A Model Driven Engineering Approach Applied to Master Data Management

Ludovic Menet[1,2] and Myriam Lamolle[1]

[1] Laboratoire d'Informatique Avancée de Saint Denis (LIASD)
IUT of Montreuil, University of Paris 8,
140 rue de la nouvelle France, 93100 Montreuil, France
{l.menet,m.lamolle}@iut.univ-paris8.fr
[2] Orchestra Networks, R&D department,
75 boulevard Haussmann, 75008 Paris, France
ludovic.menet@orchestranetworks.com

Abstract. The federation of data sources and the definition of pivot models are strongly interrelated topics. This paper explores a mediation solution based on XML architecture and the concept of Master Data Management. In this solution, pivot models use the standard XML Schema allowing the definition of complex data structures. The introduction of a MDE approach is a means to make modeling easier. We use UML as an abstract modeling layer. UML is a modeling object language, which is more and more used and recognized as a standard in the software engineering field, which makes it an ideal candidate for the modeling of XML Schema models. In this purpose we introduce features of the UML formalism, through profiles, to facilitate the definition and the exchange of models.

Keywords: MDE, MDA, Master Data Management, Metamodel, UML, XML Schema.

1 Introduction

The evolution of networks and of systems of data management led to the rise of wide scale Information Systems within companies. These systems using increasingly the Web to share and propagate information are characterized by data sources of very different kinds. Indeed, these sources can be distributed, heterogeneous and autonomous. Consequently, information management becomes complex, inefficient, costly and of uncertain quality. Concerning the heterogeneity of data sources, three needs appear: (i) to integrate data to unify the different sources, (ii) to use a unified data model federating the different models associated to data sources, (iii) to minimize the number of data management tools in order to take advantage of them efficiently.

Master Data Management (MDM) is an emerging discipline focusing on these three points. MDM aims at integrating data split in multiple systems by defining a master repository formatted as a data warehouse. This master repository centralizes data structures, thanks to the data warehouse model, the contents as well as the

R. Meersman, P. Herrero, and T. Dillon (Eds.): OTM 2009 Workshops, LNCS 5872, pp. 19–28, 2009.

implementation of management tools *via* a unique application, thus ensuring the data lasting application systems quality. Our MDM approach is based on XML standard recommended by the W3C [15], since the standardization of the XML language made it a suitable technology for data integration systems. In our architecture, the unified data model (or pivot) that we call *"adaptation model"*, is an XML Schema document [16] allowing the definition of complex, structured, typed and rich models. However, even though the use of XML is suitable for the definition of models, it requires a thorough knowledge of this language by the different actors involved in the definition process of the pivot data model. This problematic led us to introduce a thought process to guide data model designers, in order for them to concentrate solely on data modeling and integration rather than on the technology to use. The selection of an object and standard approach to improve the model understanding and associated semantics (MDM semantics in our perspectives), seems to be a simple and efficient way. Therefore, the principal objective of our works is to follow a Model Driven Engineering (MDE) approach to ignore the technological layer (physical) for the benefit of the functional layer (logical). The functional aspect is delegated to UML [14] and completed by UML profiles to define specific semantics to our MDM and XML research domains. The change-over from the functional solution to the technological solution is ensured by model transformation processes based on standard formalisms such as Meta Object Facility [11] and XMI [8]. In this paper, we present an application of the MDE concepts to the MDM domain dedicated to data integration. The continuation of this article is organized as follows: section 2 presents the data integration approach by Master Data Management; section 3 details our XML architecture for data integration; section 4 presents the introduction of an MDE approach applied to the MDM domain.

2 Data Integration by Master Data Management

In the context of interoperability of heterogeneous data sources, two principal approaches to data integration exist, namely the virtual approach (or by mediator) [5], and the materialized approach (or by warehouse) [2]. Master Data Management is an emerging method for data integration based on the materialized approach. As a recent discipline, very few works exist on MDM to date (iWays Master Data center, Oracle MDM suite, IBM MDM, OrchestraNetworks MDM). MDM was defined as a method focused on data integration and centralization, models and tools within an Information System. Currently, the majority of Information Systems is characterized by heterogeneity in terms of data and settings solutions. Indeed, this heterogeneity is present within different aspects: diversity of storage systems (databases, files, directories, etc.), of data formats (tables, owner files, XML documents, etc.), of solutions offered for managing different data types, of actors taking advantage of the reference data (functional users or not), of application domains (CRM[1], ERP[2], etc.), of activities (called "vertical" for activities such as production or supplying, or "horizontal" for activities such as marketing or human resources), etc.

[1] Customer Relationship Client.
[2] Enterprise Resource Planning.

This heterogeneity in the data, in the existing solutions on the market and in the application domains, results in making the implementation and exploitation of these data heavy, complex and costly by the applications of the company. Using a set of different applications to manage this diversity in the types of data, inevitably leads to redundancy both at the data and tools level . In the absence of MDM, the propagation of data updates is carried out without any central repository or joint information model, with an architecture style often qualified as “peer to peer”. This style is relatively common in ETL[3]/EAI[4] base architecture, without MDM.

Two conditions are necessary for a centralized MDM architecture: (i) to have a generic MDM tool capable of hosting the joint information model for all kinds of data. Without this genericity level, MDM should be accepted as silos organized around information domains (Client, Product, Organization, functional settings, technical settings, etc.), harmful consequences in terms of duplication of repositories (which is what we seek to avoid) and of duplication of governance functions (versions management, human-to-administration machine interface, etc.). (ii) A method for modeling and negotiating the joint information model is needed which would ignore the “owners” formats of the different systems. The first point is ensured by our MDM solution that we present in section 3. The introduction of an IDM approach (section 4) to define the pivot data models is a solution possible for the second point.

3 EBX.Platform as a MDM Solution

Based on the XML Schema standard, EBX.Platform[5] simplifies the definition of models aimed at unifying the reference data of a company. Using the XML Schema technology, these models can be of any type (simple, complex) and any nature (business, technical, graphical). One of the main advantages of XML Schema is to allow the definition of structured and typed data models, with powerful validation properties. Compared to existing integration systems [2] [17] our solution also deals with important aspects of data federation: *data life cycle* handling multiple "branches" in a repository and making possible to perform concurrent changes on a repository and to compare/merge them; *data inheritance* to avoid data duplication between instances; *access rights* based on a directory (internal or external - for example LDAP), EBX.Platform allows to define access profiles to the MDM and to configure rights on each action, each object or even each attribute; *data quality* using a powerful incremental validation framework ; a unique Web-based tool which dynamically generates the Graphical User Interface from Master Data models. It means that once a Master Data model has been designed, users are able to create instances, edit values and validate data content in a single application.

3.1 Concepts

EBX.Platform is based on two principles namely: (i) *adaptation models* that are XML Schema documents defining the structure of reference data, and (ii) *adaptations* that

[3] Extraction Transformation Loading.
[4] Enterprise Application Integration.
[5] Online documentation available at http://doc.orchestranetworks.com/

are XML instances of adaptation models representing the content of reference data. The use of XML Schema enables us to specify that each node of the data model corresponds to an existing type of data and conforms to the W3C standard. On the other hand, the formalism of XML Schema allows constraints (enumerations, length, inferior and superior bounds, etc.), An adaptation is an instance of the adaptation model. For any node of the adaptation model declared as recoverable corresponds a node in the adaptation. It is often found that more than three quarters of reference data are identical between two instances (for example a products catalog between a head office and a subsidiary). It is important to avoid the duplication of these data in order to prevent long input procedures that are often tedious and sources of errors. To do so, EBX.Platform relies on an inheritance technology. In this management model, each instance inherits from its parent. When an adaptation model owns several adaptations, we consider that an adaptation tree is handled.

3.2 Adaptation Model Example

This section describes how to build an adaptation model starting from a sample relational database called publications, containing data that represent a fictitious publishing business. This database contains the following tables: *Publishers* table contains the identification numbers, names, cities, and states of publishing companies ; *Authors* table contains an identification number, first and last name, address information, and contract status for each author ; *Titles* table contains the book ID, name, type, publisher ID, price, advance, royalty, year-to-date sales, comments, and publication date for each book ; *Royalties* table lists the unit sales ranges and the royalty connected with each range. The royalty is some percentage of the total sales.

Figure 1 presents the XML Schema structure, corresponding to the Publisher table of the publications database.

```
<xs:complexType name="Publisher">
  <xs:annotation><xs:appinfo>
       <osd:table>
         <primaryKeys>/pub_id</primaryKeys>
       </osd:table>
    </xs:appinfo></xs:annotation>
  <xs:sequence>
    <xs:element name="pub_id" type="xs:string"/>
    <xs:element type="name"/>
    <xs:element name="city" type=”xs:string”/>
  </xs:sequence>
</xs:complexType>
```

Fig. 1. A table within an adaptation model defined with XML Schema

The use of XML seems to be adapted to the definition of models, indeed XML Schema to define complex data structures with powerful constraints, however as we can see in figure 2 it implies an extensive knowledge of this language. A lot of software such as Altova XML Spy [1] have been developed to model graphically XML Schema models as trees. These software allow optimizing the modeling of XML schemas but each of them proposes a different formalism of representation, thus creating some confusion during the modeling of these schemas. A Model Driven Engineering (MDE) approach appears to be a solution to the difficulties encountered during the modeling of such data structures. The objective of a MDE approach is to move the complexity of the realization of an application to the specification of this one. It is then a question of making an abstraction of the programming language using an abstract modeling process. The next section deals with a MDE approach applied to the Master Data Management.

4 MDE Approach Applied to Master Data Management

The introduction of an IDM approach applied to Master Data Management aims to make the definition process of a pivot data model generic and standard. To achieve this, we introduce an abstraction layer by UML meta-modeling enabling an adaptation model to be represented regardless of its application domain.

4.1 Meta-modeling of Adaptation Models

Meta-modeling of adaptation models is a first step to the introduction of an IDM approach. Meta-modeling was standardized by the OMG [4] who recommended the use of Meta Object Facility (MOF) for the definition of meta-models. Meta-modeling of an adaptation model aims at abstractly representing the semantics dedicated to the representation of a pivot data model associated to the MDM domain. The OMG recommends using UML formalisms to define a meta-model and Object Constraint Language [12] to specify the constraints between its elements. However, these two formalisms are not sufficient and present a limited number of entities for representing models, called Platform Specific Model (PSM), associated with a particular technology. To overcome these limitations, it is possible to specialize the UML meta-model by adding supplemental elements and constraints. This specialization is possible through UML Profiles. To implement our solution, we focus essentially on the M2 layer of the MOF architecture to “meta-model” an adaptation model with UML profiles.

4.2 Meta-models Enrichment Using UML Profiles

Our objective is to facilitate and to standardize modeling of adaptation models based on the XML Schema formalism and dedicated to the MDM domain. Up to now, graphical modeling of XML Schema models is not standardized. Tools for modeling

XML models certainly exist but they are restricted to the semantics of XML Schema. Indeed, these tools are unable to guide the user in using the concepts introduced by adaptation models, and more generally in using specific extensions. Moreover, these modeling tools offer graphical representations differing from one solution to another, which represents a potential source of confusion during modeling phases when different actors may intervene. The introduction of a standard formalism of model definition is a way to standardize and to make modeling more accessible. UML is an object modeling language increasingly used and recognized nowadays as a standard in the domain of software engineering, which makes it an ideal candidate for modeling adaptation models. The specialization of the UML language through profiles is a way of standardizing and making the definition of adaptation models generic. These models being initially defined with XML Schema and dedicated to the Master Data Management domain, we define two UML profiles with the former is dedicated to the semantics of XML Schema, and the latter is applied to the semantics of Master Data Management.

4.2.1 XML Schema Profile

UML is an object modeling formalism that defines notions such as generalization, composition and aggregation. Firstly, we introduce meta-data materializing these object specificities in the XML Schema meta-model (noted as name_concept_object_UML on line 3 of Figure 2). The addition of these notions "object" is a first step to homogenizing UML and XML Schema formalisms. To do so, we use extensions mechanisms recommended by XML Schema, *i.e.*, for each meta-knowledge, a description in the form of the following extension:

```
…<xs:annotation>                              [1]
        <xs:appinfo>                                [2]
          <osd:nom_concept_objet_UML/>              [3]
        </xs:appinfo>                               [4]
</xs:annotation>…                             [5]
```

Fig. 2. XML Schema extension representing a meta-knowledge object

The addition of these meta-data in adaptation models allows UML object specificities to be included and relations between some concepts highlighted. Beyond our mapping process between XML Schema and UML, these meta-data contribute to optimizing processes such as the factorization of data, tree optimization, and removal of instances that have become useless. After introducing the UML object specificities in the XML Schema meta-model, we can define the corresponding UML profile. The UML extension mechanism enables us to extend its formalism to the semantics of XML Schema. This extension is defined by stereotypes and marked values. The

stereotypes are used to define a new type of element from an existing element of the UML meta-model. Marked values are interpreted as attributes of a UML meta-class and allow predefined values to be associated to a stereotype instance. Figure 3 presents an extract of the XML Schema profile that we defined.

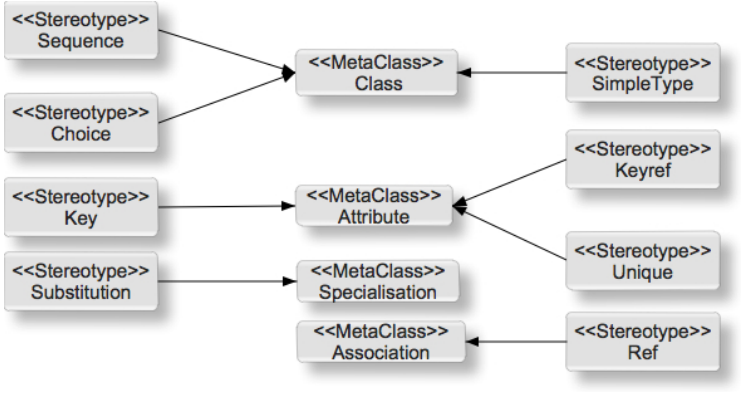

Fig. 3. XML Schema[6] profile sample

Stereotypes of Figure 3 (named « Stereotype ») inherit respectively from the elements Class, Attribute, Association and Specialization of the UML meta-model. Therefore, each of these stereotypes will be instanced by the meta-model builder in the same way as the elements *Class*, *Attribute*, *Association* or *Specialization*. Moreover, some marked values can be associated to some stereotypes. They specify keys-to-values pairs to fix a set of existing element properties or defined stereotypes. The definition of these stereotypes allows the introduction of more semantics, exterior to UML, enabling us to represent an XML Schema model with UML diagrams. However, the use of class diagrams imposes the application of restrictions concerning their definition. Indeed, some UML concepts such as operations, interfaces or internal classes, cannot be represented with XML Schema and must therefore be excluded during the definition of an XML Schema model *via* our UML profile. From this profile, it is possible to define an XML Schema model with a class diagram. The second step of specialization of the UML meta-model consists in defining a profile dedicated to the semantics of Master Data Management.

4.2.2 Master Data Management Profile

We build a profile dedicated to Master Data Management relying on the properties defined in our MDM EBX.Platform solution[7]. Figure 4 presents an extraction of our UML profile representing the MDM meta-model:

[6] See XML Schema specification for more informations http://www.w3.org/TR/xmlschema-0/
[7] Online documentation available at http://doc.orchestranetworks.com/

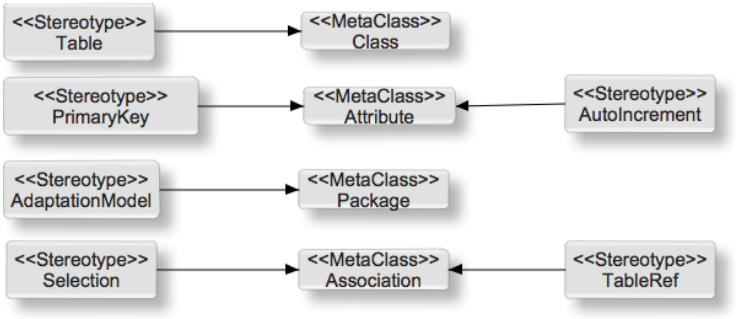

Fig. 4. MDM profile sample

Stereotypes of Figure 4 inherit from meta-classes *Class*, *Attribute*, *Package* and *Association* of the UML meta-model. The *Table* stereotype applied to the *Class* element enables us to indicate that a class must be interpreted as a table in the meaning of DBMS[8] and will have the associated properties (primary key, indexation, etc.). The *PrimaryKey* and *AutoIncrement* stereotypes applied to an element of type Attribute indicate respectively that an attribute is a primary key or is auto-incremented in the meaning of DBMS. The AdaptationModel stereotype applied to an element of type Package indicates that this element represents an adaptation model. The *TableRef* stereotype specify that an association materializes a "foreign key" constraint in the meaning of DBMS. The *Selection* stereotype is used to materialize a inversed relation of foreign key between two entities. For example, if we consider that a book is written by an author, which is made explicit by means of a foreign key, the inverse relation (an author wrote some books) is expressed with this stereotype.

Through these two profiles, we introduce an abstraction layer enabling us to represent an adaptation model independently of an application domain.

4.2.3 UML Modeling of Adaptation Models

In [10], we presented mappings enabling bidirectional transformations between UML and XML Schema to be realized. These mappings take advantage of the elements of the UML and XML Schema meta-models, and are implemented by transformation rules. Our IDM approach allows moving in an automatic manner from an abstract model (UML) to a productive model (XML Schema) interpretable by our MDM EBX.Platform solution. To apply our approach, we have developed a modeling tool that we named ArgoXSD, enabling to define adaptation models by extension of the XML Schema models. The tool us that we have developed is based on the IDE (Integrated Development Environment) ArgoUML [3]. ArgoUML is an open source tool for UML modeling. Based on ArgoUML, we developed a module including the UML profiles previously presented, and the functionalities for importing and exporting from XML Schema models. The import function allows us to generate a UML class diagram from a XML Schema model. The export function enables us to generate the XML Schema code of a class diagram defined with our UML profiles. Figure 5 presents the UML modeling for a simplified train network:

[8] DataBase Management System.

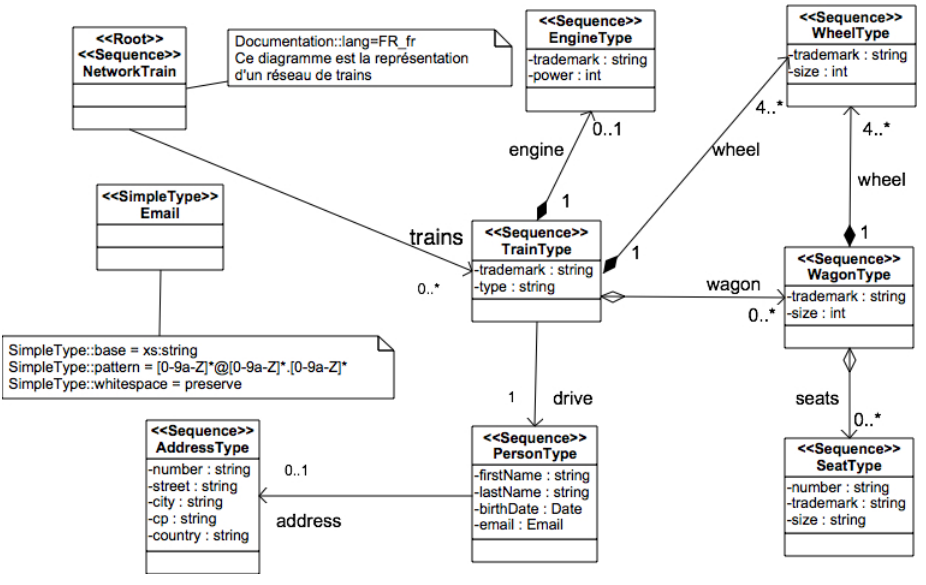

Fig. 5. Adaptation model defined with a UML class diagram

This diagram is composed of different concepts and relations between them. This example illustrates the relations of association, composition, aggregation, and derived type. We have defined the concept of train as composed of an engine, wheels (notion of composition), that may own cars (notion of aggregation), and having properties such as a type and a trademarks. We associate a driver of Person type to a train. The concept has properties such as name, first name and date of birth represented by a UML base data type. We defined an Email property representing the use of a type of redefined data. The Email class has the *SimpleType* stereotype enabling us to indicate that it is a redefined type in the XML Schema sense. The properties of this redefined type are contained in a UML annotation specifying values for *SimpleType::base, SimpleType::pattern and SimpleType::whitespace*. The root of the schema is materialized by the stereotype "Root", applied to the NetworkTrain class.

5 Conclusion

In this paper, we have presented how to introduce a Model Driven Engineering approach to optimize and standardize the definition of data models. The approach that we have used by defining two distinct UML profiles is applicable in a broad sense to all modeling of XML Schema models and can also be applied to the specialized Master Data Management domain. Coupled with transformation methods, the use of our UML profiles enables abstraction of all technical specificities linked to the definition of XML Schema models applied to the MDM to be made.

Later in our works, we will tackle problems of incremental validation of models in order to optimize the validation processes during the conception phases. The quality of MDM linked data is an aspect that we have to consider as data quality management

solutions work even better when they run from a unified data repository, constructed from a joint information model, *i.e.* from MDM. The task of meta-modeling of this model imposes itself as an essential step both for data quality and for MDM.

References

1. Altova XMLSpy, http://www.altova.com/xmlspy
2. Abiteboul, S., Cluet, S.: The Xyleme Project. Computer Networks 39 (2002)
3. ArgoUML (2002), http://argouml.tigris.org/
4. Cattell, R.G.G., Barry, D.: The Object Data Standard: ODMG 3.0. Morgan Kauffman Publishers, San Francisco (1999)
5. Garcia-Molina, H., Papakonstantinou, Y., Quass, D.: The STIMMIS approach to mediation: Data Models and Languages (1995)
6. IBM MDM, http://www-01.ibm.com/software/data/ips/products/masterdata/
7. iWays Master Data Center, http://www.iwaysoftware.com
8. Iyengar, S., Brodsky, A.: XML Metadata Interchange (XMI) Proposal to the OMG Object Analysis & Design Task. Object Management Group (1998), http://www.omg.org
9. Orchestranetworks, http://www.orchestranetworks.com
10. Menet, L., Lamolle, M.: Towards a Bidirectional Transformation of UML and XML Models. In: Proceedings of the 2008 E-Learning, E- Business, Enterprise Information System and E-Government, EEE 2008, Las Vegas, Nevada, USA, July 14-17 (2008)
11. MOF, MetaObject Facility 2.0 (2006),http://www.omg.org/mof/
12. OCL. Response to the UML 2.0 OCL.(2006) http://www.omg.org/spec/OCL/2.0/
13. Oracle MDM suite, http://www.oracle.com/master-data-management/
14. UML, Unified Modeling Language (2009),http://www.omg.org/spec/UML/2.2/
15. W3C, EXtendible Markup Language (2000), http://www.w3.org/TR/REC-xml
16. W3C, XML-Schema (2004), http://www.w3.org/TR/xmlschema-1
17. Garcia-Molina, H., Papakonstantinou, Y., Quass, D.: The STIMMIS approach to mediation: Data Models and Languages (1995)