A Case Study on the Transformation of Context-Aware Domain Data onto XML Schemas

Cléver R. G. de Farias[1], Luís Ferreira Pires[2] and Marten van Sinderen[2]

[1]Department of Physics and Mathematics, University of São Paulo
Av. Bandeirantes 3900, 14040-901, Ribeirão Preto (SP), Brazil

[2]Centre for Telematics and Information Technology, University of Twente
PO Box 217, 7500 AE, Enschede, the Netherlands

Abstract. In order to accelerate the development of context-aware applications, it would be convenient to have a smooth path between the context models and the automated services that support these models. This paper discusses how MDA technology (metamodelling and the QVT standard) can support the transformation of high-level models of context-aware services onto the implementation of these services using web services. The total transformation process from context-aware services onto web services involves the following aspects: 1. service signatures, which should be translated onto WSDL definitions; 2. context-aware domain data used as input and output data in service operations, which should be translated onto XML schemas; and 3. service behaviours, which should be used to generate the service implementation. This paper concentrates on the modelling and transformation of the context-aware domain data. The results of this paper are generally applicable to the transformation of elements of any domain-specific language expressed in terms of a metamodel onto XML Schema data.

1 Introduction

Context-aware applications are capable of using information about the application user's environment in order to improve the quality of the service provisioning. Such applications are normally expensive to build and deploy, which justifies the current research interest on context-aware services platforms meant to facilitate the development of these applications [3]. Many context-aware platforms are based on context models, in which the user's context and context-aware services can be formalised.

In order to accelerate the development of context-aware applications, it would be convenient to have a smooth path between the context models and the automated services that support these models. More specifically, we have decided to experiment with the transformation of high-level models of context-aware services (as in [2]) onto the implementation of these services using web services. We have investigated the use of MDA technology, particularly metamodelling and the QVT standard, in order to perform this transformation.

R. G. de Farias C., Ferreira Pires L. and van Sinderen M. (2007).
A Case Study on the Transformation of Context-Aware Domain Data onto XML Schemas.
In *Proceedings of the 3rd International Workshop on Model-Driven Enterprise Information Systems*, pages 63-72
DOI: 10.5220/0002431500630072

The total transformation process from this context-aware service metamodel onto web services involves the following aspects: 1) service signatures, which should be translated onto WSDL definitions; 2) context-aware domain data used as input and output data in service operations, which should be translated onto XML schemas, and; 3) service behaviours, which should be used to generate the service implementation. This paper concentrates on the modelling and transformation of the context-aware domain data. The results of this paper are generally applicable to the transformation onto XML Schema data of elements of any domain-specific language expressed in terms of a metamodel.

The paper is further structured as follows: Section 2 describes the context-aware service metamodel we have used in this paper; Section 3 introduces the XML Schema metamodel that we have defined for our transformation; Section 4 discusses the transformation specification; Section 5 gives an example to illustrate the benefits of our approach; and finally Section 6 gives our conclusions.

2 Context-aware Service Metamodel

The Context-Aware Service (CWS) metamodel represents a set of concepts used to capture service definitions and their association to context-aware information. This metamodel was defined using the OMG MOF specification [7]. The major motivation behind the definition of a context-aware service metamodel is the possibility of producing general service descriptions independently of a particular platform, and relating such descriptions to context-aware information. Fig. 1 illustrates the concepts represented in the CWS metamodel. Our context-aware service metamodel has been discussed in detail in [2].

The CWS metamodel is organised around the metaclass Service. This metaclass represents a (context-aware) service provided by a service provider. Service has a single meta-attribute called name, which identifies the service. The metaclass Provider represents a service provider.

A service consists of a number of operations, represented by the metaclass ServiceOperation. ServiceOperation has two meta-attributes, viz., name, which identifies the operation, and interaction, which defines the type of interaction used by the operation. InteractionType defines the domain of possible values of the meta-attribute interaction, viz., oneway and request-response.

Operations are composed of input, output, or fault messages. A message is represented by the metaclass Message. This metaclass has a single meta-attribute, called name, which represents the message identifier. A message is constructed based on message parts, represented by the metaclass MessagePart. This metaclass has two meta-attributes, viz., name, which represents the message part identifier, and type, which represents the type of the message part. The type of the message part may refer to a primitive datatype or to a CWClassifier.

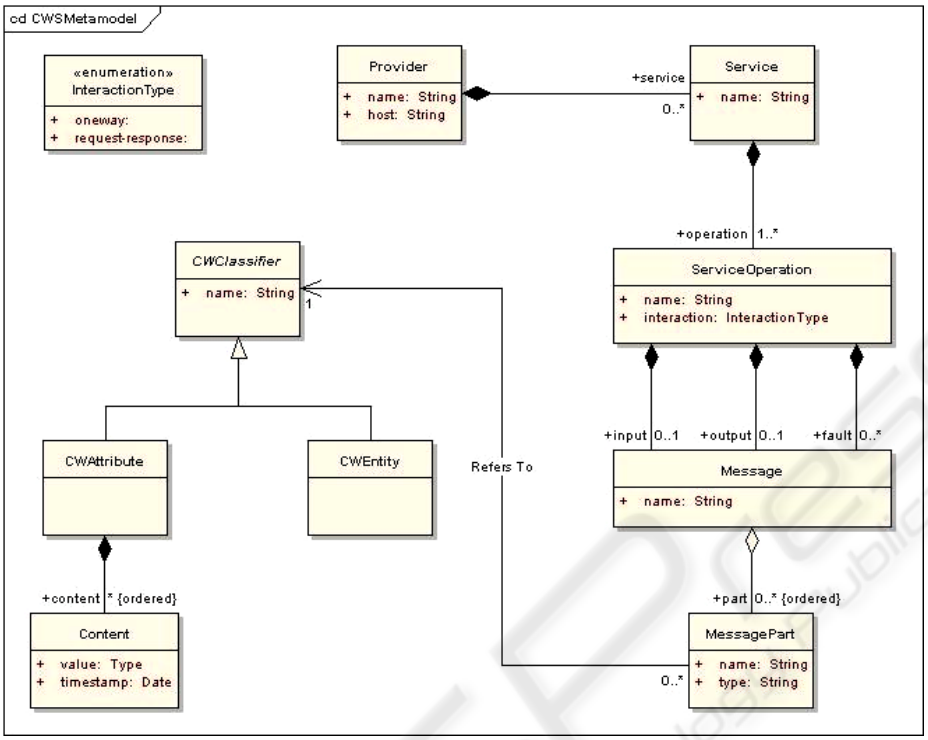

Fig. 1. CWS metamodel.

3 XML Schema Definition Metamodel

Our approach towards the model-driven development of context-aware services considers the use of web services as a target platform to implement these services. In order to facilitate the implementation of these services as web services, we have defined a metamodel for the Web Service Description Language (WSDL) [9] using MOF. This metamodel is structured around three packages [1], namely XSD, WSDLCore and BindingExtension. The XML Schema Definition (XSD) package contains the concepts used to represent XML schemas. This metamodel is a simplified representation of the XML Schema Definition language [10]. The WSDLCore package contains most of the concepts related to WSDL. The BindingExtension package contains the concepts associated to the SOAP [8] binding extension to WSDL. Since the focus of this paper is on the transformation of context-related information onto general-purpose XML Schema data, we concentrate on the description of the XSD package. Fig. 2 shows the concepts represented in the XSD metamodel.

The XSD metamodel is organised around the metaclass Schema, which represents the structure of an XML document.

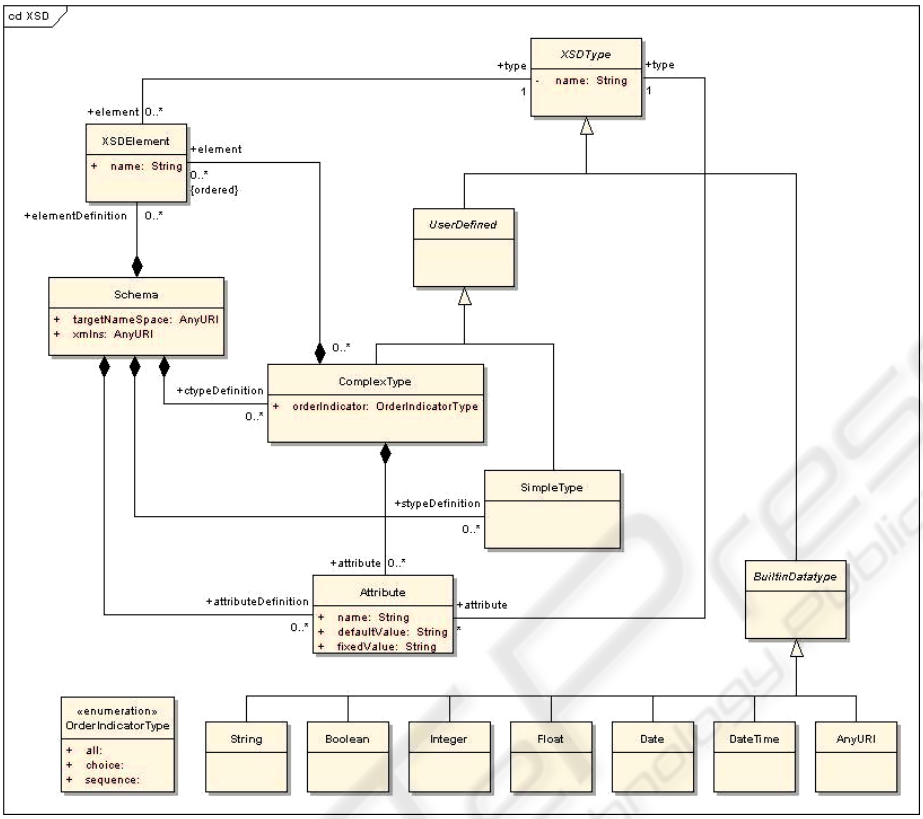

Fig. 2. XSD metamodel.

An XML schema defines the structure of an XML document through the specification of a number of element, attribute and user defined type definitions. The metaclass XSDElement represents an element definition. This metaclass has a single meta-attribute called name, which uniquely identifies the element within the document namespace.

Each defined XSD element has an associated abstract data type definition, represented by the abstract metaclass XSDType. This metaclass has a single attribute called name, which uniquely identifies the data type within the document namespace. There are two kinds of datatypes, viz., user-defined datatypes and built-in datatypes, represented by the abstract metaclasses UserDefined and BuiltinDatatype, respectively.

The metaclass BuiltinDatatype represents a number of pre-defined datatypes. In the context of this work, we consider a subset of the built-in datatypes defined in the XML schema language specification, namely string, boolean, integer, float, date, date and time and any URI. Each one of these primitive datatypes is represented by the metaclasses String, Boolean, Integer, Float, Date, DateTime and AnyURI, respectively.

The metaclass UserDefined represents a user-defined datatype specification. This metaclass was introduced to facilitate the structuring of datatypes. This metaclass is specialized into the metaclasses SimpleType and ComplexType.

The metaclass SimpleType represents a user-defined datatype that contains only text, i.e., it cannot contain any other XSD elements or attributes. The metaclass ComplexType represents a user-defined datatype that can contain other XSD elements and/or attribute definitions. This metaclass has a single meta-attribute called orderIndicator, which defines the ordering of the child XSD elements. Three different ordering indicators are defined, viz., all, choice and sequence. The all ordering indicator specifies that the child elements can appear in any order, and that each child element must occur only once. The choice ordering indicator specifies that only one of the contained child elements can appear. The sequence ordering indicator specifies that child elements must appear in a specific (sequence) order.

The metaclass Attribute represents an XSD attribute. Attributes are used to provide additional information about an element. This metaclass has three meta-attributes called name, which provides the attribute identification, defaultValue, which can be used to specify a default value for the attribute, and fixedValue, which can be used to specify a fixed value to the attribute. The meta-attributes defaultValue and fixedValue are optional and cannot be both present at the same time.

4 Transformation Specification

In the QVT specification [6], there is no clear distinction between the concepts of mapping and transformation. A transformation is defined in terms of mapping specifications, which can be specified using different languages defined in a two level declarative architecture, viz., relations, operational mappings and core. However, some authors distinguish between mapping and transformation (see, e.g., [4, 5]). According to [4], a mapping can be seen as a correspondence between the elements of two metamodels, while a transformation can be defined as the activity of transforming a source model into a target model according to a number of transformation definitions. The benefit of this approach is that it allows a mapping to be defined independently of the transformation specification.

In the context of this work, we adopt the approach mentioned in [4], in which mappings and transformations are explicitly separated. Therefore, we developed our transformation specification between the CWS and the XSD metamodel in two consecutive steps: 1) the definition of the mappings for each context-aware information element onto one or more XML schema elements, using the metamodel for mappings proposed in [4], and 2) the specification of the transformation itself, i.e., the specification of the mappings defined in the first step using the QVT notation [6].

In order to map context-aware information elements onto XML schema language elements, we have to first consider the different alternative schema definitions and make proper choices. In order to obtain these alternative schema definitions, we first instantiated some context-aware information from our CWS metamodel, then created an alternative schema definition for this sample information, and finally created one or more XML documents for each alternative schema definition.

```
XSD of the CWEntity PDA:
<xs:element name="PDA" type="CWClassifier"/>

<xs:complexType name="CWClassifier">
      <xs:attribute name="ClassifierInfo" type="xs:string"
                fixed="CWEntity"/>
</xs:complexType>


XSD of the CWAttribute DeviceType:
<xs:element name="DeviceType">
      <xs:complexType>
            <xs:attribute name="CWClassifier" type="xs:string"
                  fixed="CWAttribute"/>
            <xs:sequence>
                  <xs:element name="value" type="xs:string"/>
                  <xs:element name="timestamp"
                              type="xs:dateTime"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
```

(a)

```
XSD of the CWEntity PDA:
<xs:element name="PDA">
      <xs:complexType>
            <xs:attribute name="CWClassifier" type="xs:string"
                  fixed="CWEntity"/>
      </xs:complexType>
</xs:element>

XSD of the CWAttribute DeviceType:
<xs:element name="DeviceType">
      <xs:complexType>
            <xs:attribute name="CWClassifier" type="xs:string"
                  fixed="CWAttribute"/>
            <xs:element name="Model" type="Model_Content"/>
      </xs:complexType>
</xs:element>

<xs:complexType name="Model_Content">
      <xs:sequence>
            <xs:element name="value" type="xs:string"/>
            <xs:element name="timestamp" type="xs:dateTime"/>
      </xs:sequence>
</xs:complexType>
```

(b)

Fig. 3. Alternative schema definitions for context-aware information elements.

Fig. 3 illustrates two alternative schemas for some sample context-aware information elements. In the context of this work, we have chosen the structure presented in Fig. 3(b) because it is simple and allows the separation of the definition of CWAttribute and Content elements, which can be used for traceability purposes.

After determining the schema definition to be applied, we have defined the mapping from context-aware information onto XML schema elements. Fig. 4 shows the mappings between the two metamodel elements using the mapping metamodel proposed by [4].

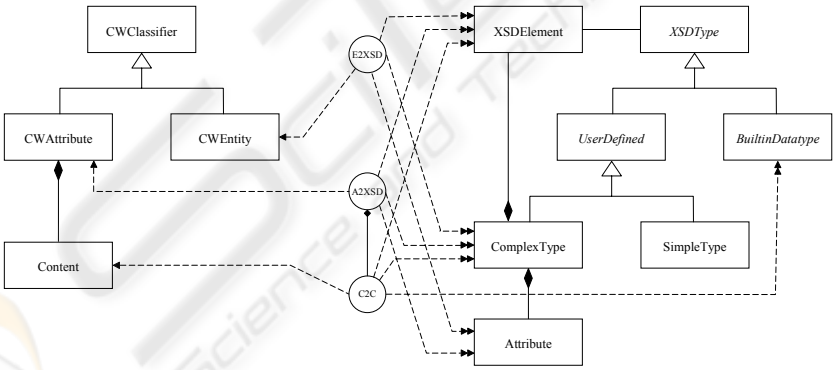

Fig. 4. Mapping from Context-aware Information elements onto XML Schema elements.

In Fig. 4, a circle represents a correspondence, i.e., a model element used to specify the relationships between two or more elements (left and right elements). Each relationship has one left element (represented by a single arrow) and one or more right elements (represented by a double arrow). A mapping definition contains all correspondences between two metamodels.

The specification of the transformation was carried out using the QVT Operational Mappings language. We have chosen this language because it offers a powerful

mechanism for specifying transformations using an imperative, procedural-like style. Fig. 5 shows some fragments of the CWS2XSD transformation specification.

```
modeltype CWS uses ContextAwareServiceMetamodel;
modeltype WSDL uses WSDLMetamodel;
transformation CWS2WSDL (in cwsModel: CWS out wsdlModel: WSDL);
main(){
  -- for each CWEntity create a corresponding XSDElement
  cwsModel.objectsOfType(CWEntity)->map E2XSD();
  -- for each CWAttribute create a corresponding XSDElement
  cwsModel.objectsOfType(CWAttribute)->map A2XSD();
}
constructor Attribute::Attribute(v: String){
  name := 'CWClassifier';
  type := new String();
  fixedValue := v;
}
mapping CWEntity::E2XSD(): result:XSDElement{…}
mapping CWAttribute::A2XSD(): result:XSDElement{
  object result:XSDElement{
    name := self.name;
    type := new ComplexType();
    type.attribute := new Attribute('CWAttribute');
    type.element := self.content->map C2XSD();
  }
}
mapping Content::C2XSD(): result:XSDElement{
  object result:XSDElement{
    name := self.name+'_Content';
    type := new ComplexType();
    type.orderIndicator := sequence;
    type.element := sequence{
       object v:XSDElement{
         -- mapping of value meta-attribute
         name := 'value';
         -- map type of value metaattribute
         type := if self.value.isKindOf(String) then
                    new String();  -- data type is String
                 elif …;
       };
       object t:XSDElement{
         -- mapping of timestamp meta-attribute
         name := 'timestamp';
         type := new DateTime();
       }
    }
  }
}
```

Fig. 5. CWS to XSD transformation specification.

The main clause of the *CWS2XSD* transformation specification defines that for each CWEntity and CWAttribute elements a corresponding XSD element should be created through the application of the *CWEntity to XSDElement* (E2XSD) and *CWAttribute to XSDElement* (A2XSD) mappings, respectively. Both mappings use a constructor called Attribute that creates an Attribute element with a provided fixedValue.

The A2XSD mapping uses the *Content to XSDElement* (C2XSD) mapping to create a corresponding XSD element for each Content element present in the CWS model. The C2XSD mapping is straightforward. This mapping basically creates an

XSDElement and an associated ComplexType element. This ComplexType element basically contains two XSD elements with associated built-in datatypes.

5 Transformation Application

In order to illustrate the proposed transformation specification, we have developed a simple example involving a simple context-aware information model. Fig. 6 illustrates our example information model. The proposed example contains a CWEntity called PDA and two associated CWAttributes, namely DeviceId and DeviceType. DeviceId has a Content element named Code, while DeviceType has a Content element named Model. The association between PDA and DeviceId is captured through the identifiedBy profiled CWAssociation, while the association between PDA and DeviceType is captured through a hasType static CWAssociation. Since the relationships between CWEntity and CWAttribute are not relevant for the transformation specification, we exempt ourselves from discussing it further in this work. More information on these relationships can be found in [2].

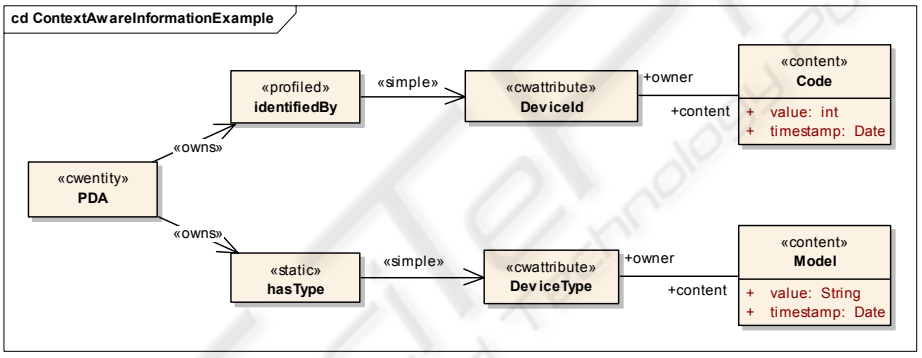

Fig. 6. Sample context-aware information model.

We have applied the proposed transformation rules to the information model above and obtained an XML schema model as result. Fig. 7 depicts a fragment of the obtained XML schema model. This fragment depicts the result of the mapping applied onto the DeviceType CWAttribute and associated Model Content element. The model fragment illustrated in Fig. 7 can be serialized in order to obtain the XML schema definition represented in Fig. 3b, while Fig. 8 shows a sample XML document that can be validated by the XML schema definition represented Fig. 3b.

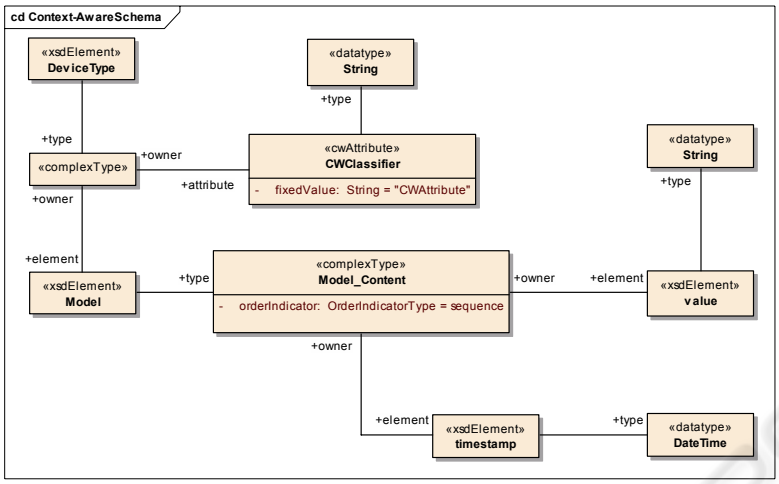

Fig. 7. XML schema model for proposed information model.

```
<DeviceType CWClassifier="CWAttribute"
        <Model_Content>
                <value>HP iPAQ hx2190</value>
                <timestamp>2007-02-19T21:30:00Z</timestamp>
        </Model_Content>
</DeviceType>
```

Fig. 8. XML document.

6 Conclusion

Model transformation is a key aspect in the model-driven development of software applications. In general, transformations enable a development approach in which high level models in a source domain are (automatically) transformed into more concrete models in a target domain. Thus, the QVT specification is at the core of the OMG´s MDA initiative, since this specification defines how transformations can be defined in a two-level declarative architecture.

This paper presents a case study on the use of QVT to define the transformation between a (source) context-aware information metamodel and a (target) XML Schema Definition metamodel. We also describe a number of steps that can be generally used to accelerate the development XML Schema Definitions from some corresponding domain-specific information metamodel.

The definition of a transformation specification is a complex task involving knowledge of both the source and target domains. This work has reinforced the notion proposed by [4, 5] that the separation between mappings and transformation specification helps structure this complexity. Additionally, tool support should be available not only for the definition of source and target metamodels, but also for the definition and execution of transformation specifications.

The work described in this paper is part of a larger initiative that aims at providing a model-driven approach for the development of context-aware applications. In this sense, we intend to work on the transformation specification of service signatures from our context-aware service metamodel onto the WSDL metamodel. Additional research is also needed to define a metamodel for service behaviour specification, before being able to define how this service behaviour metamodel should be transformed to some service implementation metamodel.

Acknowledgements

This work has been supported by the Brazilian National Council for Scientific and Technological Development (CNPq), under project number 50.6284/2004-2, and by the Freeband A-MUSE project (http://a-muse.freeband.nl), which is sponsored by the Dutch government under contract BSIK 03025.

References

1. de Farias, C. R. G., Ferreira Pires, L., van Sinderen, M.: A MOF Metamodel for the Web Service Description Language (WSDL). Project Deliverable, DBMWare/CNPq/D2.1, v.1.0, 2007.
2. de Farias, C. R. G., Medina Leite, M., Calvi, C. Z., Pessoa, R. M., Pereira Filho, J. G.: A MOF Metamodel for the Development of Context-Aware Mobile Applications. In *Proceedings of the 22nd ACM Symposium on Applied Computing (SAC´07)*, (2007) 947-952.
3. Dockhorn Costa, P., Ferreira Pires, L., van Sinderen, M.J.: Designing a Configurable Services Platform for Mobile Context-Aware Applications. *International Journal of Pervasive Computing and Communications*, 1 (1) (2005) 13-25. ISSN 1742-7371
4. Lopes, D., Hammoudi, S., Bézivin, J., Jouault, F.: Generating Transformation Definition from Mapping Specification: Application to Web Service Platform. In *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05)*, LNCS 3520 (2005) 309-325.
5. Lopes, D., Hammoudi, S., Bézivin, J., Jouault, F.: Mapping Specification in MDA: From Theory to Practice. In: Konstantas, D., Bourrières, J.-P., Léonard, M., Boudjlida, N. (Eds). *Interoperability of Enterprise Software and Applications - INTEROP-ESA*. Springer, (2006) 253-264.
6. OMG: Meta *Object Facility (MOF) 2.0 Query/View/Transformation Specification*, OMG Adopted Specification. Object Management Group (2005).
7. OMG: *Meta Object Facility (MOF) Core Specification*, OMG Available Specification, version 2.0. Object Management Group (2006).
8. W3C: Simple Object Access Protocol (SOAP) 1.1 (2000).
9. W3C: Web Services Description Language (WSDL) 1.1 (2001).
10. W3C: XML Schema Part 0: Primer, Second Edition (2004).