DESIGNING XML PIVOT MODELS FOR MASTER DATA INTEGRATION VIA UML PROFILE

Ludovic Menet, Myriam Lamolle

DESIGNING XML PIVOT MODELS FOR MASTER DATA INTEGRATION VIA UML PROFILE

Ludovic Menet, Myriam Lamolle
Laboratoire d'INformatique et Communication (LINC)
IUT de Montreuil, Université Paris8
140 rue de la Nouvelle France, F-93100 Montreuil
{m.lamolle, l.menet}@iut.univ-paris8.fr
Ludovic.menet@orchestranetworks.com

Keywords: Interoperability, Master Data Management, Metamodel, XML Metaschema, UML profile, XML Schema.

Abstract: The majority of Information Systems is concerned by heterogeneity in both data and solutions. The use of this data thus becomes complex, inefficient and expensive in business applications. The issues of data integration, data storage, design and exchange of models are strongly linked. The need to federate data sources and to use standard modelling formalism is apparent. In this paper, we propose a mediation solution based on XML architecture. The integration of the heterogeneous data sources is done by defining a pivot model. This model uses the standard XML Schema allowing the definition of complex data structures. We introduce features of the UML formalism, through a profile, to facilitate the collaborative definition and the exchange of these models.

1 INTRODUCTION

The evolution of computer networks and data management systems has led to the advent of large-scale information systems inside companies. These systems increasingly using the Web for sharing information are characterised by geographically distributed and heterogeneous data sources. Consequently, the information management becomes complex, inefficient, insecure and expensive. Most research into heterogeneous data integration has opted for solutions based on mediation architectures such as (Abiteboul, 2002), (Garcia-Molina, 1995), (Lamolle, 2005). These mediation architectures are based on the definition of a data pivot model federating several heterogeneous data sources. The definition of this model is carried out by a semi-automatic process requiring human intervention, or is done by an expert in the field. However, all existing solutions use various tools, technologies and formalisms to define models. In this paper, we describe a means to facilitate the definition of models for data integration using standards components and formalisms. Our mediation solution is based on the standard XML recommended by the W3C (W3C, 2004). The standardisation of the XML language and its easy use has made it a suitable technology for data system integration. In our architecture, the pivot model is XML Schema document allowing complex, structured, typed and rich data structures that we call *adaptation models* to be defined. The definition of an adaptation model is based on the XML Schema technology in keeping with the standard defined by the W3C. The use of XML is appropriate to the definition of models but involves an extensive knowledge of this language, so various people from several teams can participate in the process. We propose using UML as a common formalism in order to make the definition, understanding and exchange of a model easier.

2 DATA INTEGRATION AND MANAGEMENT

In an Information System, data are represented and persisted in heterogeneous data sources. The first approaches to integration have been done within the framework of relational, object/relational or object database systems across a platform of database federation. The essential need is to be able to

interrogate several data sources at a same time, and to give the user the impression that is interrogating a single source. The virtualised approach and the materialised approach imply two different solutions.

2.1 Main data integration approaches

2.1.1 Virtualised approach

The virtual (or mediator) approach represents a global vision by the means of a single schema representing all the heterogeneous data sources. This schema can be automatically defined with tools or schema extractors (Lamolle, 2005). Some existing projects focused on this approach. The *TSIMMIS* project (Garcia-Molina, 2005), developed by the university of Stanford, is based on this approach. One of the purposes of *TSIMMIS* is to integrate heterogeneous sources which may be no well structured and changing. In this context, we can also quote other projects such as *DISCO* (Tomasic, 1997) and *YAT* (Siméon, 2000). In which, user requests are formulated with the semantic of the global schema. The execution of these requests requires translation into sub-requests understandable by all the sub-schemas of the various data sources. Figure 1 illustrates an architecture based on this approach.

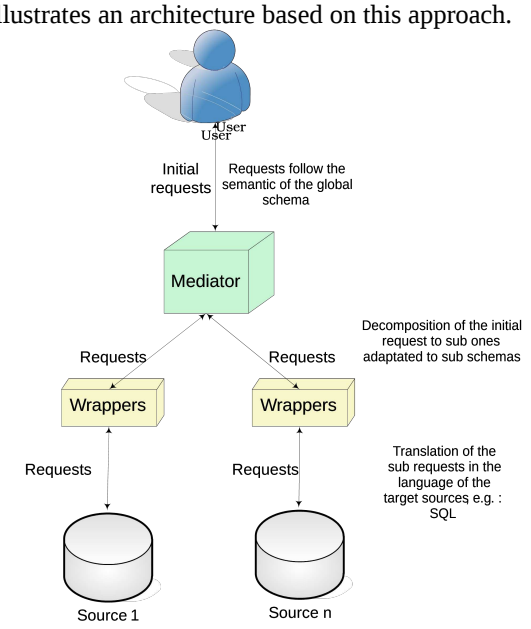

Figure 1: Architecture based on a virtualised approach.

In this case, the data are stored in the data source (cf. *source1*, ..., *source n* in figure 1), and all the operations are synchronised with them. The *mediator* knows the global schema and handles abstract views on sources. Through these views, the mediator can decompose the initial request into sub-requests. The mediator submits these sub-requests to wrappers which translate them into the language of data sources.

2.1.2 Materialised approach

In the materialised approach, the data from the heterogeneous sources are stored in a data warehouse (or repository). The *Xylème* project (Abiteboul, 2002) is a dynamic data warehouse focused on the persistence and the semi-automatic integration of all the Web XML resources. The end user thus has single and transparent access to the heterogeneous data sources. The use of a system based on trees (Delobel, 2003) makes Xylème an efficient system for requests evaluation and data integration and management. Other works (Baril, 2003) (Gofarelli, 2001) (Nassis, 2004) focus on an XML architecture and specifically for the integration of XML documents. This approach is focused on a copy of the data in a data warehouse. As a result, all the actions in the repository are asynchronous with the data sources. The propagation of the modifications from the repository to the data sources is performed with updating processes. Unlike with the virtualised approach, user requests are directly executed in the repository without having to access data sources. Updates from the repository to data sources, and vice versa, are delegated to an *integrator* which carries out the matching between the repository schema and the data sources sub-schemas. The user requests are submitted to the integrator which breaks them down into sub-requests. These sub-requests are delegated to *wrappers* which translate them into the language of data sources. Figure 2 illustrates the architecture of a system based on a materialised approach.

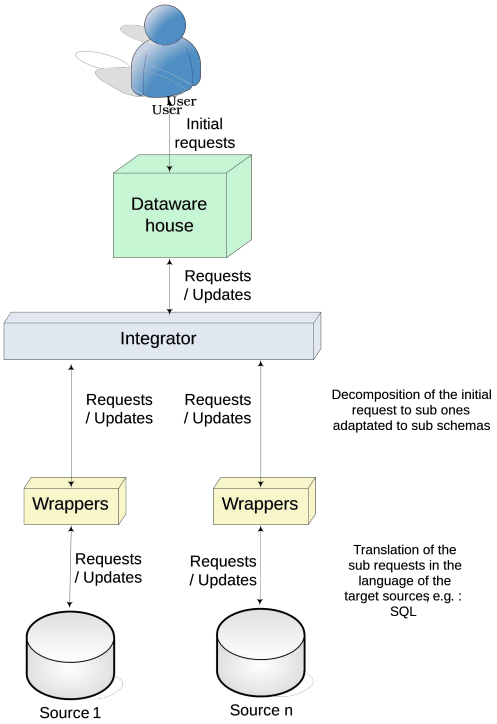

Figure 2: Architecture based on a materialised approach.

The virtualised and the materialised approaches are two solutions for data integration. Our research falls within materialised approach.

2.2 EBX.Platform

We propose a Master Data Management solution called EBX.Platform. Based on Java and XML Schema, EBX.Platform is a standard and non-intrusive solution that helps companies to unify and manage their reference business data and parameters across their Information System.

2.2.1 Master Data Management (MDM)

The Master Data Management is a means of reference data unification, management and integration across the company Information System. These data can be of various kinds (products, services, offers, prices, customers, providers, lawful data, financial data, organisations, structures, people, etc.). Currently, the majority of Information Systems is concerned by heterogeneity in both data and solutions. In this context, there are three kinds of heterogeneity:

- storage systems heterogeneity (databases, directories, files, ...).
- data format heterogeneity (owner files, XML documents, tables, ...).
- solutions heterogeneity for different types of data management.

The management of the data thus becomes complex, insecure, inefficient and expensive. Moreover, using different applications to manage this diversity involves some redundancy in both data and tools.

An Information System without Master Data Management (MDM) raises issues such as:

- lack of unified vision of the reference data.
- data duplication in several systems.
- lack of coherence between companies and subsidiaries.
- lack of single tool for users.

EBX.Platform is a MDM solution based on powerful concepts to solve these problems.

2.2.2 EBX.Platform's concepts

EBX.Platform is based on two concepts: **(i) an adaptation model** (an XML Schema document) which is a data model for a set of Master Data and **(ii) an adaptation** which is an XML instance of the adaptation model containing Master Data Values. Using XML Schema allows each node of the data model corresponding to an existing data type in keeping with the W3C standard to be specified. EBX.Platform supports the main XML Schema datatypes as well as multi-occurrence complex types. Indeed, the XML Schema formalism allows us to define constraints (enumeration, length, lower and higher limits, etc.), information about adaptation and its instances (access connector, Java factory class, access restriction, etc.) and layout information (label, description, formatting, etc.) to be specified for each node of the schema. For each declared possible instances node of the adaptation model corresponds a node in the adaptation. If an adaptation model has several adaptations, we consider that an adaptation tree is used. Figure 3 shows an adaptation model and its instances as an adaptation tree.

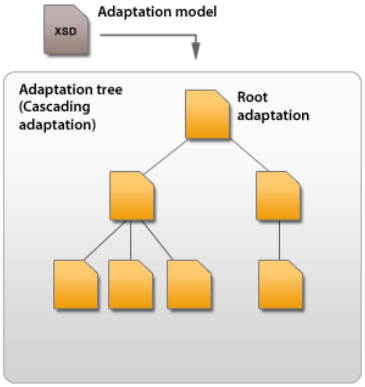

Figure 3: An adaptation model and its instances.

In an adaptation, each node has the following properties:

- **An adaptation value**: if this value is not defined in the current adaptation then it is inherited recursively from its ancestor (parent adaptation). If no ancestor defines a value, then the value is inherited, by default, from the data model.
- **An access right** for descendants: the adaptation node can be either hidden, in read only or in read/write mode (for the descendants).

These powerful concepts are deployed on a specific architecture made up of several components. Figure 4 presents the EBX.Platform architecture.

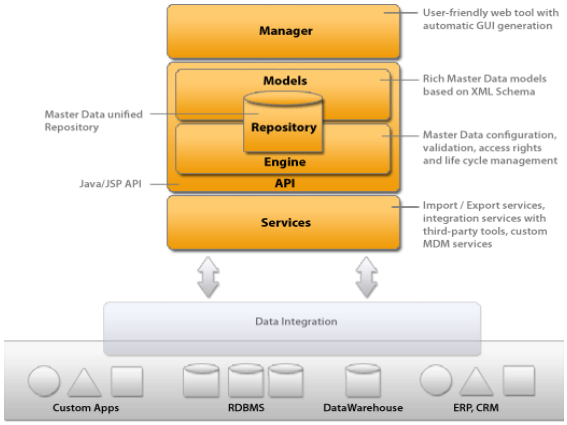

Figure 4: EBX.Platform architecture.

The architecture shown in figure 4 is based on three important components:

- EBX.Platform Engine is based on a technology that allows multiple instances of Master Data in a core repository to be managed. EBX.Platform Engine main features are data validation, data configuration, life cycle management and access rights management.
- EBX.Platform, with EBX.Manager, provides both business and technical users with a Web-based tool for Master Data Management. EBX.Manager dynamically generates a rich User Interface from Master Data models without any programming.
- EBX.Platform services allows Master Data into Information Systems to be integrated providing import/export features and integration with third party tools, such as EAI, ETL, ESB, directories. Custom MDM services can be developed using a standard Java API. Indeed, using our Java API, it is possible to integrate new features in EBX.Platform. For example, services can be used to perform reporting, data versioning, process management, etc

The problems associated with the existence of multiple applications across an Information System can be resolved using a MDM solution (lamolle, 2007). This solution is based on the definition of a model federating heterogeneous data sources. In our solution, the model is an XML Schema document generally defined by experts in the field. However, these experts may not have the same technical skills, and the model has to be understood by each of them. In this way, we introduce a modelling standard as a common formalism of model design.

3 MODELS DESIGN AND EXCHANGE VIA UML

The data models complexity evolves with the needs of the given field. These models are becoming more difficult to define and design due to the diversity of existing technologies. In order to resolve this issue, a standard notation and a standard design method has to be used. We should keep in mind that the quality of a model is based on three criteria:

- *Expressivity* to exploit the information contained by data,
- *Powerful* in order to meet the needs of the field,
- *Abstraction*, in order to be accessible by specialists from different fields.

The last point is the issue that we want to address using the UML formalism (Pilone, 2006).

3.1 Master Data Modelling profile (MDM-p)

In addition to its modelling abilities, UML allows profiles to be defined. A profile specialises the UML formalism for an application field or a particular technology. Many profiles have been developed for several goals : for expressing the whole semantic of CORBA (OMG, 2002), a profile defining the features of EJB (Greenfield, 2001), the Turtle-P profile for the formal validation of critical and distributed systems (Apvrille, 2006), or the Omega project which defines a profile for the modelling and the validation of real-time systems (Ober, 2005). An UML profile can be used for giving syntax to concepts that have none, giving a different notation to existing symbols, or adding semantic to existing concepts. We define MDM-p in three steps. First, we proceed to the definition of the domain of the profile, i.e. the metamodel defining the concepts and the relations between them. Secondly, we carry out the technical definition of the profile which establishes the matching between UML concepts and those defined in the profile. Thirdly, we define the profile extending the standard UML class diagram.

3.1.1 Domain definition of MDM-p

The domain of our profile is defined by a metamodel. The MOF (Meta Object Facility) has been defined by the OMG (OMG, 2003) as a standard for defining metamodels. The MOF metamodel framework is depicted as a four layer architecture. Our metamodel has been defined to meet to this standard as shown in figure 6.

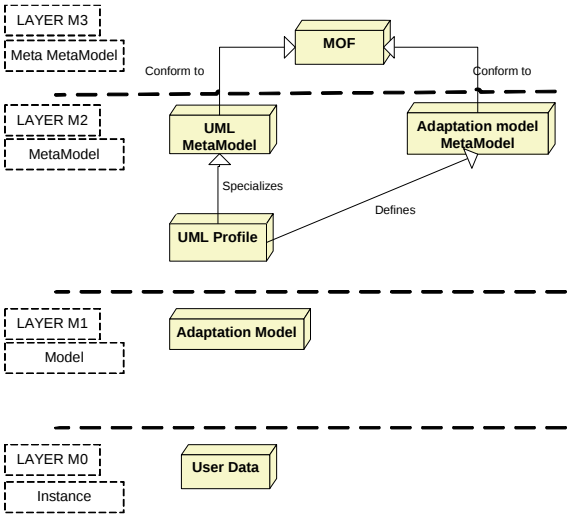

Figure 6: OMG Metamodel architecture.

3.1.2 Technical definition of MDM-p

Table 1 shows part of the technical definition of our profile.

Stereotype	*Applied to*	*Description*
AdaptationModel	Class	Defines an adaptation model.
Root	Class	Defines the root of an adaptation model.
Sequence	Class	Defines a XML Schema complex element of type <xs:sequence>.
SimpleType	Class	Defines an XML Schema redefined type.
Table	Class	Defines a table element.

Table 1: Extract of the technical definition of UML profile representing EBX.Platform metamodel

The technical definition is used to realise the UML profile of the adaptation models.

3.1.3 Definition of MDM-p

Our profile extends the UML class diagram for the description and the structure of the adaptation models. Figure 7 presents part of the UML profile

defining the relations between the different concepts introduced by EBX.Platform.

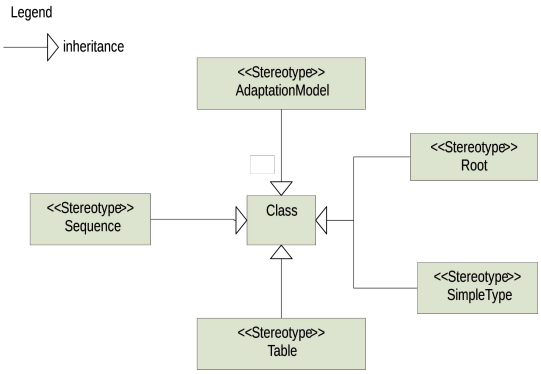

Figure 7: Part of the UML profile representing EBX.Platform metamodel.

Using UML extension mechanism enables us to extend the UML formalism to our semantic. This extension is done using *stereotypes* and *marked values*. Stereotypes are employed to define a new type of element from an existing one. We can see on the figure 7 that the stereotypes (labelled <<Stereotypes>>) inherit from the element *Class* of the UML metamodel. As a result, the stereotypes will be instantiated from the metamodel constructor in the same way as the element *Class*. Tagged values specify keyword-value pairs of model elements to set properties for existing elements or for stereotypes. The definition of these stereotypes allows more semantics to be introduced, out of the concepts defined in UML which will enable us to define an adaptation model with an UML diagram.

3.1.4 Application of MDM-p

This section illustrates part of an adaptation model defined with MDM-p and its equivalent in XML Schema. Figure 9 represents the UML modeling of an adaptation model defining a simplified train network. This model is composed of concepts and relations between them In our example, we have defined the concept *Train* as composed of an *engine*, *wheels* (notion of composition), being able to have *wagons* (notion of aggregation) and having properties such as a *type* and a *trademark*. We associate a driver of type *Person* with a train. The concept *Person* defines properties such as *firstname*, *lastname* and *birthday*. We have also defined the property *email* representing the use of a redefined type. The class *Email* uses the stereotype *<<SimpleType>>* indicating that it is a redefined type in the XML Schema sense. The properties of this redefined type are expressed in an UML annotation specifying values for *SimpleType::base*, *SimpleType::pattern* and *SimpleType::whitespac*e. The root of the schema is specified by the stereotype *<<Root>>*, applied to the class *NetworkTrain*. The stereotype *<<AdaptationModel>>*, applied to the class *AdaptationModelRoot*, indicates the definition of an adaptation model. The classes defined in this model (excepted for the class *AdaptationModelRoot*) are also stereotyped *<<Sequence>>*. This stereotype specifies that the corresponding class represents a complex element of the type <xs:sequence> (in the XML Schema sense).

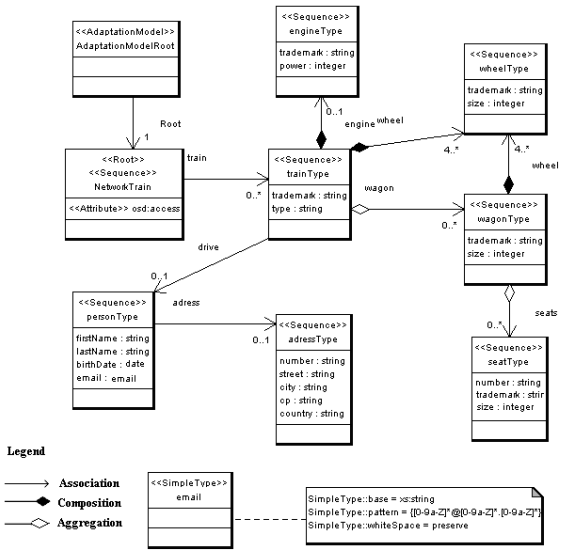

Figure 9: Definition of an adaptation model with UML.

The program code generation of a part of the corresponding adaptation model as defined in XML Schema is shown below:

```
<!-- Start of  the adaptation model -->
<xs:schema xmlns:fmt="urn:ebx-
schemas:format_1.0"
xmlns:xs="http://www.w3.org/2001/XMLSch
ema">
<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
<!--Root of the adaptation model  -->
<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
<xs:element name="NetworkTrain"
osd:access="--">
<xs:complexType>
<xs:sequence>
<xs:element name="train"
type="trainType" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
```

```
</xs:element>
<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!--Class : trainType  -->
<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!--trainType -->
<xs:complexType name="trainType">
<xs:sequence>
<xs:element name="trademark"
type="xs:string"/>
<xs:element name="type"
type="xs:string"/>
<xs:element name="wagon"
type="wagonType" maxOccurs="unbounded">
     <xs:annotation>
          <xs:appinfo>
               <osd:aggregation/>
          </xs:appinfo>
     </xs:annotation>
</xs:element>
<xs:element name="engine"
type="engineType">
     <xs:annotation>
          <xs:appinfo>
               <osd:composition/>
          </xs:appinfo>
     </xs:annotation>
</xs:element>
<xs:element name="wheel"
type="wheelType" minOccurs="4"
maxOccurs="unbounded">
     <xs:annotation>
          <xs:appinfo>
               <osd:composition/>
          </xs:appinfo>
     </xs:annotation>
</xs:element>
<xs:element name="drive"
type="personType"/>
</xs:sequence>
</xs:complexType>
<!--End train -->
…
```

3.2 Exchanging models

The need to communicate and to exchange is the result of various evolutions in the model development process. Indeed, the increasing size and complexity of models, the heavy use of networks and the intervention of different specialists in the field in the design imply communication and exchange. Previously we saw that we can facilitate communication between people using a common formalism, in our case using UML as an abstraction of any technical language. The heavy use of networks shows that we have to find a standard and normalised way to exchange models across networks. The W3C has introduced a standard called XML Metadata Interchange (XMI). XMI is a model of data exchange allowing different people to exchange data via the web. In this context, our problem is being able to share models over the internet and possibly for different applications. XMI plays an important role in bringing consistency and compatibility to models created in different applications. Figure 10 illustrates the process of model exchange between two people involved in the model design.

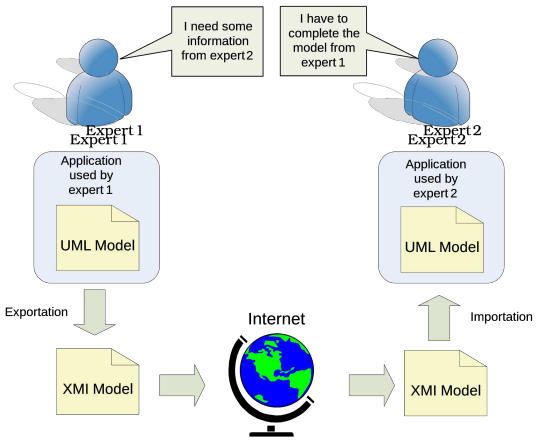

Figure 10: Exchange of model.

In our solution, we represent a data model in UML. UML is a graphical formalism and cannot be exchanged via the web in this state. XMI formalism is perfectly adapted for our needs. Indeed XMI is in keeping with the XML standard which is commonly used for sharing data over the web. Our solution can transform an UML model into an XMI model which can be exchanged between different users and applications. The exchange of models by applications is subject to constraints. Some applications can be more technical than others, the information has to be adapted to the application and to the user. This adaptation is performed by a mapping. To perform the mapping of one XML document to another, the XSL language has been standardised by the W3C. XSL allows rules to defined. A selector and a target form compose a rule. The selector specifies the element in the source document which has to be transformed into the target form. Using XSL, it is possible to carry out transformations in order to extract the information adapted to the different actors involved in the design of a model, and then to exchange data models independently of the application used.

4 CONCLUSION

UML is a powerful and flexible modelling language and XML became a standard for data interchange on the Web. The use of these two technologies addresses interoperability and a standard means of exchanging and defining models. In this paper, we have shown how to integrate, design and exchange data using standard technologies. We propose a generic solution for data integration based on the XML technology. Indeed, our solution can integrate data from several kinds of sources such as databases or XML documents. We have seen that every adaptation model is an XML Schema standard document. Some XML Schema features are difficult to handle for the designer of models. Therefore, we have shown how UML profiles facilitate the definition of an adaptation model and the communication between different actors involved in the design of models. However, the needs of improvement to the import and export of XMI documents. Indeed, our experience has shown that the importa and the export of a XMI document can differ considerably from one application to another. This is a consequence of the different existing XMI versions. In addition, some applications bring additional information used by a custom graphical representation.

Our follow-up work will consist of the enrichment of MDM-p and the creation of a complete data modelling language for the Master Data Management module in the following two ways: (i) a graphical language using UML modelling, (ii) a specialised language using UML meta-modelling. This language will take into account the definition of constraints and the validation of models using OCL (Object Constraint Language).

ACKNOWLEDGEMENTS

We would like to thank Orchestra Networks for their support for our research.

REFERENCES

Abiteboul S., Cluet S., Ferran G., Rousset M.-C., 2002. The Xyleme Project. Computer Networks 39.

Apvrille L., De Saqui-Sannes P., Khendek F., 2006. TURTLE-P: A UML Profile for the Formal Validation of critical and Distributed Systems. SoSym (Software and System Modeling) Journal, Springer, ISSN 1619-1366, DOI: 10.1007/s10270-006-0029-5, Pages: 1-18, July.

Baril X., Bellahsène Z., 2003. XML Data Management: Native XML and XML-Enabled Database Systems. Chapter Designing and Managing an XML Warehouse, pp.455–474. Addison Wesley Professional.

Delobel C., Reynaud C., Rousset M.C., Sirot J.P., Vodislav D., 2003. Semantic integration in Xyleme : A uniform tree-based approach. Data and Knowledge Engineering 44. pp267-298.

Garcia-Molina H., Papakonstantinou Y., Quass D., Rajaraman A., Sagiv Y., Ullman J., Widom J., 1995. The STIMMIS approach to mediation: Data Models and Languages. NGITS (Next Generation Information Technologies and Systems), Naharia, Isreal, June 27-29.

Gofarelli M., Rizzi S., Vrdoljak B., 2001. Data Warehouse Design from XML Sources, In Proc. The 4th ACM Intl Workshop on Data Warehousing and OLAP (DOLAP01), pp. 4047, Atlanta.

Greenfield J., 2001. UML Profile For EJB. Rational Software Corp, May.

Lamolle, M., and Zerdazi, A., 2005. Intégration de Bases de données hétérogènes par une modélisation conceptuelle XML, In *Conférence sur l'Optimisation et les Systèmes d'Information (COSI'05)*, 216-227.

Lamolle M., Menet L., 2007, Meta-Modelling "object": expresssion of semantic constraints constraints in complex data structure, in proceedings of ERIMA'07, pp.104–107, Biarritz, France

Nassis V., 2004. Conceptual Design of XML Document Warehouses, In Proc. Data Warehousing and Knowledge Discovery, 6th International Conference, DaWaK 2004, pp. 114, Zaragoza, Spain.

Ober J., Ober I., Graf S., Lesens D., 2005. Projet OMEGA : Un profil UML et un outil pour la modélisation et la validation de systèmes temps réel. Génie logiciel ISSN 1265-1397. Congrès Ingénierie dirigée pas les modèles, vérification de modèles Journée de travail NEPTUNE N°2, Paris, FRANCE , n° 73 (9 ref.), pp. 33-38.

OMG/MOF, 2000. Meta Object Facility (MOF) Specification, OMG Document formal/2000-04-03, mars. http://www.omg.org/technology/documents/formal/mof.htm.

OMG, 2002. "CORBA specifications". http://www.omg.org/technology/documents/formal/profile_corba.htm.

Pilone D., Pitman N., 2006. *UML 2.0 in a Nutshell*, O'Reilly.

Siméon, 2000. Data Integration with XML : A Solution for Modern Web Applications. Lecture at Temple University, March.

Tomasic A., Louiqua Rashid, Patrick Valduriez, 1997. A data model and query processing techniques for scaling access to distributed heterogeneous databases in Disco . IEEE Transactions on computers, special issues on Distributed Computing Systems.