SADIO Electronic Journal of Informatics and Operations Research

http://www.dc.uba.ar/sadio/ejs

vol. 9, no. 1, pp. 24-48 (2010)

# An MDA Approach to Business Process Model Transformations

Mauro Argañaraz Ana Funes Aristides Dasso

Universidad Nacional de San Luis,
Ejército de los Andes 950,
5700 San Luis,
Argentina,
e-mail: arganaraz@hotmail.com, {afunes, arisdas}@unsl.edu.ar

**Abstract**

We present in this work an MDA approach for the definition of transformations for business process models. These transformations are based on the use of two platform independent workflow universal languages –UML 2.0 Activity Diagrams and BPMN– and a platform specific language, the XPDL language. The first two languages are used in the definition of a horizontal transformation, while BPMN and XPDL are used in the definition of a vertical transformation. Although there are several options for a model transformation language, we have adhered to one of the principles of MDA, namely the use of standards, therefore adopting the QVT language, which is the transformation language proposed by the OMG. We also show, in this work, a practical case of an application of the transformations proposed here.

**Keywords:** MDA, BPMN, XPDL, UML Activity Diagrams, QVT.

## 1 Introduction

Since some years ago, different kind of IT market products have been supporting features of workflow functionality, however from the early 90s workflow has received increasing recognition being used more and more for business in a number of different organizations. Additionally, with the recent coming of business process management systems, we have seen a radical change in the place and the role played by workflow in an organization.

New design features have been added as workflow technology progressed, but at the same time the main characteristic that distinguishes workflow from other systems, i.e. supporting people in business activities, is still valid. Aiming to achieve a standard and interoperability among different in-development or currently-in-the-market workflow management systems (WfMS), the Workflow Management Coalition (WfMC) published the Workflow Reference Model (WfRM) in 1995.

At present time, there are a number of organizations (W3C, WfMC, OMG, OASIS, AIM, etc.) and standards (UML [Booch et al., 2005], [OMG, 2005]; BPMN [OMG, 2006], [Owen and Raj, 2003]; BPEL [OASIS, 2007]; WSCL [W3C, 2002a]; WSCI [W3C, 2002b]; XPDL [Workflow Management Coalition, 2005]) that have normalized some of workflow features. They have also contributed with different design notations. Two of these standards –the UML Activity Diagrams (UML AD) and the Business Process Modeling Notation (BPMN)– have provided us with easy to read graphical notations for the modeling of workflow process. Some authors state that UML AD and BPMN are similar [White, 2004] and there is an initiative of the OMG [Watson, 2005] (see `http://bmi.omg.org/`) for the integration of UML AD and BPMN under a unified metamodel. On the other hand, the WfMC has proposed XPDL (XML Process Definition Language) for the area of business process definition. XPDL 2.0 specifies a standard file format for the persistence of BPMN diagrams and the interchange of process definitions. This format is based in the WfMC metamodel, which provides a framework for the definition and interchange of process definitions for a number of products, including workflow engines, simulators, Business Process Analysis (BPA) tools, report and activity monitoring tools [Shapiro, 2006].

Some years ago, the lack of a standard notation for business process modeling caused a technical separation between the business process models developed by business analysts and the process representations required by the systems designed to implement and execute those processes. Therefore, it was necessary to manually translate the original business process models to execution models. These translations were subject to misinterpretations and they made difficult the analysis of process evolution and performance by process owners.

While the organizations were discovering the advantages of modeling their processes, during the last years the use of this kind of models went from a luxury article to a daily use artifact. At the same time, many organizations having large numbers of models describing their business could see how these models changed with time (for example, the interoperability B2B arrived with new inventions in communications and new business process) making necessary to keep their models up-to-date and synchronize or translate them to a contemporary modeling language with an executable counterpart. Having these needs in mind, we believe that model transformation techniques can improve the flexibility of business process, reduce the time between process modeling and its transformation to executable code, and decrease the number of people involved in the design and implementation of the organization business processes.

In addition, Model Driven Engineering (MDE) appears as a good way to transfer changes in business processes to those systems implementing such processes. So, using an MDE approach, like Model Driven Architecture (MDA) [OMG, 2003b], the interaction between business people and software engineers can be improved.

Having these considerations in mind, we present in this work a definition of two transformations for business process models making use of MDA with an elaborationist approach. UML 2.0 AD and BPMN, appearing in the role of workflow design universal languages in the definition of an horizontal transformation, allow the definition of business process models, which from a MDA perspective correspond to Platform Independent Models (PIM)[6]. In the role of specific platform language, we have used XPDL to define a vertical transformation from a BPMN PIM to an XPDL PSM (Platform Specific Model). Although a version of BPEL could be a platform specific language suitable for workflow development, given that many tools support transformations from BPMN to BPEL, these tools are not yet flexible enough. In addition, the transformation from BPMN to BPEL defined in the BPMN standard [OMG, 2006] is given in an informal way. There are also some proposals for a direct transformation from UML AD to a BPEL dialect (see [Beck et al., 2005], [Bordbar and Staikopoulos, 2004], [Bézivin et al., 2004], [Gronmo and Jaeger, 2005], [Gardner, 2004]).

[6] Although some authors ([García et al., 2007]; [Rodríguez et al., 2007]) establish that a business model in an MDA context is a Computation Independent Model (CIM); others ([Kalnins and Vitolins, 2006]; [Debnath et al., 2007]) qualify any process model as a PIM. Following the later authors, in this work we consider the particular case of a business process model as a PIM.

However, given that the version 2.0 of XPDL released on October 2005 contains extensions allowing the representation of all BPMN essential features, we decided in this work to make use of XPDL as a platform specific language. Regarding to the transformation language, there are several options such as ATL [INRIA, 2008] or MOLA [MOLA, 2008], both capable of providing an adequate solution for the definition of 1:1 and 1:N mappings. However, we have followed one of the principles of MDA, that is the use of standards, and we have opted for QVT (Query/View/Transformation) (see [OMG, 2007], [Kurtev, 2008]) −the standard model transformation language proposed by the OMG− as a model transformation language.

The rest of this work is organized as follows. Related work is presented in Section 2. In Section 3, we recall some concepts present in model transformations and MDA that are used in this work. In Section 4, we give a perspective of business process modeling in relation to the notations used in our transformations. In Section 5 we describe both transformations from an MDA perspective and we present the QVT transformation rules. In order to validate our proposal, we present in Section 6 an application of our MDA approach to a real practical case. Finally, in Section 7 we discuss some conclusions and future work.

## 2 Related Work

There are several works related to the use of transformations for business process modeling. Gardner et al. (2003) introduce a UML 1.4 profile for automated business processes that allows BPEL processes being modeled using a UML tool as well as permitting these models to be transformed to BPEL from the UML profile so as to automatically generate web services artifacts (BPEL, WSDL, XSD). The output of this process is a BPEL document that can be executed in a BPEL engine. The profile only supports those concepts common to both languages. Mappings are presented informally.

A different approach is taken in [Guelfi et al, 2006]. They make use of a set of formal translation rules for the transformation of UML 1.5 Activity Diagrams to XPDL 1.0 specifications. The transformation is done in two steps; the first is the transformation itself while the second step consists of an optimization algorithm over the XPDL 1.0 specification.

Other works analyze the possibility of transforming directly UML 2.0 AD to BPEL following an MDA approach. For example, Bordbar and Staikopoulos (2004) present the transformation of a UML 2.0 AD to BPEL where web service behavior features are developed using a MOF metamodel for BPEL 1.1 and where OCL is employed as model transformation language. Bézivin et al. (2004) use the ATL language to specify the transformation from a UML 2.0 AD (PIM) to BPEL and JAVA (PSM) languages.

Some other proposals consider the translating of UML 2.0 AD to XPDL. Gallina et al. (2006) give an informal transformation from UML 2.0 AD to XPDL 1.0 centred on the use of transactions and the exceptions mechanisms. Others such as Lohmann et al. (2007) propose a model directed approach to transform a workflow model –developed with UML 2.0 AD– into a BPEL 1.1 or XPDL 2.0 description using a technology based on Triple Graph Grammars. This allows the description of the structural relations between the different elements using declarative and graphic rules that can be applied bidirectionally.

On the other hand several authors have adopted the new notation, BPMN. In [White, 2005] a BPMN–BPEL transformation is described informally. Some tools already implement the BPMN to BPEL transformation although in a superficial manner (see [BPMI, 2008]); moreover the details are proprietary. Filograna et al., (2007), introduce an open source tool based on the web that gives the user the possibility of modelling business processes using BPMN 1.0. Also the tool traduces a BPMN diagram to XPDL 2.0. The mapping description is given informally. Mora et al. (2007) present a translation from BPMN a XPDL 2.0 under a MDA approach using the ATL language.

We can also find in the literature several works on model transformations but they are not associated to business processes. For example in [Debnath et al., 2007], the automation of software development processes specified with SPEM is proposed by a transformation into the WfMC workflow metamodel using the QVT Relations language. García et al. (2007) give a number of rules defined in QVT that implement heuristics for

the derivation of Analysis classes from secure process models. These models are constructed using UML 2.0 AD and BPMN 1.0 which are extended to capture a set of security requisites specified by the business expert.

Rodríguez et al. (2007) define a set of rules using QVT that allow getting use cases from a BPMN business process model that considers security requirements. In a similar work, Rodríguez et al. (2008) propose using an MDA approach to transform BPMN models into UML AD and from them obtain Analysis classes and use cases.

Closer to our proposal, Kalnins et al. (2006) describe a UML 2.0 AD to BPMN 1.0 transformation that supports characteristics of workflow using an ad-hoc UML profile. The transformation is defined using the graphic language MOLA (Model Transformation Language).

## 3 Model Transformations in an MDA context

The concept of model transformation is central to Model Driven Engineering (MDE). A model transformation takes as input a model conforming to a given metamodel and produces as output another model that conforms to a given metamodel. More precisely, following Kleppe et al. (2003) a transformation is the automatic generation of a target model from a given source model according to a transformation definition.

A transformation definition is a set of rules that, all together, describe how a model, expressed in a source language, can be mapped into a model in a target language. A transformation rule is a description of how one or more building blocks of the input language can be mapped into one or more elements of the output language.

Depending on the languages used for the source and target models, we can talk of endogenous or exogenous transformations. When the target and source metamodels are the same, the transformation is called endogenous; if they are different the transformation is called exogenous (also referred as translation).

We can also talk of vertical transformations versus horizontal transformations. A transformation is horizontal when both the target and the source models are specified in the same abstraction level; in a vertical transformation, the models are expressed in different abstraction levels. A taxonomy on model transformations can be found in [Mens et al., 2006].

The OMG consortium has developed the Model Driven Architecture (MDA) proposal as an implementation for MDE. MDA was born with the idea of separating the system specification from its operational logic, namely separating those details defining how the system uses the capabilities from the technologic platform where it is implemented. Therefore, the developer must only be concern with the business logic while the corresponding specific tools are in charge of generating the code for the implementation platforms.

With these ideas in mind, the main goals of MDA are the portability, the interoperability, and the reusability, which are achieved through an architectural separation. The platform independency concept appears frequently in MDA. This is the model property of being independent from the features of any kind of technological platform.

By means of the application of this paradigm, the life cycle of a software system can be completely covered, going from the requirement acquisition to the system maintenance, through the source code generation. In this sense, MDA proposes in the first place the definition of computation independent models (CIM), then the generation of platform independent models (PIM) from the formers, which in turn are transformed into platform specific models (PSM) to finally get the executable code. Each model can be expressed in a different language, and the transformations, CIM-PIM, PIM-PSM, and PSM-code, require transformation tools. These tools receive as input not only a source model but also a transformation definition, which establishes the mapping between the source model language (source metamodel) and the target model language (target metamodel). Note that, in practice, things can be much more complex given that gaps between models can

exist making a straightforward transformation impossible. In such cases, there is the possibility of having several horizontal transformations in a same abstraction layer. For example, a PIM could be transformed several times in PIMs that are more detailed.

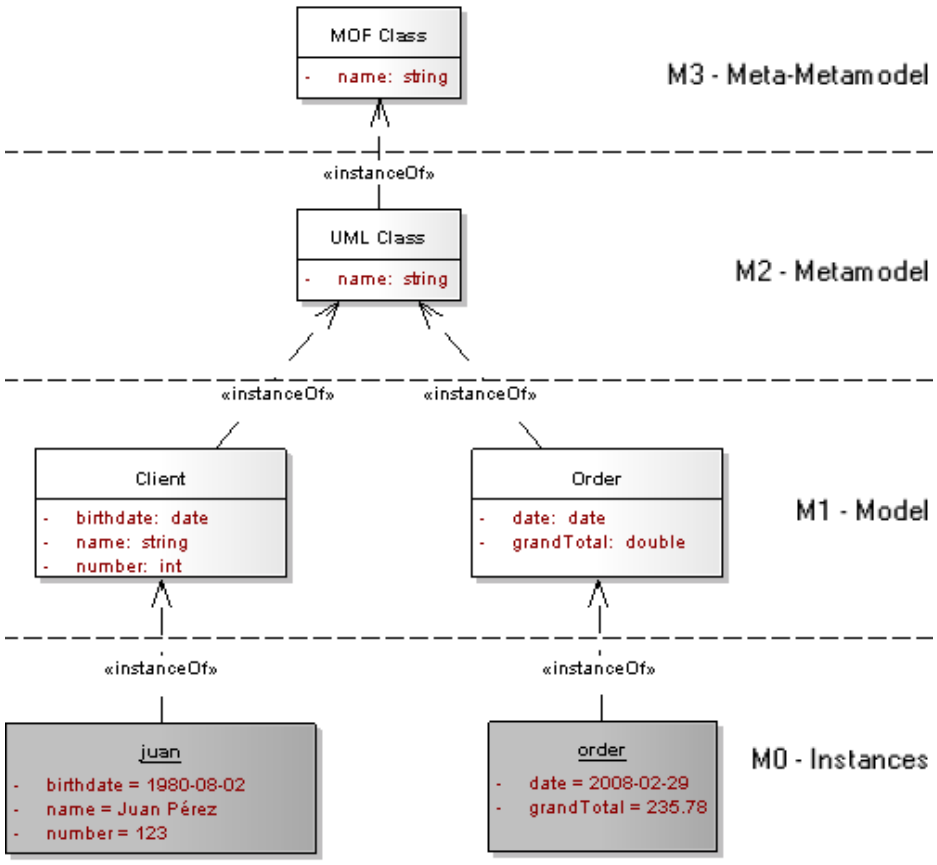


**Figure 1. MOF Metamodels and models.**

Automation and the use of a modeling language together with the adoption of standards form the three pillars on which MDA is based. OMG has defined a set of standards to support its proposal. The standard language proposed by the OMG for the definition of model transformations is QVT, which is based on the OCL (Object Constraint Language) [OMG, 2004]. Using QVT makes possible the definition of generic transformations between metamodels; in this way any instance of a given input metamodel can be transformed into an instance of a given output metamodel. This standard is based on MOF (Meta Object Facility) [OMG, 2003a] and it establishes a language for model transformations (T), a language for model queries (Q) and a language for definitions and generations of views (V) allowing the model analysis from different perspectives.

The use of the MOF standard and the metamodel concept are central in MDA. A metamodel is a model that describes models. Therefore, for example any UML model can be considered an instance of the UML metamodel, which describes all the elements used to create instances of UML models. The definition of metamodels as instances of the MOF meta-metamodel is a crucial point in MDA. The MOF technology describes an architecture based on four levels of abstraction called M0, M1, M2, and M3 as it is shown in Figure 1.

Level M0 is the instance level and it models the real system, where its elements are the concrete instances that form part of the system. An element belonging to this level is for example the client Juan Pérez. Level M1 is the level of the system model. Elements in level M1 are models of concrete systems. In this level, for

example, we can found the definition of the concept "Client". Level M2 is the level of the model of the model (the metamodel). Elements in level M2 are modeling languages. Level M2 defines the elements that can be used to define a model of level M1. Concepts belonging to this level are, for example, the concept of class, attribute, etc. Finally, level M3 corresponds to the model of the metamodel (the meta-metamodel), where the elements used for the definition of diverse modeling languages of level M2 are defined. An element belonging to this level is, for instance, the classifier, while the concept of class in UML defined in level M2 is an instance of the classifier. Fundamentals of MDA and the role of software model transformations for this technology can be found in [Kuznetsov, 2007].

According to McNeile (2003) there are two different interpretations to put MDA into practice. These two different schools of thinking have been named *elaborationist* and *translationist*. In this work, we have adopted the elaborationist approach. Here, models of the application are gradually built by going from a PIM to a PSM and from the later to the code. Once the PIM has been created, a tool can generate a skeleton of the PSM. This skeleton can be "elaborated" by the developer by adding more information or details. In the same way, the developer can also "elaborate" the final code generated by a tool from the PSM. Because of this process, the lower level models can be unsynchronized with those in the upper levels. Due to this problem, modern tools generally have the capability of regenerating high-level models from those of lower level of abstraction. The capability of refining models, modifying and synchronizing with the lower levels is known as Round-trip Engineering.

When adopting this approach, the developer must understand the generated artifacts (PSM and code); otherwise, the modification (elaboration) could not be possible. The elaborationist approach represents the main trend in MDA.

## 4 Business Process Modeling with UML AD and BPMN

Modeling and specification of business process workflow is a research area that has been studied for more than one decade. At present, it is still a research topic in the academic, industrial and commercial fields. Several proposals for the modeling of workflow processes have been developed. Some of them based on Petri nets extensions [Garrido, 2005], some on process algebra [Baeten, 2004] or on UML [Booch et al., 2005], among others. In spite of this large quantity of modeling proposals, there is no standard graphical methodology.

On the one hand, UML AD is an attractive notation for the definition of business process workflows, especially due to the popularity of UML. The usefulness of UML AD for the definition of workflows has been confirmed by several authors ([White, 2004]; [Bordbar and Staikopoulos, 2004]; [Bézivin et al. ,2004]; [Gronmo and Jaeger, 2005]; [Russel et al. 2006]). UML AD is also used for the definition of business processes given that they provide a precise definition of the domain model as well as of software interfaces, including Web services. However, at the same time several drawbacks have been observed from an analysis of workflow patterns ([Russel et al., 2005]; [White, 2004]). In this analysis a list of patterns mainly inspired by case studies were selected in order to determine which ones can be easily supported by a given notation.

On the other hand, a recognized graphical language −that was born for workflow definitions− in the business processes world is BPMN. BPMN benefits users as did UML standardizing modeling activity in Software Engineering. It also provides a certain support for tools (see http://www.bpmn.org/BPMN_Supporters.htm#current) and while UML proposes an object oriented approach for the modeling of applications, BPMN considers a process-oriented approach for the modeling of systems. BPMN is focused on business processes, while UML emphasizes the design of software artifacts. In the context of this work, let us note that according to Eloranta et al. (2006), a transformation from one notation to the other is possible. This is particularly so due to the similarity in the expressiveness for the modeling of workflows of both notations.

Among the five interfaces identified by the WfMC as part of its standardization process, we can find Interface 1. Interface 1 supports the definition of processes and the interchange of models (import and export operations). Part of the standardization process carried out by the WfMC in this area has resulted in the proposal of XPDL (XML Process Definition Language) as the interchange format for business process definitions [Workflow Management Coalition, 2005]. Although BPMN and XPDL are intended for workflow modeling, they do it from different perspectives. While BPMN provides a graphical notation that makes easier the understanding and the communication among several users, XPDL gives us a file in a XML format that can be used for the interchange of business processes between different tools. Consequently, XPDL is the serialization format for BPMN, and it provides us with a file format capable of supporting all the essential features of BPMN including not only the executable properties used in run time but also the graphical description of the diagram. In this way, a tool with the capability of drawing BPMN business processes can save the definition of a process with absolute fidelity and a different tool can read and interpret it exactly as the first tool produced it.

## 5 Business Model Transformations: From UML AD to BPMN and from BPMN to XPDL

A business process model developed by a business analyst is not only useful in the business field itself but it is also helpful for building process representations that are required by systems designed to implement and execute such processes. When model translations are done manually they are subject to errors and misinterpretations that make difficult an evolution and performance analysis of the developed processes by the process owners.

In our proposal, we consider an approach driven by MDA models that apply an elaborationist view in order to transfer the design of a business process from one notation to another without any loss of meaning and by means of a minimum analyst intervention.

In this work, the MDA core process is based on two model transformations that consist in generating new target models from a source model by following a set of rules. We have expressed these rules in the QVT language. In this context, we have defined two main transformations: (a) a horizontal transformation that we have called U2B, which maps a PIM UML AD to a PIM BPMN and (b) a vertical transformation called B2X that produces a PSM XPDL from a PIM BPMN.

Given that our proposal follows an elaborationist approach, the analyst can add more detail to refine the models resulting from any of these transformations. Such transformations can be applied by the developer either individually or one after the other, according to his needs.

On the one hand, the horizontal transformation U2B allows the use of MDA as a bridge between the business processes of diverse organizations using models written in different notations (in this case UML AD and BPMN) at the same time that they keep their models in a higher degree of abstraction. On the other hand, the vertical transformation, B2X, allows an organization to interchange and run models expressed as PSMs XPDL, which can be obtained not only from BPMN PIMs but also from UML AD PIMs, via the corresponding BPMN PIMs.

MDA is based on the MOF technology that determines a four-layer architecture. These four abstraction levels in the MOF architecture are called M0, M1, M2, and M3 as shown in Figure 1. In our approach, as we can see in Figure 2, the UML AD metamodel as well as the BPMN metamodel and the XPDL metamodel are at level M2 and they are instances of the MOF meta-metamodel. The UML AD metamodel describes how to build instances of activity diagrams in UML. These activity diagrams are located at level M1 in the MOF hierarchy. In the same way, the BPMN and XPDL metamodels tell us what are the modeling elements used to build instances of business process models in BPMN and XPDL respectively. These business processes models

correspond to models at level M1 in the pyramid. At level M0, we can find the concrete execution instances of the real world business process modeled with the models at level M1.

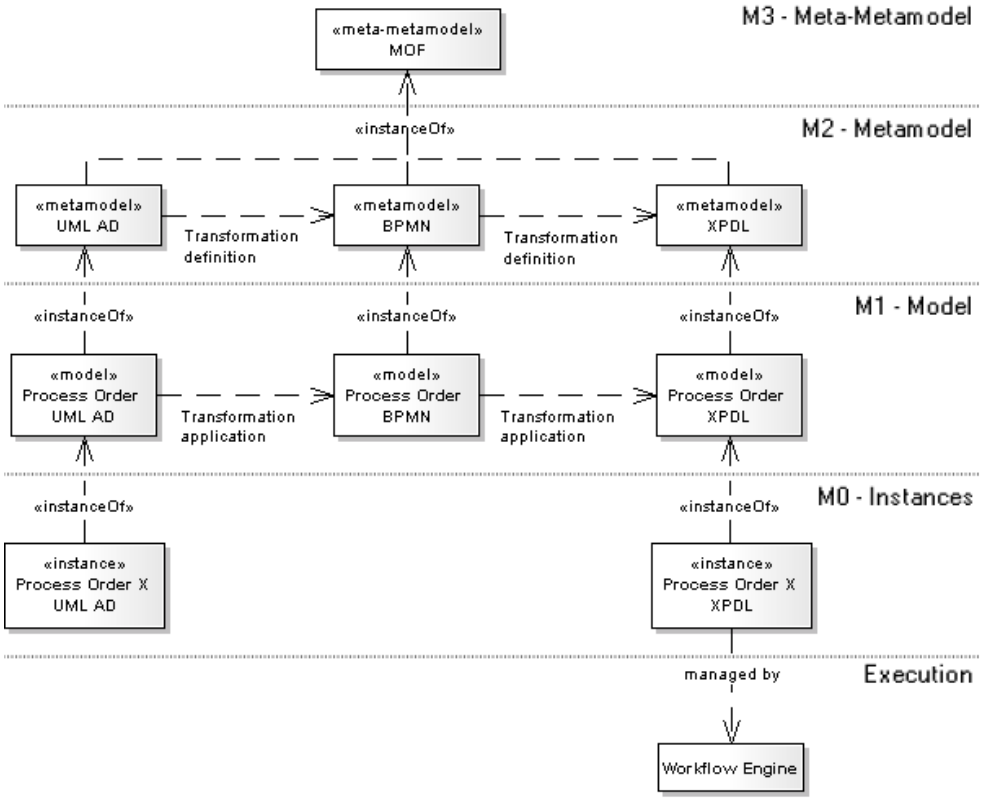


**Figure 2. An MDA view of the proposed transformations.**

In addition, Figure 2 gives us a general view of the business process model transformations described before. At the metamodeling level, besides the three metamodels, we find the definitions of both transformations proposed in this work. In the following level, the one corresponding to the models, we can find the UML AD models that are defined as instances of the upper level UML AD metamodel. Figure 2 shows the UML AD model "Process Order" which is an instance of the UML AD metamodel. This figure also shows how, by applying the U2B transformation on a UML AD model, we get the corresponding BPMN model, which is in turn an instance of the upper level BPMN metamodel. Let us note that a similar reasoning can be applied for the second transformation B2X, i.e. by applying the QVT transformation B2X we get the corresponding model expressed in XPDL. This XPDL model is in turn an instance of the upper level XPDL metamodel. Finally, in the lower level M0, i.e. the instance level, we can see the concrete execution instances of the business process defined by the models of level M1. The concrete case of an XPDL execution instance could be thought as the execution of a business process that is running in a workflow engine.

With respect to the metamodels, we decided to create three partial metamodels for UML AD, BPMN, and XPDL. These metamodels describe only those elements that are considered in our proposal.

In Figure 3, we can see the UML 2.0 AD partial metamodel we worked with. It provides the abstract syntax for those modeling elements in UML 2.0 AD that allow the design of business process models. This metamodel was used together with the BPMN metamodel in the definition of the transformation U2B. Figure 4 and Figure 5 show the BPMN and XPDL metamodels, respectively, both used in the definition of the

transformation B2X. While the UML AD and the XPDL metamodels are compatible subsets of their respective standards, in the case of BPMN since there are no existing standard for it, we built and used the partial ad-hoc metamodel shown in Figure 4.

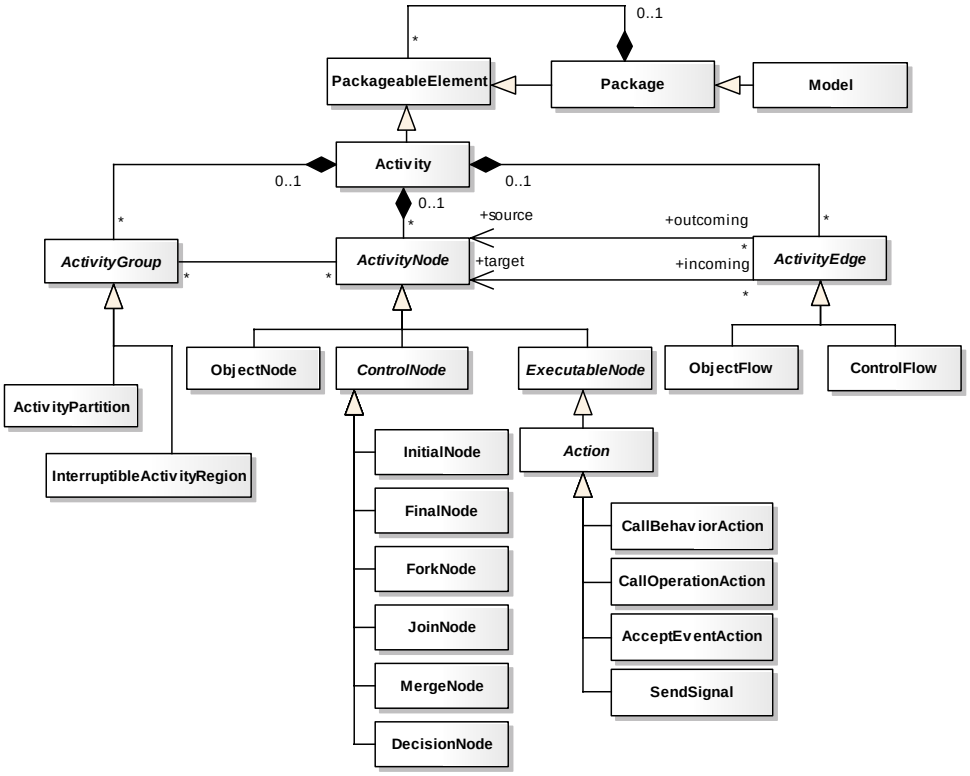


**Figure 3. UML 2.0 AD partial metamodel.**

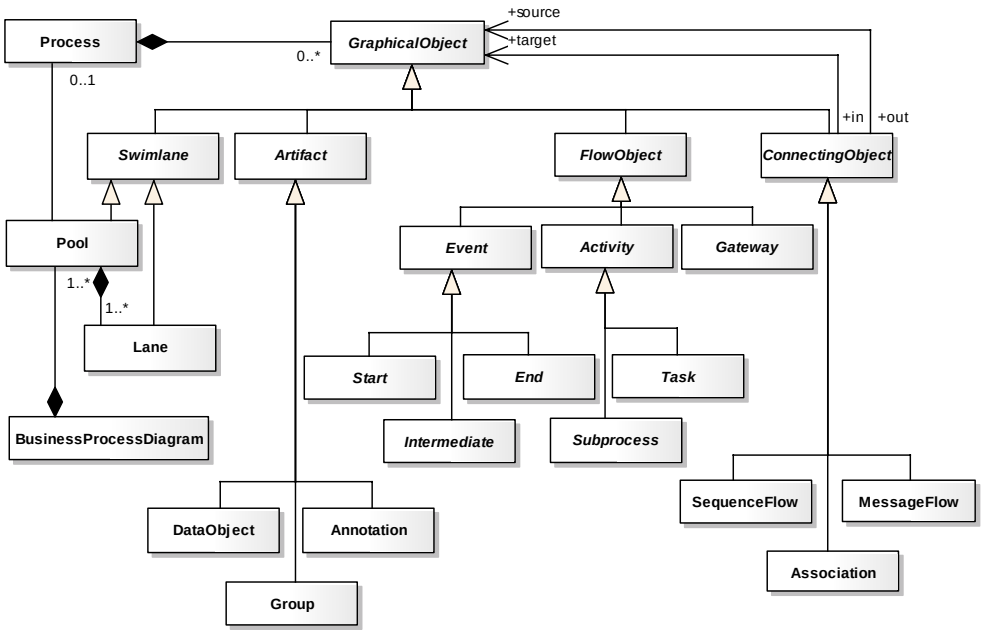


**Figure 4. BPMN partial metamodel.**

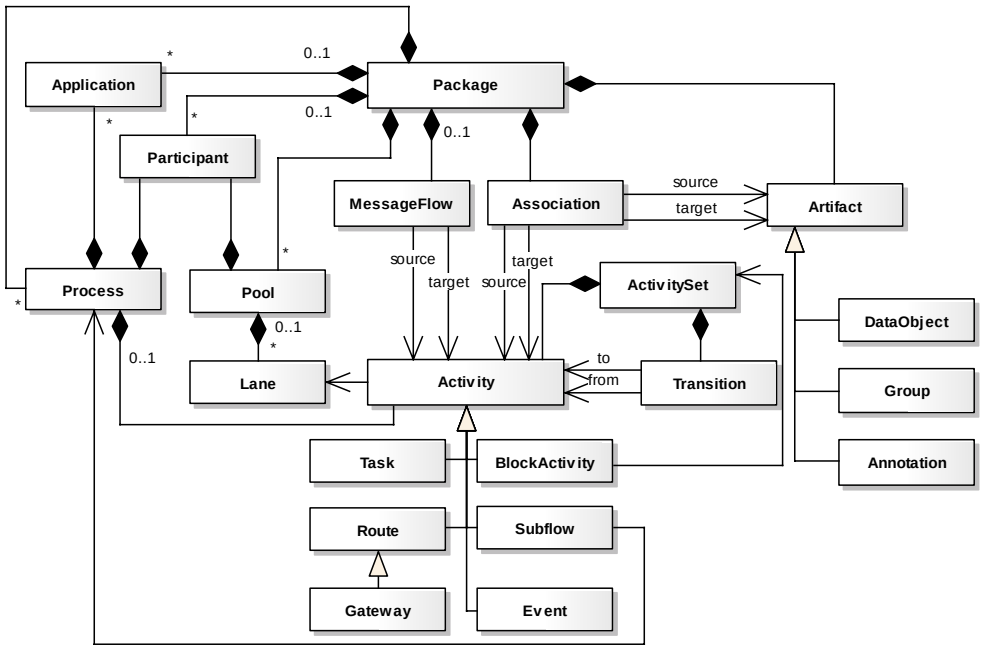


**Figure 5. XPDL partial metamodel.**

Before writing the U2B transformation rules, we established a correspondence between the UML AD metamodel elements and the modeling elements described by the BPMN metamodel. This correspondence was established based on an expressiveness study of the respective notations using workflow patterns ([White,

2004]; [Russel et al., 2005]). To define the B2X transformation, we established a correspondence between the BPMN and XPDL modeling elements by also working with their respective metamodels. XPDL version 2.0 added new capabilities; among them, we can mention the extensions to represent all BPMN features. Because of this, each essential BPMN element has its corresponding equivalent element in the XPDL specification. However, let us note that there are some graphical elements in BPMN that cannot be translated to XPDL; at the same time there are attributes and elements in XPDL that do not have a counterpart in BPMN.

We discovered in both transformations a lack of full correspondences between the metamodels; therefore, we decided to provide the transformation rules with alternative elements from the target metamodel as well as a default transformation rule. In case the analyst does not agree with the predefined option (the default rule), according to the postulates of the elaborationist approach, he can refine the output model.

In the U2B transformation, we defined the mappings between the modeling elements following the groups of concepts established by the WfMC for the workflow technology: processes, roles, tasks, routings, documents and objects, exception handling and compensations. This gave raise to the definition of 69 QVT transformation rules. On the third column of Table 1 we show only the transformation rules for the U2B transformation. The first and second columns refer to modeling elements in the source and target notations, respectively.

**Table 1. U2B transformation rules: from UML AD to BPMN**.

| ***UML AD Element*** | ***BPMN Element*** | ***Transformation Rule*** |
|---|---|---|
| Accept Event Action | Receive Task | `acceptEventAction2receiveTask` |
| Activity Final Node | Terminate End Event | `activityFinalNode2terminateEndEvent` |
| Action | Activity | `action2activity` |
| Activity Edge | Connecting Object | `activityEdge2connectingObject` |
| Activity Group | Embedded Subprocess | `activityGroup2embeddedSubprocess` |
| Activity Group | Swimlane | `activityGroup2swimlane` |
| Activity Node | Artifact | `activityNode2artifact` |
| Activity Node | Flow Object | `activityNode2flowObject` |
| Activity Partition | Lane | `activityPartition2lane` |
| Activity Partition | Pool | `activityPartition2pool` |
| Activity Partition | Process | `activityPartition2process` |
| Call Behavior Action | Independent Subprocess | `callBehaviorAction2indepentSubprocess` |
| Call Operation Action | Service Task | `callOperationAction2serviceTask` |
| Classifier | Element | `classifier2element` |
| Comment | Text Annotation | `comment2textAnnotation` |
| Control Flow | Message Flow | `controlFlow2messageFlow` |
| Control Flow | Sequence Flow | `controlFlow2sequenceFlow` |
| Decision Node | Data Based Exclusive Gateway | `decisionNode2dataBasedExclusiveGateway` |
| Expansion Region | Embedded Subprocess | `expansionRegion2embeddedSubprocess` |
| Flow Final Node | None End Event | `flowFinalNode2noneEndEvent` |
| Fork Node | Inclusive Gateway | `forkNode2inclusiveGateway` |
| Fork Node | Parallel Gateway | `forkNode2parallelGateway` |
| Initial Node | None Start Event | `initialNode2noneStartEvent` |
| Interruptible Activity Region | Embedded Subprocess | `interruptibleActivityRegion2embeddedSubprocess` |
| Join Node | Inclusive Gateway | `joinNode2inclusiveGateway` |
| Join Node | Parallel Gateway | `joinNode2parallelGateway` |
| Merge Node | Data Based Exclusive Gateway | `mergeNode2dataBasedExclusiveGateway` |
| Object Flow | Association | `objectFlow2association` |
| Object Node | Data Object | `objectNode2dataObject` |
| Opaque Action | Script Task | `opaqueAction2scriptTask` |
| Send Signal Action | Send Task | `sendSignalAction2sendTask` |
| UML Activity Diagram | Business Process Diagram | `ad2bpd` |

In the case of the B2X transformation, we create other 37 rules, which are shown in Table 2. Like in the U2B transformation, to give the correspondences between the elements of both notations, we follow the groups of

concepts established by the WfMC for the workflow technology. Table 2 shows the modeling elements identified en each notation as well as the names of the respective QVT transformation rules.

**Table 2. B2X transformation rules: from BPMN to XPDL.**

| *BPMN Element* | *XPDL Element* | *Transformation Rule* |
|---|---|---|
| Activity | Activity | `activity2activity` |
| Activity | Block Activity | `activity2blockActivity` |
| Artifact | Artifact | `artifact2artifact` |
| Association | Association | `association2association` |
| Business Process Diagram | Package | `bpd2package` |
| Compensation Intermediate Event | Result Compensation | `compensationIntermediateEvent2resultCompensation` |
| Data Object | Data Object | `dataObject2dataObject` |
| Element | Element | `element2element` |
| Embedded Subprocess | Activity Set | `embeddedSubprocess2activitySet` |
| Error End Event | Result Error | `errorEndEvent2resultError` |
| Error Intermediate Event | Result Error | `errorIntermediateEvent2resultError` |
| Event | Activity | `event2eventActivity` |
| Gateway | Activity | `gateway2routeActivity` |
| Independent Subprocess | Sub Flow | `independentSubprocess2subFlow` |
| Lane | Lane | `lane2lane` |
| Manual Task | Task Manual | `manualTask2taskManual` |
| Message | Message Type | `message2messageType` |
| Message End Event | | `messageEndEvent2triggerResultMessage` |
| Message Flow | Message Flow | `messageFlow2messageFlow` |
| Message Intermediate Event | Trigger Result Message | `messageIntermediateEvent2triggerResultMessage` |
| Message Start Event | Trigger Result Message | `messageStartEvent2triggerResultMessage` |
| Multi Instance Loop | Loop Muli Instance | `multiInstanceLoop2loopMultiInstance` |
| Participant | Participant | `participant2participant` |
| Pool | Pool | `pool2pool` |
| Process | Process | `process2process` |
| Receive Task | Task Receive | `receiveTask2taskReceive` |
| Rule Intermediate Event | Trigger Rule | `ruleIntermediateEvent2triggerRule` |
| Rule Start Event | Trigger Rule | `ruleStartEvent2triggerRule` |
| Script Task | Task Script | `scriptTask2taskScript` |
| Send Task | Task Send | `sendTask2taskSend` |
| Sequence Flow | Transition | `sequenceFlow2transition` |
| Service Task | Task Service | `serviceTask2taskService` |
| Standard Loop | Loop Standard | `standardLoop2loopStandard` |
| Timer Intermediate Event | Trigger Timer | `timerIntermediateEvent2triggerTimer` |
| Timer Start Event | Trigger Timer | `timerStartEvent2triggerTimer` |
| Transaction | Transaction | `transaction2transaction` |
| User Task | Task User | `userTask2taskUser` |

Next, and to illustrate the rules, we give the QVT code for the rule `umlad2bpmn` that captures the U2B transformation. This transformation takes a UML AD model as input and returns a BPMN output model. In the operation `main`, which is the entry point to the transformation, the elements `UMLActivityDiagram` are selected and the map operation `ad2bpd` is applied to them.

```
transformation umlad2bpmn(in src:umlad, out tar:bpmn);
main() {
src.objects()[UMLActivityDiagram]->map ad2bpd();
}
```

Each instance of `UMLActivityDiagram` is translated to an instance of `BusinessProcessDiagram` in the operation `ad2bpd`. In order to carry out this, `ad2bpd` takes all the partitions (`ActivityPartition`) that compose the activities of a UML AD model and applies the operation `activityPartition2pool` on them.

```
mapping UMLActivityDiagram::ad2bpd():BusinessProcessDiagram {
      result.pool:=self.activity->group[ActivityPartition]->
                        map activityPartition2pool();
}
```

The mapping `activityPartition2pool` translates only the external partitions or those partitions that are not subordinated to an element `Pool`. We can also observe how the lanes associated to a pool and one instance of `Process` are created by means of the mappings `activityPartition2lane` and `activityPartition2process`, respectively.

```
mapping ActivityPartition::activityPartition2pool():Pool
when { self.isTopParent(); }
{
      result.lane:=self.activity.group[ActivityPartition]->
            select(g|g.superGroup==self)->map activityPartition2lane();
      result.process:=self.map activityPartition2process();
}
```

## 6 Applying the Transformations to a Business Process Model

In this section, we present an application of the proposed transformations to a business process of books on consignment in the context of a company that sells school and literature books. The UML AD diagram shown in Figure 6 models this process. The process determines a workflow that starts when a Publisher sends a package of books on consignment to the bookshop and finishes with the corresponding payment, including the accounting process.

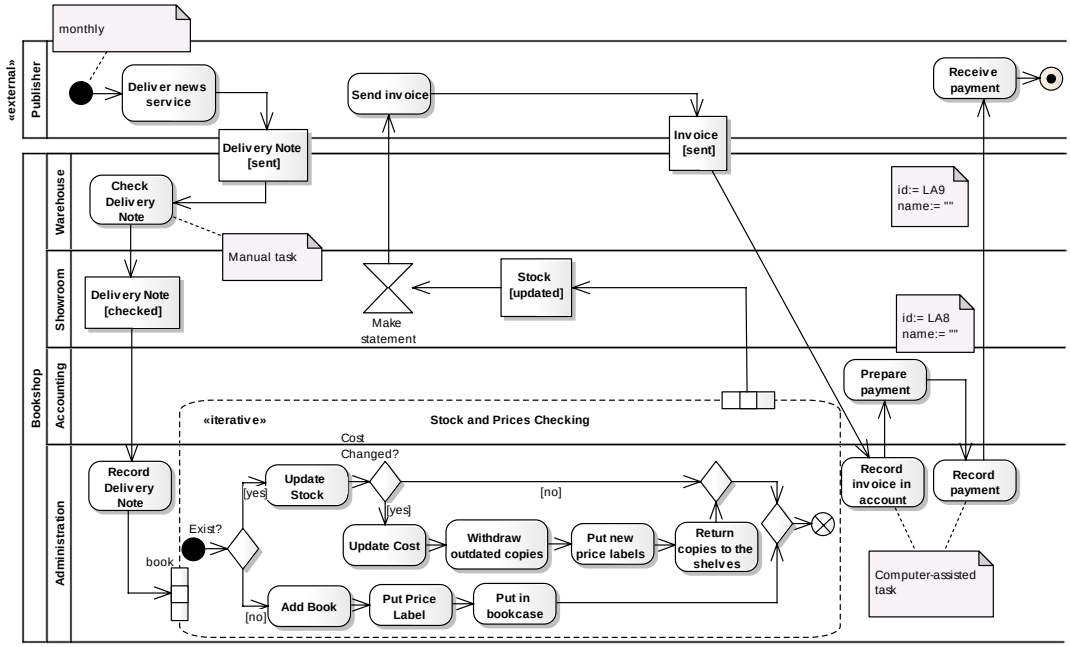


**Figure 6.** UML AD model corresponding to the process "Book Consignment".

The workflow starts with the Publisher who provides a delivery service of books on consignment known in the circle as "News Service". This service consists in sending to the clients a package of publications on consignment. The publications can be new titles or existing copies sent for completing the stock. The Publisher sends, together with the book package, a delivery note that documents the transaction.

When the package arrives to the bookshop warehouse, the person in charge of the warehouse controls each item; then signs and stamps the document and he personally takes it to the Administration department. This delivery note is recorded in the system by the Administration department. When the delivery note is recorded, at the same time the stock and price of each item are updated. In case that the price of an item changes, an additional step takes place −all the copies of this book are taken out from the shelves of the showroom and are relabeled with the new price. At the end, all the copies are returned to the shelves.

According with the schedule agreed with the Publisher, the showroom manager prepares a statement of the books sold and sends it to the Publisher together with the books that have not been sold for its control and subsequent invoicing.

The Publisher receives the statement and checks it against the copies of the delivery notes that it has sent to the bookshop previously, during the corresponding period, and its records of pendings. Then, it produces the invoice and sends it to the client for its subsequent payment. The invoice arrives to the Administration department where it is recorded in the Publisher account. The Accounting department prepares a check for the payment and then it is passed to the Administration department to be recorded and paid, ending in this way the consignment process.

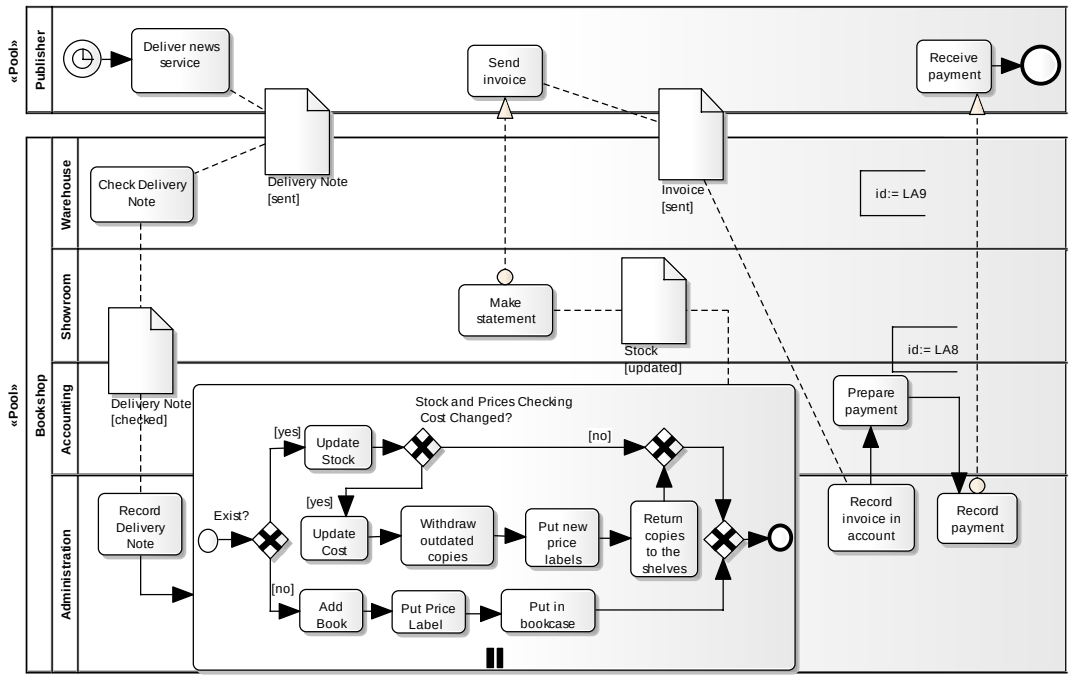


**Figure 7.** BPMN model corresponding to the business process "Book Consignment".

When applying the U2B transformation to the UML AD input model given in Figure 6, the corresponding transformation rules are executed in the order shown by the sequence in Table 3. We obtain as a result a BPMN output model similar to the one shown in Figure 7. Once the automatic transformation process has taken place, the analyst might want to elaborate the BPMN model in order to refine it according to the considerations pointed out in the comments of the input model. More concretely, in this example we have replaced the task `bpmn::ScriptTask` "Check delivery note" by a task `bpmn::ManualTask`. We did

the same with the tasks "Record invoice in account" and "Record payment" which were replaced by tasks of kind `bpmn::UserTask`. Finally, we changed the start event (`bpmn::NoneStartEvent`) by an element `bpmn::TimerStartEvent`.

**Table 3.** Execution sequence of the U2B transformation rules for the business process "Book Consignment".

| ***UMLAD Element*** | ***BPMN Element*** | ***Transformation Rule*** |
|---|---|---|
| UML Activity Diagram | Business Process Diagram | `ad2bpd` |
| Activity Partition | Pool | `activityPartition2pool` |
| Activity Partition | Lane | `activityPartition2lane` |
| Activity Partition | Process | `activityPartition2process` |
| Initial Node | None Start Event | `initialNode2noneStartEvent` |
| Decision Node | Data Based Exclusive Gateway | `decisionNode2dataBasedExclusiveGateway` |
| Merge Node | Data Based Exclusive Gateway | `mergeNode2dataBasedExclusiveGateway` |
| Flow Final Node | None End Event | `flowFinalNode2noneEndEvent` |
| Activity Final Node | Terminate End Event | `activityFinalNode2terminateEndEvent` |
| Accept Event Action | Receive Task | `acceptEventAction2receiveTask` |
| Opaque Action | Script Task | `opaqueAction2scriptTask` |
| Object Node | Data Object | `objectNode2dataObject` |
| Expansion Region | Embedded Subprocess | `expansionRegion2embeddedSubprocess` |
| Object Flow | Association | `objectFlow2association` |
| Control Flow | Sequence Flow | `controlFlow2sequenceFlow` |
| Control Flow | Message Flow | `controlFlow2messageFlow` |

The application of the B2X transformation to the refined BPMN model gave as a result the XPDL specification shown in the appendix. In this case, the sequence of transformation rules given in Table 4 was executed.

**Table 4.** Execution sequence of the B2X transformation rules for the business process "Book Consignment".

| ***BPMN Element*** | ***XPDL Element*** | ***Transformation Rule*** |
|---|---|---|
| Business Process Diagram | Package | `bpd2package` |
| Participant | Participant | `participant2participant` |
| Process | Process | `process2process` |
| Embedded Subprocess | Activity Set | `embeddedSubprocess2activitySet` |
| Activity | Activity | `activity2activity` |
| Manual Task | Task Manual | `manualTask2taskManual` |
| Script Task | Task Script | `scriptTask2taskScript` |
| User Task | Task User | `userTask2taskUser` |
| Receive Task | Task Receive | `receiveTask2taskReceive` |
| Multi Instance Loop | Loop Muli Instance | `multiInstanceLoop2loopMultiInstance` |
| Activity | Block Activity | `activity2blockActivity` |
| Gateway | Activity | `gateway2routeActivity` |
| Event | Activity | `event2eventActivity` |
| Timer Start Event | Trigger Timer | `timerStartEvent2triggerTimer` |
| Sequence Flow | Transition | `sequenceFlow2transition` |
| Pool | Pool | `pool2pool` |
| Lane | Lane | `lane2lane` |
| Artifact | Artifact | `artifact2artifact` |
| Data Object | Data Object | `dataObject2dataObject` |
| Association | Association | `association2association` |
| Message Flow | Message Flow | `messageFlow2messageFlow` |

In Figure 8, we can observe a partial representation of the MOF pyramid showing the U2B and B2X transformations at levels M1 and M2.

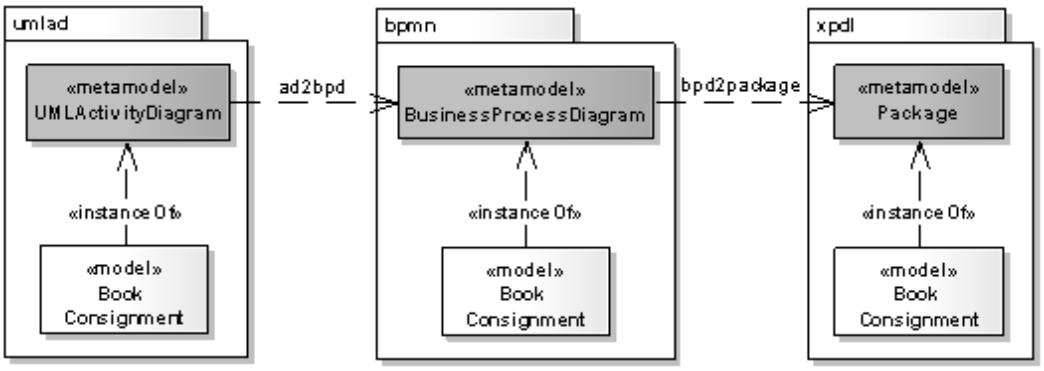


**Figure 8.** Level M1 (model) and M2 (metamodel) of the MOF hierarchy for ours transformations.

Below we describe how each modeling element in the UML AD model is translated to the corresponding XPDL modeling element via an intermediate BPMN model, by following the concepts established by the WfMC for the workflow technology: processes, roles, transitions, tasks, routings and documents.

**Processes**. The transformation creates an element `bpmn::Process` for each partition (`umlad::ActivityPartition`) that is in correspondence with an element `bpmn::Pool`. Therefore, the external partition "Publisher" produces an abstract process, while the partition "Bookshop" produces an internal process. In XPDL, we obtain two elements `xpdl::Process` with the previously mentioned characteristics.

**Roles**. The external partition "Publisher" and the partition "Bookshop" are translated in elements `bpmn::Pool`. At the same time, the subordinated partitions "Administration", "Accounting", "Showroom", "Warehouse" generate instances of `bpmn::Lane`, which are associated to the pool "Bookshop". Next, we obtain the elements `xpdl::Pool` and `xpdl::Lane` based on the existing one-to-one relation with the elements `bpmn::Pool` and `bpmn::Lane`. In Figure 9, we show a graphical representation of the translation of roles according to the MOF standard.

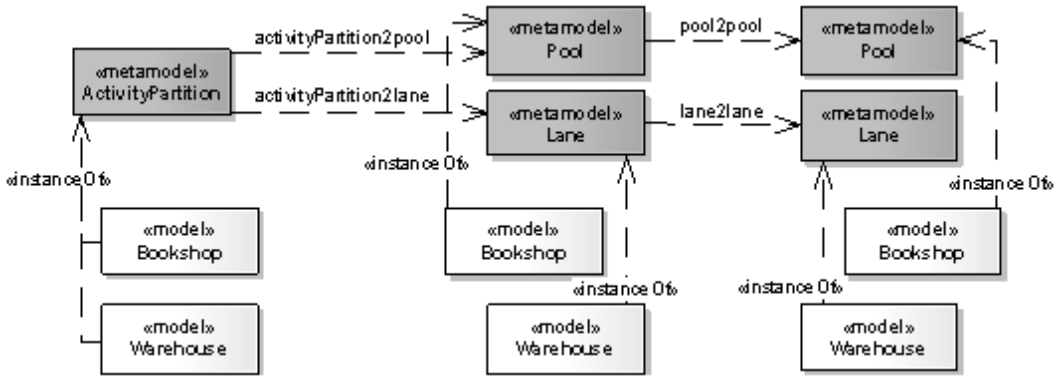


**Figure 9.** Levels M1 (model) and M2 (metamodel) of the MOF hierarchy for the translation of roles.

**Transitions**. Each element `umlad::ControlFlow` is translated in one instance of `bpmn::SequenceFlow` if it connects two nodes in the same partition of upper level. By the contrary, an instance of `umlad::ControlFlow` is translated into one element `bpmn::MessageFlow` if the connected nodes are in different partitions of the upper level. One instance of `umlad::ObjectFlow` is transformed in one element `bpmn::Association`. In XPDL, we can find the same connection objects that

in BPMN, therefore the respective instances of `xpdl::Transition`, `xpdl::MessageFlow` and `xpdl::Association` are created. We can see, in Figure 10, a graphical representation according to the MOF standard of the correspondences for transitions.

**Tasks**. The subtypes of actions `umlad::AcceptEventAction` and `umlad::OpaqueAction` are translated in tasks `ReceiveTask` and `ScriptTask`, respectively. After the application of the previous transformation, we made some adjustments to replace the tasks `bpmn::ScriptTask` "Check delivery note", "Record invoice in account", and "Record payment" by instances of `bpmn::ManualTask`, `bpmn::UserTask` and `bpmn::UserTask`, respectively. XPDL has a direct correspondence with the BPMN tasks; therefore, we obtain automatically the respective elements `xpdl::TaskReceive`, `xpdl::TaskScript`, `xpdl::TaskUser` and `xpdl::TaskManual`.

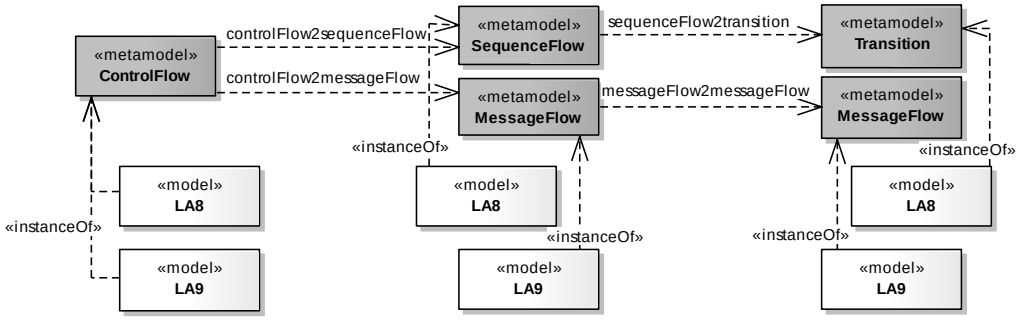


**Figure 10.** Levels M1 (model) and M2 (metamodel) of the MOF hierarchy for the translation of transitions.

**Rountings**

*Decisions*. The decision nodes (`umlad::DesicionNode`) and the fusion nodes (`umlad::MergeNode`) that appears in an expansion region represent exclusive decisions and fusions, respectively. The translation to BPMN generates XOR gateways based in data (`bpmn::DataBasedExclusiveGateway`). The transformation to XPDL produces elements of kind `xpdl::Route`. In Figure 11, we can observe these correspondences in a graphical fashion according to the levels M1 and M2 of the MOF standard.

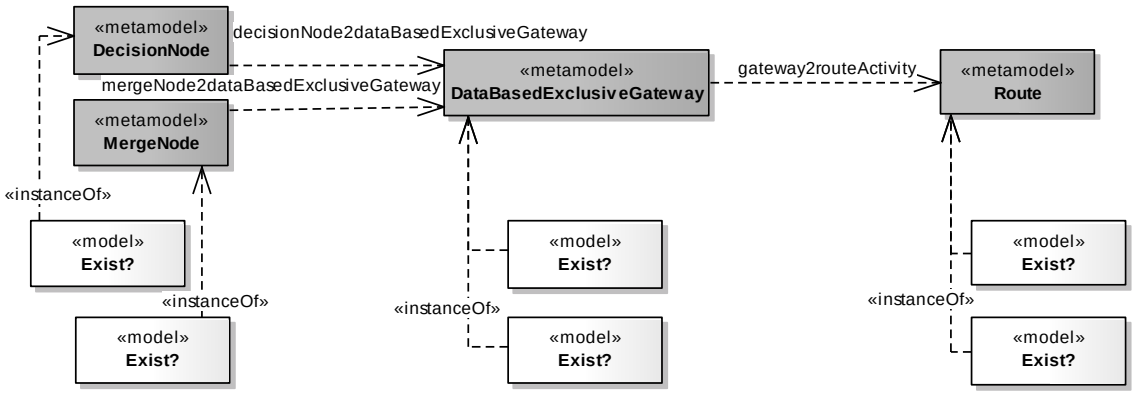


**Figure 11.** Levels M1 (model) and M2 (meta model) of the MOF scheme for the translation of decisions.

*Initial and final nodes*. The initial nodes (`umlad::InitialNode`) produce by default an event `bpmn::NoneStartEvent`. However, as the annotation attached to the node that starts the activity in the UML AD model establishes that the task must be done monthly, we may replace, by following the

elaborationist postulate, the predefined option by one event `bpmn::TimerStartEvent`. The transformation to XPDL of the element `bpmn::TimerStartEvent` produces one element `xpdl::StartEvent`, which references to one object `xpdl::TriggerTimer`; while the element `bpmn::NoneStartEvent` generates only one element `xpdl::StartEvent`.

The flow final node (`umlad::FlowFinalNode`) of an expansion region is translated into one element `bpmn::NoneEndEvent`. The activity final node (`umlad::ActivityFinalNode`) is translated into one instance of `bpmn::TerminateEndEvent`. In XPDL 2.0, both elements produce instances of `xpdl::EndEvent`.

*Loop or cycle*. The expansion region (`umlad::ExpansionRegion`) "Stock and price checking" is translated into one embedded sub-process (`bpmn::EmbeddedSubprocess`) pointing to a loop `bpmn::MultiInstanceLoop`. The items in the delivery note determine the number of times the sub-process must be repeat. In XPDL, this sub-process is transformed in one element `xpdl::ActivitySet` and one activity `xpdl::BlockActivity` is associated to it, which acts as a trigger. Therefore, the activity `xpdl::BlockActivity` has a reference to a loop `xpdl::LoopMultiInstance`.

**Documents**. The elements `umlad::ObjectNode` "Delivery Note" and "Invoice" generate data objects (`bpmn::DataObject`). In XPDL, two related objects are created, one element `xpdl::Artifact` that points to one element `xpdl::DataObject`. In Figure 12, we represent these transformation rules in the levels M1 and M2 of the MOF hierarchy.

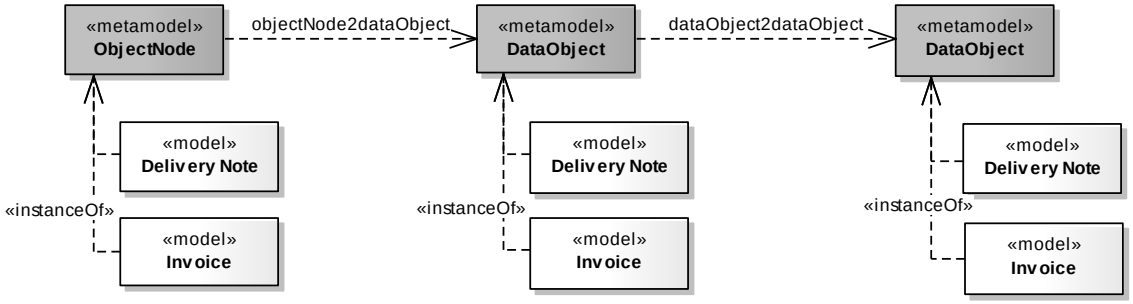


**Figure 12.** Levels M1 (model) and M2 (metamodel) of the MOF hierarchy for the translation of documents.

## 7 Conclusions

In this work, we have brought the MDA approach to the area of workflow definition. This area corresponds to Interface 1 named Process Definition in the Reference Model of the WfMC. It allows the modeling and documentation of a workflow, and it defines a separation between the development and the execution environments. As a result of the modeling and design processes, we obtain a process definition, which can be used as input to different workflow products, especially to workflow engines. By applying this approach, we have shown how the business processes, modeled using UML AD or BPMN, can be transformed to an XPDL specification in a semi-automatic fashion, with no loss of meaning and with a minimum intervention of the analyst.

As part of this work, we have made an analysis of the semantics of UML AD, BPMN, and XPDL languages. At the same time, we have defined the semantic equivalencies between UML AD and BPMN, and between BPMN and XPDL, by means of informal descriptions as well as formally through the respective transformation rules given in QVT language. The transformations were defined using the SmartQVT tool [France Telecom R & D, 2008], an open source implementation for Eclipse, that supports the QVT language. Finally, to validate our proposal, we have presented an application of the transformations to a business process of goods on consignment.

Let us note that our approach is not limited to this specific option of output notation. A reverse transformation or the use of a different output notation could be treated in a similar way.

An important work for the future is the definition of transformation rules allowing the translation not only of UML AD and BPMN but also of other business processes modeling languages to a metamodel independent of the notation used. This should work as an intermediary between two different notations in the PIM perspective of our proposal.

## References

Baeten, J.C.M., A Brief History of Process Algebra. Rapport CSR 04-02, TU Eindhoven, 2004. http://www.win.tue.nl/fm/0402history.pdf, (2004)

Beck, K., Joseph, J., Goldszmidt, G., Learn business process modeling basics for the analyst. IBM Developerworks. http://www-128.ibm.com/developerworks/library/ws-bpm4analyst/ (2005).

Bézivin, J.; Hammoudi, S.; Lopes, D.; Jouault, F., Applying MDA Approach to B2B Applications: A Road Map. Workshop on Model Driven Development (WMDD 2004) at ECOOP 2004, Springer-Verlag, LNCS, vol. 3344, June 2004. (2004)

Bordbar, B., Staikopoulos, A., On behavioural model transformation in Web services. 5th International Workshop on Conceptual Modeling Approaches for e-Business eCOMO 2004, November 8-12, 2004. (2004)

Booch, G., Rumbaugh, J., Jacobson, I., The Unified Modeling Language User Guide. Second edn. Addison Wesley Professional (2005)

BPMI, Current Implementations of BPMN. http://www.bpmn.org/BPMN_Supporters.htm#current, last access: 10/01/2008 (2008)

Debnath, N., Zorzan, F.A., Montejano, G., Riesco, D., Transformation of BPMN Subprocesses Based in SPEM Using QVT. Proceedings of IEEE EIT 2007, pp. 146–151 (2007)

Eloranta, L., Kallio, E., Terho, I., A Notation Evaluation of BPMN and UML Activity Diagrams. http://www.soberit.hut.fi/T-86/T-86.5161/2006/BPMN vs UML final.pdf (2006)

Filograna, A.; Giunta, G.; Ingraffia, N.; Loiacono, L., An integrated approach for modelling Business Processes using BPMN and XPDL standards.
http://semantics.eng.it/bxmodeller/docs/An%20integrated%20approach%20for%20modelling%20business%20processes%20using%20BPMN%20and%20XPDL%20standards.pdf. (2007)

France Telecom R & D, SmartQVT Documentation. http://smartqvt.elibel.tm.fr (2008)

Gallina, B., Guelfi, N., Mammar, A., Structuring Business Nested Processes Using UML 2.0 Activity Diagrams and Translating into XPDL. Proceedings of the 3rd GI-Workshop XML4BPM - XML Integration and Transformation for Business Process Management. Passau, Germany, February 20-22. pp: 281 - 296. (2006)

García, I., Rodriguez, A., Fernandez-Medina, E., Piattini, M., Implementación de Heurísticas en QVT para la obtención de Clases de Análisis a partir de Modelos de Proceso de Negocio Seguros. IV Taller sobre Desarrollo de Software Dirigido por Modelos, MDA y Aplicaciones (DSDM'07). Zaragoza, España, (2007)

Gardner, T.; Amsden, J.; Griffin, C.; Iyengar, S., Draft UML 1.4 Profile for Automated Business Processes with a mapping to BPEL 1.0. Version 1.1. IBM. http://www.ibm.com/developerworks/rational/library/content/04April/3103/3103_UMLProfileForBusinesProcesses1.1.pdf. (2003).

Gardner, T., UML Modelling of Automated Business Processes with a Mapping to BPEL4WS. http://www-128.ibm.com/developerworks/rational/library/4593.html (2004)

Garrido, S.D., Modelado de workflow con redes de Petri coloreadas condicionales. Master's thesis, Instituto Politécnico Nacional. México (2005)

Gronmo, R., Jaeger, M.C., Model-Driven Semantic Web Service Composition, 12th Asia-Pacific Software Engineering Conference (APSEC), Taipei, Taiwan (2005)

Guelfi, N.; Mammar, A., A formal framework to generate XPDL specifications from UML activity diagrams. Proceedings of the 2006 ACM symposium on Applied computing, (2006)

INRIA, The Atlas Transformation Language (ATL). http://modelware.inria.fr/rubrique12.html, last access: 10/04/2008, (2008)

Kalnins, A., Vitolins, V., Use of UML and Model Transformations for Workflow Process Definitions. TECHNIKA, http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0607044, (2006) pp. 3-15

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

Kurtev, I., State of the Art of QVT: A Model Transformation Language Standard, Applications of Graph Transformations with Industrial Relevance, Third International Symposium, AGTIVE 2007, Kassel, Germany, October 10-12, 2007, Revised Selected and Invited Papers. Section: Queries, Views, and Model Transformations. LNCS, Vol 5088, pp.: 377 - 393. Publisher: Springer-Verlag Berlin, Heidelberg (2008)

Kuznetsov, M. B., UML Model Transformation and Its Application to MDA Technology, Programming and Computer Software, Vol. 33, No. 1, pp. 44–53. Pleiades Publishing Ltd., 2007. Original Russian Text published in Programmirovanie, Vol. 33, No. 1. ISSN 0361-7688, (2007)

Lohmann, C., Greenyer, J., Jiang, J., Applying Triple Graph Grammars For Pattern-Based Workflow Model Transformations. Proceedings of the Journal of Object Technology, (2007).

McNeile, A., MDA: The Vision with the Hole?, http://www.metamaxim.com/download/documents/MDAv1.pdf, (2003)

Mens, T., Czarnecki, K., Pieter Van Gorp, P., A Taxonomy of Model Transformations, Electronic Notes in Theoretical Computer Science (ENTCS) Volume 152, (March 2006), pp. 125-142, ISSN: 1571-0661. Elsevier Science Publishers B. V. Amsterdam, The Netherlands (2006)

Mora, B.; Ruiz, F.; García, F.; Piattini, M., Experiencia en Transformación de Modelos de Procesos de Negocios desde BPMN a XPDL, Ideas'07, X Workshop Iberoamericano de Requisitos y Ambientes de Software, Venezuela, May 26, 2007. (2007)

MOLA Project, http://mola.mii.lu.lv, Last access: 03/04/2008, (2008)

OASIS, WS-BPEL 2.0 Specification. Technical report, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf (2007)

OMG, Meta Object Facility (MOF) 2.0 Adopted Specification. Technical report (2003a)

OMG, Model Driven Architecture Guide Version 1.0.1. Document number: omg/2003-06-01 (2003b)

OMG, The Object Constraint Language Specification. Version 2.0. Technical report (2004)

OMG, Unified Modeling Language: Superstructure. Version 2.0. Formal/05-07-04, http://www.omg.org/spec/UML/2.0/Superstructure/PDF/ (2005)

OMG, Business process modeling notation specification. Final Adopted Specification dtc/06-02-01, http://www.bpmn.org/Documents/OMG_Final_Adopted_BPMN_1-0_Spec_06-02-01.pdf, (2006)

OMG, Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. Final adopted specification ptc/07-07-07, (2007)

Owen, M., Raj, J., BPMN and Business Process Management: Introduction to the New Business Process Modeling Standard. Popkin Software (2003)

Rodríguez, A., Fernández-Medina, E., Piattini, M., Using QVT to obtain Use Cases from Secure Business Processes modeled with BPMN, 8º Workshop on Business Process Modeling, Development, and Support (BPMDS), June 2007, Trondheim, Norway (2007) pp:319-323

Rodríguez, A., Fernández-Medina, E. y Piattini, M., Towards Obtaining Analysis-Level Class and Use Case Diagrams from Business Process Models, 4º International Workshop on Foundations and Practices of UML (FP-UML), Barcelona, España. Lecture Notes in Computer Science Volumen 5232, (2008). pp:103-112.

Russel, N., van der Aalst, W., ter Hofstede, A.H.M., Edmond, D., Workflow Resource Patterns: Identification, Representation and Tool Support. In: Proc. of the 17th Conference on Advanced Information Systems Engineering. Volume 3520 of LNCS. Springer, Berlin (2005) pp.216–232

Russel, N., van der Aalst, W., ter Hofstede, A.H.M., Wohed, P., On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling. Technical Report BPM-06-03, BPM Center (2006)

Shapiro, R.M., XPDL 2.0: Integrating Process Interchange and BPMN. Workflow Handbook. (2006)

W3C, Web Services Conversation Language (WSCL) 1.0. Technical report, http://www.w3.org/TR/2002/NOTE-wscl10-20020314/ (2002a)

W3C, Web Service Choreography Interface (WSCI) 1.0. Technical report, http://www.w3.org/TR/2002/NOTE-wsci-20020808/ (2002b)

Watson, A.: OMG's new modeling specifications. Keynote speech, ECMDA-FA 2005, Nuremberg, Germany, November 2005, (2005)

White, S. A. P., Process Modelling Notations and Workflow Patterns, http://www.omg.org /bp-corner/bp-files/Process_Modeling_Notations.pdf (2004)

White, S., Using BPMN to Model a BPEL Process. BPTrends, http://www.bptrends.com/publicationfiles/03-05%20wp%20mapping%20bpmn%20to%20bpel-%20white.pdf (2005)

Workflow Management Coalition: XML Process Definition Language (WFMCTC-1025). Technical report (2005)

## Appendix. XPDL specification for the business process “Book Consignment”.

```
<Package>
 <Pools>
  <Pool Process=”PE” Id=”E” Name=”Publisher” BoundaryVisible=”true”>
   <Lanes>
    <Lane Id=”E0” Name=”Publisher”></Lane>
   </Lanes>
  </Pool>
  <Pool Process=”PL” Id=”L” Name=”Bookshop” BoundaryVisible=”true”>
   <Lanes>
    <Lane Id=”L0” Name=”Warehouse”></Lane>
    <Lane Id=”L1” Name=”Showroom”></Lane>
    <Lane Id=”L2” Name=”Accounting”></Lane>
    <Lane Id=”L3” Name=”Administration”></Lane>
   </Lanes>
  </Pool>
 </Pools>
 <Participants>
  <Participant Id=”EP” Name=”Publisher”>
   <ParticipantType=”ROLE” />
  </Participant>
  <Participant Id=”LP” Name=”Bookshop”>
   <ParticipantType=”ROLE” />
  </Participant>
 </Participants>
 <Artifacts>
  <Artifact Id=”LN1D” Name=”Delivery Note” ArtifactType=”DataObject”>
   <DataObject Id=”LN1” Name=”Delivery Note” />
  </Artifact>
  <Artifact Id=”LN3D” Name=”Delivery Note” ArtifactType=”DataObject”>
   <DataObject Id=”LN3” Name=”Delivery Note” />
  </Artifact>
  <Artifact Id=”LN6D” Name=”Invoice” ArtifactType=”DataObject”>
   <DataObject Id=”LN6” Name=”Invoice” />
  </Artifact>
  <Artifact Id=”LN11D” Name=”Stock” ArtifactType=”DataObject”>
   <DataObject Id=”LN11” Name=”Stock” />
  </Artifact>
 </Artifacts>
 <MessageFlows>
  <MessageFlow Id=”LA5” Source=”LN5” Target=”EN3” />
  <MessageFlow Id=”LA9” Source=”LN9” Target=”EN4” />
 </MessageFlows>
 <Associations>
  <Association Id=”EA2” Source=”EN2” Target=”LN1D” AssociationDirection=”None”>
  <Association Id=”EA3” Source=”EN3” Target=”LN6D” AssociationDirection=”None”>
  <Association Id=”LA1” Source=”LN1D” Target=”LN2” AssociationDirection=”None”>
  <Association Id=”LA2” Source=”LN2” Target=”LN3D” AssociationDirection=”None”>
  <Association Id=”LA3” Source=”LN3D” Target=”LN4” AssociationDirection=”None”>
  <Association Id=”LA6” Source=”LN6D” Target=”LN7” AssociationDirection=”None”>
  <Association Id=”LA10” Source=”LN10” Target=”LN11D” AssociationDirection=”None”>
  <Association Id=”LA11” Source=”LN11D” Target=”LN5” AssociationDirection=”None”>
 </Associations>
 <WorkflowProcesses>
  <WorkflowProcess Id=”PE” Name=”Publisher”>
   <Actitities>
    <Activity Id=”EN1” Name=””>
     <StartEvent Trigger=”Timer”>
      <TriggerTimer TimeCycle=”Monthly” />
     </StartEvent>
    </Activity>
    <Activity Id=”EN2” Name=”Deliver news service”>
     <Implementation>
      <No/>
     </Implementation>
    </Activity>
    <Activity Id=”EN3” Name=”Send invoice”>
     <Implementation>
      <No/>
```

```
    </Implementation>
   </Activity>
   <Activity Id="EN4" Name="Receive payment">
    <Implementation>
     <No/>
    </Implementation>
   </Activity>
   <Activity Id="EN5" Name="">
    <EndEvent Result="Terminate" />
   </Activity>
  </Activities>
  <Transitions>
   <Transition Id="EA1" Name="" From="EN1" To="EN2" />
   <Transition Id="EA4" Name="" From="EN4" To="EN5" />
  </Transitions>
 </WorkflowProcess>
 <WorkflowProcess Id="PL" Name="Bookshop">
  <ActivitySets>
   <ActivitySet Id="LN10AS">
    <Activities>
     <Activity Id="SN1" Name="">
      <StartEvent Trigger="None" />
     </Activity>
     <Activity Id="SN2" Name="Exist?">
      <Route GatewayType="XOR" XORType="Data" MarkerVisible="true" />
      <TransitionRestrictions>
       <TransitionRestriction>
        <Split Type="XOR">
         <TransitionRefs>
          <TransitionRef Id="SA2" />
          <TransitionRef Id="SA3" />
         </TransitionRefs>
        </Split>
       </TransitionRestriction>
      </TransitionRestrictions>
     </Activity>
     <Activity Id="SN3" Name="Update Stock">
      <Implementation>
       <Task>
        <TaskScript />
       </Task>
      </Implementation>
     </Activity>
     <Activity Id="SN4" Name="Add Book">
      <Implementation>
       <Task>
        <TaskScript />
       </Task>
      </Implementation>
     </Activity>
     <Activity Id="SN5" Name="Put Price Label">
      <Implementation>
       <Task>
        <TaskScript />
       </Task>
      </Implementation>
     </Activity>
     <Activity Id="SN6" Name="Cost Changed?">
      <Route GatewayType="XOR" XORType="Data" MarkerVisible="true" />
      <TransitionRestrictions>
       <TransitionRestriction>
        <Split Type="XOR">
         <TransitionRefs>
          <TransitionRef Id="SA7" />
          <TransitionRef Id="SA8" />
         </TransitionRefs>
        </Split>
       </TransitionRestriction>
      </TransitionRestrictions>
     </Activity>
     <Activity Id="SN7" Name="Update Cost">
```

```
  <Implementation>
   <Task>
    <TaskScript />
   </Task>
  </Implementation>
 </Activity>
 <Activity Id="SN8" Name="">
  <Route GatewayType="XOR" XORType="Data" MarkerVisible="true" />
  <TransitionRestrictions>
   <TransitionRestriction>
    <Join Type="XOR" />
   </TransitionRestriction>
  </TransitionRestrictions>
 </Activity>
 <Activity Id="SN9" Name="Put new price labels">
  <Implementation>
   <Task>
    <TaskScript />
   </Task>
  </Implementation>
 </Activity>
 <Activity Id="SN10" Name="">
  <Route GatewayType="XOR" XORType="Data" MarkerVisible="true" />
  <TransitionRestrictions>
   <TransitionRestriction>
    <Join Type="XOR" />
   </TransitionRestriction>
  </TransitionRestrictions>
 </Activity>
 <Activity Id="SN11" Name="">
  <EndEvent Result="None" />
 </Activity>
 <Activity Id="SN12" Name="Put in bookcase">
  <Implementation>
   <Task>
    <TaskScript />
   </Task>
  </Implementation>
 </Activity>
 <Activity Id="SN13" Name="Withdraw outdated copies">
  <Implementation>
   <Task>
    <TaskScript />
   </Task>
  </Implementation>
 </Activity>
 <Activity Id="SN14" Name="Return copies to the shelves">
  <Implementation>
   <Task>
    <TaskScript />
   </Task>
  </Implementation>
 </Activity>
</Activities>
<Transitions>
 <Transition Id="SA1" Name="" From="SN1" To="SN2" />
 <Transition Id="SA2" Name="" From="SN2" To="SN3">
  <Condition Type="Condition">si</Condition>
 </Transition>
 <Transition Id="SA3" Name="" From="SN2" To="SN4" >
  <Condition Type="Condition">no</Condition>
 </Transition>
 <Transition Id="SA4" Name="" From="SN4" To="SN5" />
 <Transition Id="SA5" Name="" From="SN3" To="SN6" />
 <Transition Id="SA6" Name="" From="SN12" To="SN10" />
 <Transition Id="SA7" Name="" From="SN6" To="SN8">
  <Condition Type="Condition">no</Condition>
 </Transition>
 <Transition Id="SA8" Name="" From="SN6" To="SN7">
  <Condition Type="Condition">si</Condition>
 </Transition>
```

```
      <Transition Id="SA9" Name="" From="SN7" To="SN13" />
      <Transition Id="SA10" Name="" From="SN14" To="SN8" />
      <Transition Id="SA11" Name="" From="SN8" To="SN10" />
      <Transition Id="SA12" Name="" From="SN10" To="SN11" />
      <Transition Id="SA13" Name="" From="SN5" To="SN12" />
      <Transition Id="SA14" Name="" From="SN13" To="SN9" />
      <Transition Id="SA15" Name="" From="SN9" To="SN14" />
     </Transitions>
    </ActivitySet>
   </ActivitySets>
   <Actitities>
    <Activity Id="LN2" Name="Check Delivery Note">
     <Implementation>
      <Task>
       <TaskManual />
      </Task>
     </Implementation>
    </Activity>
    <Activity Id="LN4" Name="Record Delivery Note">
     <Implementation>
      <Task>
       <TaskUser />
      </Task>
     </Implementation>
    </Activity>
    <Activity Id="LN5" Name="Make statement">
     <Implementation>
      <Task>
       <TaskReceive />
      </Task>
     </Implementation>
    </Activity>
    <Activity Id="LN7" Name="Record invoice in account">
     <Implementation>
      <Task>
       <TaskUser />
      </Task>
     </Implementation>
    </Activity>
    <Activity Id="LN8" Name="Prepare payment">
     <Implementation>
      <Task>
       <TaskManual />
      </Task>
     </Implementation>
    </Activity>
    <Activity Id="LN9" Name="Record payment">
     <Implementation>
      <Task>
       <TaskUser />
      </Task>
     </Implementation>
    </Activity>
    <Activity Id="LN10" Name="Stock and Prices Checking" >
     <BlockActivity ActivitySetId="LN10AS" />
     <Loop LoopType="MultiInstance">
      <LoopMultiInstance MI_Ordering="Sequential" />
     </Loop>
    </Activity>
   </Activities>
   <Transitions>
    <Transition Id="LA7" Name="" From="LN7" To="LN8" />
    <Transition Id="LA8" Name="" From="LN8" To="LN9" />
   </Transitions>
  </WorkflowProcess>
 </WorkflowProcesses>
</Package>
```