19. B2B Applications, BPEL4WS, Web Services and .NET in the Context of MDA

Jean Bézivin[1], Slimane Hammoudi[2],
Denivaldo Lopes[1,2] and Frédéric Jouault[1,3]
1 University of Nantes Email: Jean.Bezivin@lina.univ-nantes.fr
2 ESEO Email: {shammoudi, dlopes}@eseo.fr
3 TNI-Valiosys Email: Frederic.Jouault@lina.univ-nantes.fr

Recently, Model-Driven Architecture (MDA) has been proposed to take into account the development of large software systems, such as B2B applications on the Internet. However, before this becomes a reality, some issues need solutions, such as the definition of various Domain-Specific Languages (DSL) and also automatic transformation between these domain languages representing business concerns and those offering platform executability. In this paper, we provide some insights into transformation between some specific DSL particularly relevant to Business-to-Business (B2B) applications.

1. INTRODUCTION

Business-to-Business (B2B) applications existed before the Web. On the one hand, the emergence of the Internet and its services (such as the Web) has diffused the B2B applications. On the other hand, B2B applications on the Internet are often large and complex software systems.

Model Driven Architecture (MDA™)[71] (OMG, 2001) has been proposed for supporting the development of large software systems, such as B2B applications. However, before this becomes a reality some issues need to be resolved such as the model transformation. The objective of this paper is to provide some insights into the creation of meta-models, mappings and model transformation rules. We proceed here using UML[72] to create Platform-Independent Models (PIM) and transforming them into a Platform-Specific Models (PSM) based on Business Process Execution Language for Web Services (BPEL4WS) (Tony Andrews, 2003), Web Service (W3C, 2004) and .NET (Alex Ferrata, 2002). The model transformation is described using the Atlas Transformation Language (ATL) (Jean Bézivin1, 2003).

This paper is organized as follows. Section 2 is an overview of B2B applications in the context of MDA. Section 3 presents meta-models and mappings. Section 4 discusses some problems found during our research. The last section is the conclusion of our research.

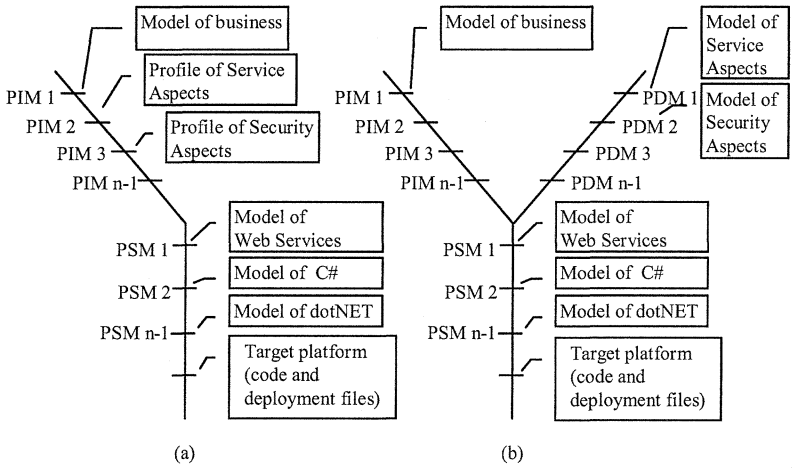
---

[71] MDA™ is a trademark of the Object Management Group.
[72] Unified Modeling Language (UML) version 1.4.

2. OVERVIEW

One of the first domains in which computers were applied was B2B applications. In fact, B2B applications have profited from and financed advances in computer science.

Recently, the computer science community was confronted to new problems, such as the fast evolution of B2B applications, and the integration between different technologies. In order to make face to this new context, a change of paradigm was necessary. On the one hand, the object paradigm has given all that it could and does not seem in a position for giving much more. On the other hand, the service and the model paradigm have been developed and applied for meeting these new requirements.



* This work had been done in the context of the INTEROP European Network of Excellence.

PIM –Platform-Independent Model PSM – Platform-Specific Model
PDM – Platform-Description Model
Figure 1: B2B Application Design and MDA

Figure 1 presents the B2B application design in the context of MDA. According to this figure, two techniques are possible: (a) marking and (b) weaving. Marking is based on UML Profiles for decorating a Platform-Independent Model (PIM) with aspects such as services and security. Weaving is based on the idea of making a texture between a PIM and a Platform-Description Model (PDM) (Jean Bézivin, 2002) before generating a Platform-Specific Model (PSM). In our research, we have privileged the weaving technique, but we will not describe this technique here. In this paper, we are more concerned by the model transformation that is one of the main challenges of the MDA Approach. According to figure 1, a PIM (e.g. business model) is transformed into a PSM (e.g. based on Web Service and BPEL4WS), refined in other PSMs (e.g. based on C# and .NET), until exported as code, deployment files, and config files. So, many levels of PIM and PSM are possible.

3. FROM UML INTO BPEL4WS, WEB SERVICE AND .NET

Before applying a model transformation for generating a target model from a source model, we need to obtain two things: a meta-model of each participant (i.e. BPEL4WS, Web Services and .NET) and a mapping from one into another meta-model. For this purpose, this section presents a meta-model for each platform and mappings.

3.1 Business Process and BPEL4WS Meta-model

Business process languages can support the definition and execution of a business process. Some business process languages were created for defining generic business processes (Assaf Arkin, 2002). Other business process languages consider a process as a composite Web Service (Frank Leymann, 2001). Other process languages were created for defining processes using Workflow (WfMC, 2002). So, a business process can be created in different ways, either as generic or a specialized one.

BPEL4WS (Tony Andrews, 2003) defines a model and a grammar for describing the behavior of a business process. BPEL4WS is the result of the merging of WSFL and XLANG (Satish Thatte, 2001). In fact, it has some concepts of WSFL and XLANG, such as directed graphs and block-structured language, respectively. It depends on the WSDL (W3C, 2001b), i.e. a BPEL4WS Process makes references to portTypes of the services involved (see section 3.3). In the right side of Figure 2 is presented a BPEL4WS meta-model (fragment) with the following main elements:

- Process – composed of activities, partners, correlation sets, fault handlers and compensation handlers. It has some attributes such as abstractProcess specifying whether the process is defined as abstract or as executable.
- PartnerLinks - defines the different parties that interact within a business process in execution. It is characterized by a PartnerLink that specifies the conversational relationship between two services through the declaration of their roles. Each role specifies only one WSDL portType that a partner needs to implement.
- Partners - defined as a subset of PartnerLink (i.e. references), it introduces a constraint on the functionality that a business partner provides.
- Variables - defines the data variables used by a process, and allows processes to maintain state data and process history based on messages exchanged. A variable can be defined in terms of WSDL message types (i.e. messageType) or XML Schema simple types (i.e. type) or XML Schema elements (i.e. element).
- Activity - structured in some parts such as control flows (e.g. Switch, While), message flow (e.g. Invoke, Receive, Reply), data flow (data is transferred between activities using Assign), transaction flow (Long-Running Transaction is only supported within a single business process) and extensibility (name-space codified on URI can be used in some BPEL4WS elements).

3.2 Mapping from UML into BPEL4WS

In Figure 2, we use a graphical notation for illustrating a mapping from UML (activity diagram) into BPEL4WS. The UML meta-model is presented on the left side, the mapping in the center, and the BPEL4WS meta-model on the right side. The mapping part is formed by four main graphical elements: connection (source and target), association, transformation rule and composition.

A connection links one or more meta-model element(s) to a transformation rule. The association shows a relationship between rules. The composition shows a tight relationship between rules. The transformation rule takes a source element and generates a suitable target element. We have used ATL transformation language for defining transformation rules (Jean Bézivin, 2003).

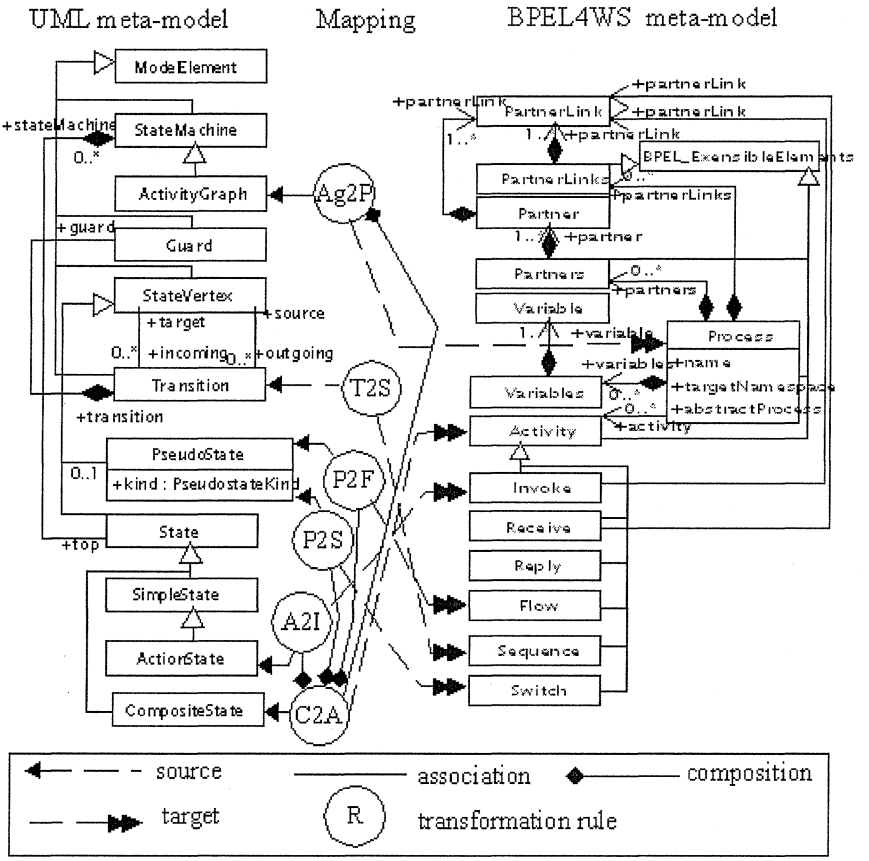


Figure 2 : Mapping from UML (activity diagram) into BPEL4WS

According to Figure 2, we have the following mappings (we will present only a few ATL transformation rules). (A transformation rule takes an element from a source meta-model and transforms it into an equivalent element in a target meta-model. Elements are equivalents, if they have equivalent semantics and structures. The rule P2S take a UML PseudoState of type 'Choice' and transform it into BPEL Switch.)

The UML PseudoState (Choice) is mapped into BPEL4WS Switch through the rule P2S:

-- Helper for P2S

helper context UML!PseudoState def:

helperGetCondition(): Collection(BPEL!BooleanExpr) =

-- (body omitted due to space limitations)

```
helper context UML!PseudoState def:
helperGetActivity():Collection(Activity) =
    -- (body omitted due to space limitations)

helper context UML!PseudoState def:
helperGetOtherwise():Collection(Activity) =
    -- (body omitted due to space limitations)
rule P2S{
 from ps : UML!PseudoState(ps.kind=#pk_choice)
 to sw: BPEL!Switch
 mapsTo ps(
   name ← ps.name,
   case ← cases,
   otherwise ← otherw   ),
 cases : BPEL:Case(
  condition ← helperGetCondition(ps),
  ref ← helperGetActivity(ps)   ),
 otherw : BPEL:Otherwise(
  ref ← helperGetOtherwise(ps)   ) }
```

3.3 Web Service Meta-model

A service is an abstraction of programs, business process, and other artifacts of software defined in terms of what it does. Services can be organized in a Service Oriented Architecture (SOA) (W3C, 2004). SOA is a form of distributed system architecture based on the concept of services that is characterized by the following properties (W3C, 2004):

- Logical view - a service is a logical view of a system.
- Message oriented - the communication between an agent provider and an agent requester is defined in terms of messages exchanged.
- Description orientation - a service is described using meta-data.
- Granularity - services communicate using a small number of large and complex messages.
- Platform neutral - services communicate using messages codified in one platform-independent representation.

Figure 3 presents a simplified SOA model. An agent provider has services. These services are described through a meta-data representation, i.e. ServiceDescription. Afterwards, the agent provider stores information of its services in a Registry. An agent requester searches in the Registry for a specific service following a determined criterion. The Registry returns information of a desired service. The agent requester finds the meta-data of this service and uses it to exchange messages with the service. According to this figure and Web Service Architecture (W3C, 2004), we have the

following similarities: Universal Description, Discovery, and Integration (UDDI) (UDDI.ORG, 2002) is the registry; Web Service Description Language (WSDL) (W3C, 2001b) is the service description; the service uses Simple Object Access Protocol (SOAP) (W3C1, 2001a) as a communication protocol for exchanging messages.

In this paper, we will only present a meta-model for WSDL. In the right side of Figure 4, a WSDL meta-model is presented. It is formed by:

- Definition - the main element of this meta-model which has a set of imports, types, messages, portTypes[73], bindings and services.
- ImportType - allows the association of a namespace with a document location.
- TypesType - employed for defining a simple or complex type defined by XML Schema.
- MessageType - describes the abstract format of a particular message that a Web Service sends or receives. It has parts (i.e. PartTypes) which describe each part of a message.
- portTypeType - defines the interface of a service. It has a set of messages that a service sends and/or receives.
- BindingType - describes a concrete binding of interface components, i.e., describes how to call up a service.
- ServiceType - describes a service, its interface (i.e. PortTypeType) and its endpoints.

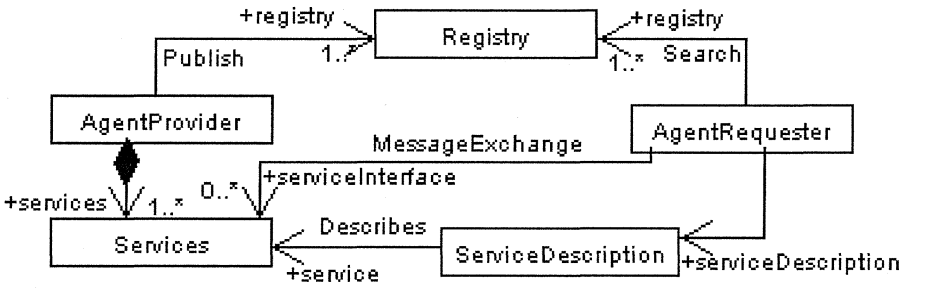


Figure 3: Service-Oriented Architecture

3.4 Mapping from UML into Web Service

Figure 4 presents a mapping from UML (class diagram) into WSDL.
According to Figure 4, we have the following mappings (we will present only a few ATL transformation rules). (UML Operation is equivalent to WSDL OperationType. An Operation Type is composed of Paramtype,..., Message.)

The UML Operation is mapped into WSDL OperationType through the rule O2O:

rule O2O{
from op : UML!Operation
to wsdlop : WSDL!OperationType
mapsTo op(name ← op.name, parameterOrder ← op.getOrder(),
input ← inp, output ← out),

[73] PortType was renamed to Interface in WSDL 1.2

inp : WSDL!ParamType (message ← inp_m),
out : WSDL!Paramtype(message← outp_m),
inp_m : WSDL!Message(name ← op.owner.name + '_' + op.name),
outp_m: WSDL!Message(name ← op.owner.name + '_' +
op.name + 'Response'),
wsdlob: WSDL!BindingOperationType (name ← op.name,
input ← inputb, output ← outputb)
inputb : WSDL!StartWithExtensionType(
encodingStyle ← http://' + 'schemas.xmlsoap.org/soap/encoding/',
use ← 'encoded',
namespace ← 'urn://'+ op.feature.owner.name + '.wsdl'),
outputb : WSDL!StartWithExtensionType(
encodingStyle ← 'http://' + 'schemas.xmlsoap.org/soap/encoding/'
use ← 'encoded',
namespace ← 'urn://'+
op.feature.owner.name + '.wsdl') }

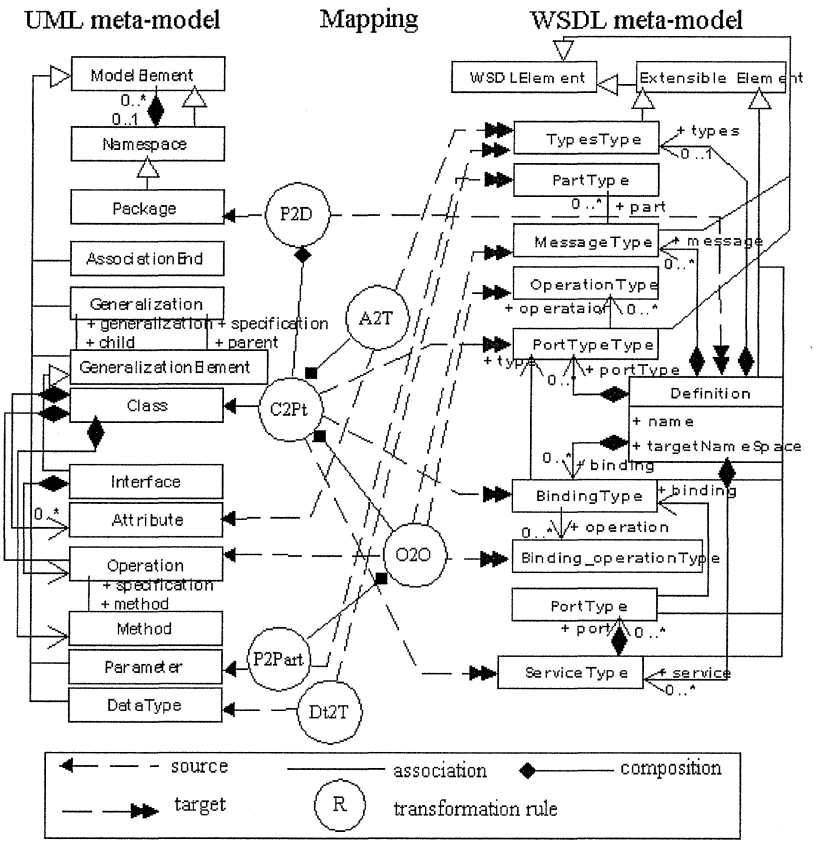


Figure 4 : Mapping from UML (class diagram) into WSDL

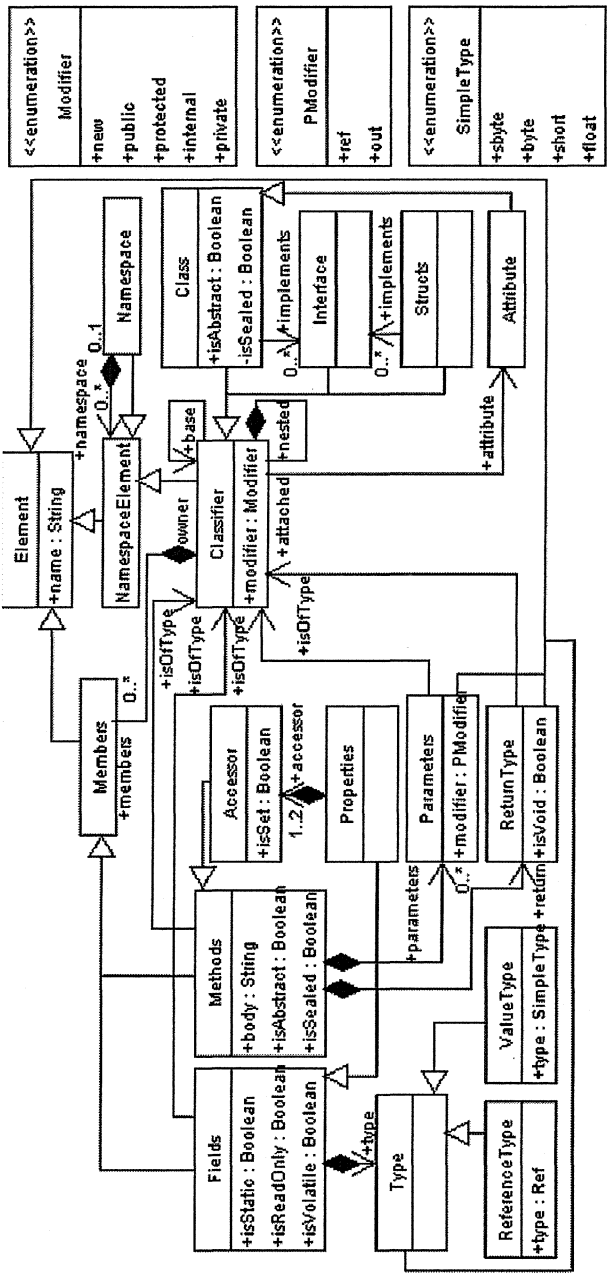


Figure 5: C# meta-model (fragment)

3.4 .NET meta-model

In our research, we have used .NET as target platform. The .NET platform for Web Services can be formed by C#, .NET Framework and Internet Information System (ISS) (Alex Ferrata, 2002). Figure 5 depicts a C# meta-model (fragment).

This C# meta-model is formed by:

- Namespace - a container for all other elements.
- Classifier - a generalization for members, classes, interfaces and structures. It has a set of members that can be fields, methods, and properties.
- Class - a specialization for Classifier and implements interfaces.
- Interface - another specialization for Classifier (only with methods' signatures).
- Structs - similar to Class, but it does not have heritage.
- Attribute – a declarative information that can be attached to programs' entities (such as Class and Methods) and retrieved at runtime. All attribute classes derive from the System.Attribute base class provided by the .NET Framework. Although Attribute belongs to C# API (i.e. model or M1 layer), we have used it as part of the C# metamodel, in order to manipulate it in the meta-model layer (i.e. M2 layer). C# Attribute does not have the same meaning of UML Attribute. In UML, an attribute is a feature within a classifier that describes a range of values whose type is a classifier.
- Fields - a composition of one type that can be a ValueType or a ReferenceType.
- Methods - has the signature of operations (return type, identifier and parameters).
- The creation of a Web Service in .NET using C# is made using the classes:
- WebService - base class for all Web Services.
- WebServiceAttribute - an attribute that can be associated with a class implementing a service.
- WebMethodAttribute - an attribute associated with the methods of a class implementing a service.

Figure 6 presents a simplified template and .NET meta-model, detailing the main elements for creating a Web Service. The template indicates that a Web Service is implemented using a class that extends the WebService class. This class is attached to the attribute named WebServiceAttribute. The methods accessed as services are attached to the attribute named WebMethodAttribute. The .NET template uses the Application Programming Interface (API) from .NET Framework to define Web Services, thus this template belongs to the model layer (or M1 layer). The .NET meta-model presents the deployment files web.config and disco needed to deploy a Web Service.

The deployment files Web.config and Disco are necessary for deploying Web Services using IIS and .NET framework. Web.config has specific information, which enables IIS for running a Web Service. Disco is employed for advertising a Web Service publicly and it has the location of the description of the service, i.e. WSDL.

3.5 Mapping from UML into .NET platform

Figure 7 presents a mapping from a UML (class diagram) into C# (fragment). According to Figure 7, we have the following mappings (we will present only a few ATL transformation rules). The UML Package is mapped into C# Namespace through the rule P2N:

```
rule P2N{
 from pck : UML!Package
 to cn : Csharp!Namespace
 mapsTo pck(
  name ← pck.name ) }
```

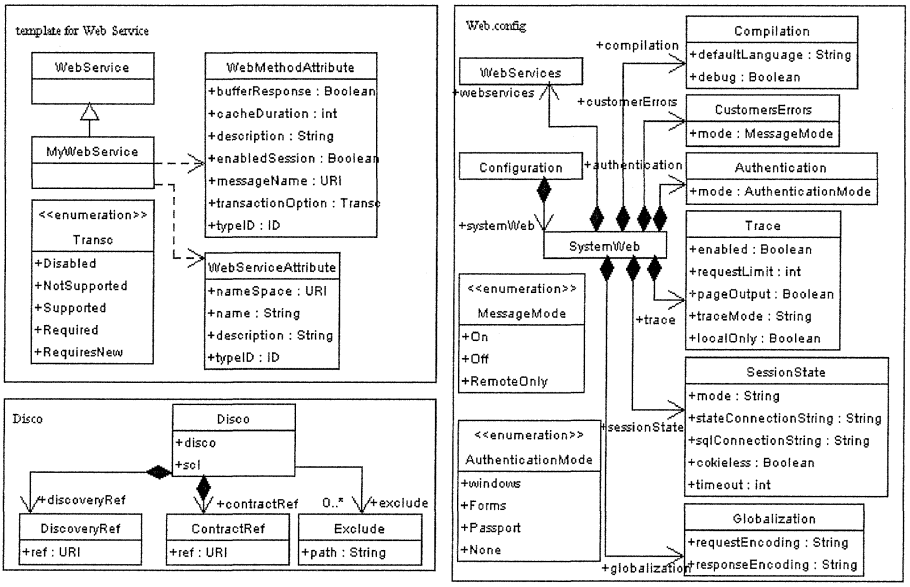


Figure 6 : .NET template and meta-model

The UML AssociationEnd is mapped into C# Field through the rule Ae2F:

-- Helper for getOtherEnd()
helper context UML!AssociationEnd def: getOtherEnd() :
UML!AssociationEnd =
self.association.connection-> select(e|e <> self)->first();
rule Ae2F{
from ae : UML!AssociationEnd
to cf : Csharp!Field
mapsTo ae(
name ← ae.name,
modifier ← if ae.visibility = #vk_public then #public else #private endif,
isVolatile ← false,
isStatic ← false,
isReadOnly ← false,
isOfType ← ae.participant,
owner ← ae.getOtherEnd().participant) }

Afterwards, a model generated in this step is refined using the template and the .NET meta-model.

4. DISCUSSION

During our research, we have been confronted with real and unexpected problems, which merit a brief discussion, hence:

- Is it relevant to take into account APIs in the MDA context? - Someone could doubt the importance to take into account APIs in the model driven approach. However, Web Services, EJB, CORBA and other platforms are implemented using APIs. Moreover, one of the characteristics of MDA is to consider everything as model, thus it should include APIs too.
- Is it relevant to take into account Attribute-Oriented Programming in the context of MDA? - C# uses attributes for attaching information to program's entities such as Class and Methods. This information can be used by tools for generating complementary code or deployment files, or it can be retrieved at runtime. For developing Web Services in .NET, we need to use attributes. Thus, in Figure 5 we presented a C# meta-model where Attribute is one of its elements. However, the importance of taking this into account in the meta-model layer must be examined more carefully.

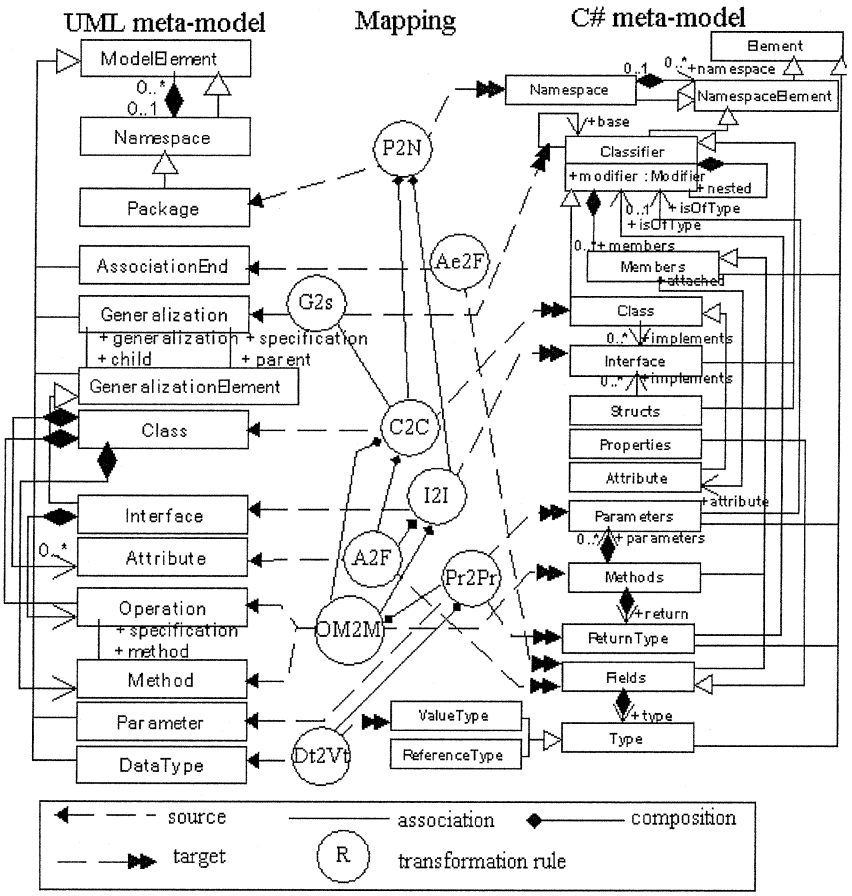


Figure 7 : Mapping from UML (class diagram) into C#

5. CONCLUSION

This paper emphasized solutions for some issues, mainly creation of meta-models and mapping specification for the development of B2B applications in Web Service

platforms using an MDA approach. However, the integration of security and availability in B2B models from the start is still an open issue. We started presenting a BPEL4WS, a Web Service, and a .NET meta-model, afterwards we discussed some possible mappings between the UML meta-model and these meta-models. The material presented in this paper is a good illustration of the current trend in model-driven engineering with such recent proposal as IBM's MDA manifesto (Grady Booch 2004).

REFERENCES

Tony Andrews, Francisco Curbera, Hitesh Dholakia, and *et al.* (2003) Business Process Execution Language for Web Services (BPEL4WS) version 1.1, May 2003.

Assaf Arkin (2002). Business Process Modeling Language (BPML), November 2002.

Jean Bézivin, Grégoire Dupé, Frédéric Jouault, Gilles Pitette, and Jamal Eddine Rougui (2003). First Experiments with the ATL Model Transformation Language: Transforming XSLT into XQuery. 2nd OOPSLA Workshop on Generative Techniques in the context of Model Driven Architecture, 2003.

Jean Bézivin and Sébastian Gérard (2002). A Preliminary Identification of MDA Components. OOPSLA 2002 Workshop on Generative Techniques in the context of Model Driven Architecture, 2002.

Alex Ferrata and Matthew MacDonald (2002). Programming .NET Web Services. O'Reilly & Associates, 1st edition, September 2002.

Frank Leymann (2001). Web Services Flow Language (WSFL 1.0), May 2001.

OMG (2001). Model Driven Architecture (MDA)- document number ormsc/2001-07-01, 2001.

Satish Thatte (2001). XLANG - Web Services for Business Process Design, 2001.

UDDI.ORG (2002). Universal, Description, Discovery and Integration (UDDI) Version 3.0, July 2002.

W3C (2001a). Simple Object Access Protocol (SOAP) 1.1, May 2001.

W3C (2001b). Web Services Description Language (WSDL) 1.1, March 2001.

W3C (2004). Web Services Architecture (WSA), February 2004. (NOTE-ws-arch-20040211)

WfMC (2002). Workflow Process Definition Interface – XML Process Definition Language (XPDL), October 2002.

Grady Booch, Alan Brown, Shridhar Iyengar, Jim Rumbaugh, Bran Selic (2004) An MDA Manifesto, The MDA Journal, Mai 2004, http://www.bptrends.com/publicationfiles