformation rules that the original XML documents should follow, to translate the original XML document into an XML document conforming to the syntax in another format. The XSLT processor then reads in the original XML documents, performs transformations using XSLT stylesheets and produces the output documents needed. See figure 6.

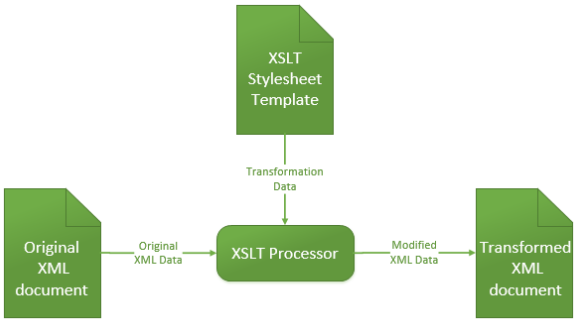

Figure 6 XSLT

As figure 6 illustrates, the original XML data is input into the processor as one input, and an XSLT stylesheet is input as a another input. The output will then be saved directly in the local folder (or a web browser in other situation) as an output XML document. The XSLT stylesheet provides XML formatting instructions, while the XML provides raw data (Burke 2001).

To transform the imported BPMN file to workflow format XML file and process format XML file, XSL code has been designed to process input files.

4. IMPLEMENTATION

In this project, to perform the mapping transformation from BPMN to coding, Java is used. There are four ways to operate XML in Java, DOM (Document Object Model), SAX (Simple API for XML), DOM4J, and JDOM (Java Document Object Model). DOM and SAX are basic means to parse XML, DOM parses XML based on the tree structure of the document, while SAX parses XML based on events stream.

With the help of these methods, there exist different XSLT processors. Among them, Saxon was chosen to be used for Java and .NET.

After developing the program, the transformation can be performed. See the realization process in figure 7

As figure 7 shows, with the business process diagram created in Signavio web, firstly export the diagram in BPMN 2.0 XML format from Signavio, then the original XML document is obtained as one input of the Saxon processor.

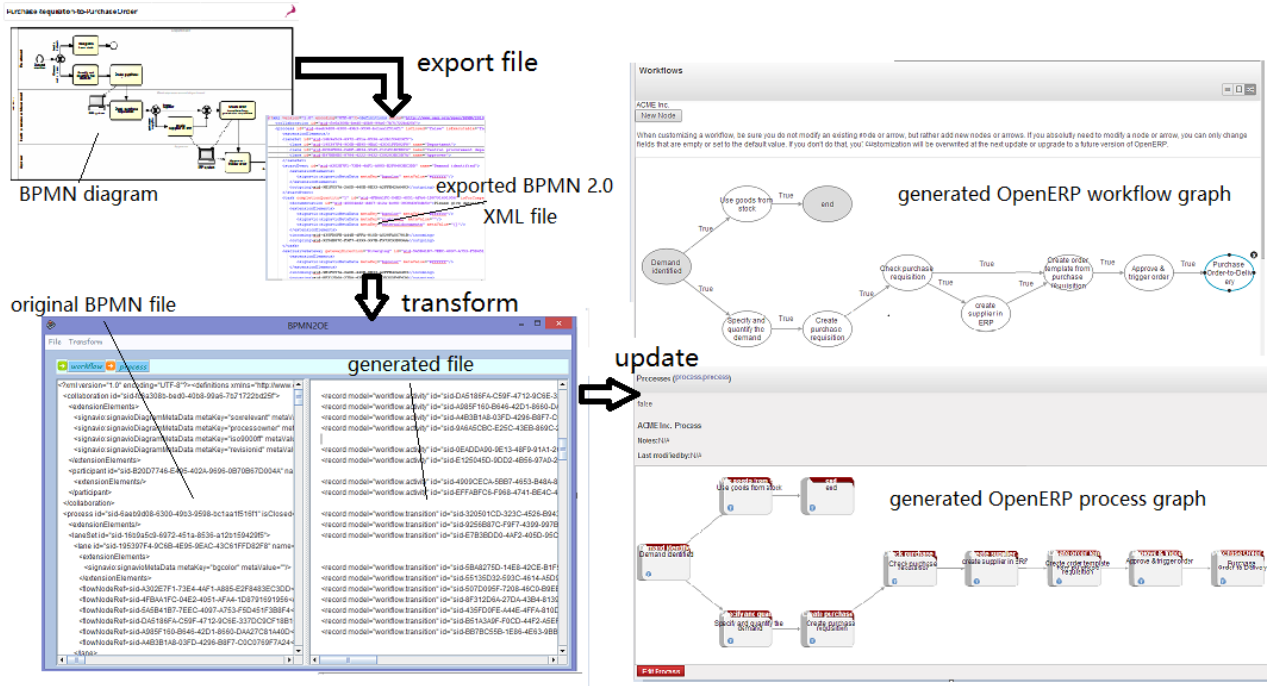

Figure 7 Implementation steps

Open the application and choose the source XML document to be transformed, the source document will be shown in the left pane. The XSLT stylesheets are designed before and provided in the application as the second input of the Saxon processor. Clicking on the button “workflow” or “process”, the application will automatically generate the transformed XML document confirms to OpenERP file format (workflow or process), which will be shown on the right pane. Save the file in the proper folder and update the related module in OpenERP, the workflow or process graphs can be generated easily.

5. CONCLUSION

The paper presented a means of converting PIM BPMN to OpenERP workflow and process at PSM, and it gave an original mapping relation between BPMN and OpenERP workflow and process. This work has permitted to perform the XML files’ transformation with the help of the code, thus it achieves the implemented conversion from BPMN to OpenERP workflow and process.

The next step of the work will consist in the reverse transformation from OpenERP workflow and process to BPMN.

REFERENCES

Bazoun H., Zacharewicz G., Ducq Y., Boye H.: Transformation of extended actigram star to BPMN2.0 and simulation model in the frame of model driven service engineering architecture. SpringSim (TMS-DEVS) 2013: 20

Burke E. M., 2001. Java and XSLT [M]. O’Reilly Media, Inc.

Clark J, 1999. XSL transformations (XSLT) version 1.0. W3C recommendation [J].

France R., Rumpe B., 2007. Model-driven development of complex software: A research roadmap[C]. 2007 Future of Software Engineering. IEEE Computer Society: 37-54.

Jouault F., Allilaire F., Bézivin J., Kurtev I., 2008. ATL: A model transformation tool [J]. Science of computer programming, 72(1): 31-39.

Kleppe A, Warmer J., Bast W., et al., 2003. The model driven architecture: practice and promise [J].

Mellor S. J., Clark A. N., Futagami T., 2003. odel-driven development: guest editors' introduction [J]. IEEE software, 20(5): 14-18.

Mellor S. J., Scott K., Uhl A., Weise D., 2004. Model-driven architecture [J]. Computing Reviews, 45(10): 631.

Mens T., Van Gorp P., 2006. A taxonomy of model transformation [J]. Electronic Notes in Theoretical Computer Science, 152: 125-142.

Pinckaers F., Gardiner G., Van Vossel E., 2013. Open ERP, a modern approach to integrated business management (Release 7.0.0).

Sendall S., Kozaczynski W., 2003. Model transformation: The heart and soul of model-driven software development [J]. IEEE software, 20(5): 42-45.

Soley R., 2000. Model driven architecture [J]. OMG white paper, 308: 308.

WfMC, 1996. Workflow Management Coalition Glossary and Terminology. Available from: http://www.aiai.ed.ac.uk/project/wfmc/ARCHIVE/DOCS/glossary/glossary.html

White S. A. 2004. Business Process Modeling Notation v1.0. For the Business Process Management Initiative (BPMI).

Xiangfu X, 2010. OpenERP applications and development foundation.