Integrating Different Technologies for Data Semantics Modeling

Ma. Laura Caliusco
CIDISI (UTN-FRSF)-CONICET
Santa Fe, Argentina, (3000).
mcaliusc@frsf.utn.edu.ar

Ma. Rosa Galli and Omar Chiotti
INGAR-CONICET-UTN
Santa Fe, Argentina, (3000).
{mrgalli, chiotti}@ceride.gov.ar

Abstract

Nowadays, an increasing percentage of data is becoming available in eXtensible Markup Language (XML). Even small discrepancies in the way XML data is defined could cause misunderstanding problems. Consequently, being able to explicitly model the data semantics promises to move information integration technology to a new level of flexibility and automation. The main tools for data semantics specification are based on ontology definition from artificial intelligence techniques. Although these tools provide the functionalities that are necessary and enough for defining a well-conformed ontology, they have not been incorporated into the information system development as expected since these tools assume a certain amount of background knowledge that a lot of people working in ontology lack. This paper presents a tool that make the task of information semantics modeling friendly for people who has not background knowledge in artificial intelligence techniques.

Keywords: conceptual data modelling, contextual ontology, information semantic interoperability.

Resumen

Hoy en día, un porcentaje creciente de datos está disponible en el lenguaje eXtensible Markup Language (XML). Aún pequeñas diferencias en la forma en que los datos basados en XML son definidos podría causar problemas de malos entendidos. En consecuencia, modelar explícitamente la semántica de los datos promete llevar la tecnología de integración de la información a un nuevo nivel de flexibilidad y automatización. Las principales herramientas para la especificación de la semántica de los datos están basadas en la definición de ontologías a partir de técnicas de inteligencia artificial. A pesar de que estas herramientas proveen las funcionalidades necesarias y suficientes para definir una ontología bien conformada, ellas no han sido incorporadas en el desarrollo de los sistemas de información como se esperaba debido a que asumen cierto conocimiento que muchas personas que trabajan en ontologías carece. Este trabajo presenta una herramienta que hace la tarea de modelado de la semántica de la información más amigable para las personas que carecen de conocimientos en el área de inteligencia artificial.

Palabras claves: modelado conceptual de datos, ontología contextual, interoperabilidad semántica de la información.

1 INTRODUCTION

Conceptual modeling plays an important role in the process of information system development. Conceptual models represent the main requirements of the user requirements in an abstract representation about some relevant aspects of a

real world domain. This paper presents our interest in the problem of defining a conceptual data model, that is, a precise definition of the data requirements of an information system that is understandable to both users and developers.

During the design phase of an information system development, system analysts capture and represent all relevant data types, their relationships and constraints on a conceptual data model. Nowadays, the major software applications are based on the object-oriented paradigm, so the language used for data modeling is the Unified Modeling Language (UML) [30]. A conceptual data model is then used to define the physical schema within a certain application. For years, these data structures were represented by data bases. However, during the last years, an increasing percentage of corporate data is becoming available in or is being completely moved to the eXtensible Markup Language (XML) [31].

An important aspect to be taken into account is the information integration. Even small discrepancies in the way XML Schemas are defined could cause serious problems during information sharing among software applications. Consequently, the ability to explicitly model the meaning of information, its semantics, promises to move information integration technology to a new level of flexibility and automation [1]. The main approach for data semantics specification is based on the ontology definition. Since ontologies are intended to be used at run-time of applications, several machine processable languages for ontology definition were defined. But, a great deal of work remains to be done at conceptual model level of semantics.

Taking into account the considerations above discussed, we have worked on a solution for satisfying the need for data models with richer semantics and the later mapping into an object-oriented and XML-based applications. This paper presents the D@SS-Modeler: a data syntactic and semantic modeling tool that integrates object oriented, XML and ontologies approaches. The main objective of this tool is to support data modeling at the design phase taking into account syntactic and semantic aspects; enabling the later translation into object oriented applications and ontology specification languages.

The paper is structured as follows. In Section 2, we discuss the state of the art of conceptual data model techniques and the need of creating new approaches. Section 3, comprises our approach for conceptual data modeling. In Section 4, we present the D@SS-Modeler tool that integrates the ideas presented in our approach. Finally, we discuss our conclusions and future work in Section 5.

2 State of the Art and Motivation

An information system development can be viewed as a two-phase process: Domain Modeling and Programming, as it is shown in Figure 1. During the Domain Modeling Phase, modelers define a conceptual data model about the world under consideration. A conceptual data model is a map of concepts and their relationships. Then, this model is used to generate the logical schema, a description of physical data structures in an abstract and often graphical way [24]. Generally, these tasks are supported by a software modeling tool so as to make them easier. Finally, at the Programming Phase these models are used by programmers, to define the physical schema within applications to be used at run-time. A physical schema defines the data structures using an implementation language and contains all the needed physical design choices and physical storage parameters. In addition, a physical schema depends on current computer technology. Finally, due to the dynamics of certain applications, these tasks are carried out in an incremental way.

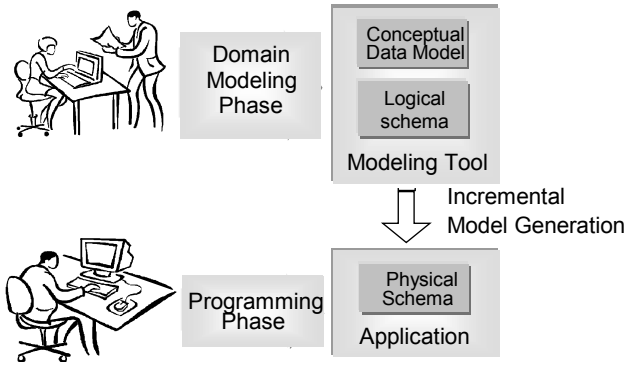

Figure 1. Phases of the information system development.

If we focus on the Domain Modeling Phase of Figure 1, we can say that there are several specification languages that modelers can use to generate *conceptual data models*. The most used are: *Entity-relationship (ER) model* [8] which

provides a graphical notation for representing data models in the form of entity-relationship diagrams; and the *Unified Modeling Language (UML)* [30] which is an open method used to specify, visualize, construct, and document the artifacts of an object-oriented software-intensive system under development.

Nowadays, UML is a widely recognized and used standard. The UML represents a compilation of the best engineering practices which have proved to be successful in modeling large and complex systems, especially at the architectural level. One reason for this is the growing existence of object oriented-based approaches and, as a result, the availability of different tools for conceptual modeling using the UML language that make the modeler tasks easier.

2.2 From Conceptual data modeling to Physical Schemas

A conceptual data model has to be first mapped on a logical schema and then on a physical schema, as it is shown in Figure 1. Generally, the physical schema is generated by programmers taking into account that this schema has to be manipulated by applications at run time.

During the last years, an increasing percentage of corporate data is becoming available in or is being completely moved to the eXtensible Markup Language (XML) [1]. Moreover, a lot of business document specifications for electronic commerce have been defined. Examples of these specifications are: ebXML [32], RosettaNet [25], and OAGIS [21].

All XML documents have to be associated to a definition and have to comply with the grammatical constraints of that definition. This definition could be expressed using a DTD (Document Type Definition) or an XML Schema [16]. However, due to its expressiveness, XML Schema is becoming the most common method for defining and validating highly structured XML documents.

So, the physical schema and the logical schema are now depending on XML technologies. Consequently, different works have been developed trying to link XML technologies with conceptual modeling techniques. Examples of these efforts are extensions of the ER model and UML for creating logical schemas of XML Schemas.

As regards the ER extensions, we can mention X-Entity [15] and ERX (Entity Relationship for XML) [23]. On the one hand, the X-Entity representation provides a cleaner description for XML schemas hiding implementation details and focusing on semantically relevant concepts. However, X-Entity does not consider some issues, such as: hierarchy of elements and attributes, cardinality of group of elements, elements with mixed content and order of elements imposed by a sequence compositor. However, this model can be easily extended with additional features. On the other hand, Psaila [23] introduces ERX as a conceptual model based on the entity relationship model that copes with the features of XML. But, ERX does not support multiple features of XML such as mixed content; and does not describe how complex types with their various nuances can be modeled into the system.

With regard to the research lines concentrated on the definition of UML Profiles for modeling XML Schemas, the most significant works are the following. Carlson [7] has defined an UML profile that is a good support for defining model groups' elements, local attribute, and local elements. But, it does not support the definition of all XML Schema elements. On the other hand, Routledge, Bird and Goodchild [24] define an approach in which conceptual level UML class diagrams are transformed through successive steps into XML Schemas. This approach is more complete than the one defined by Carlson with regard to model elements, but, it violates UML semantics in the definition of some elements such as for example the global element definition. Finally, the work defined by Bernauer et al [6] is a complete UML Profile and solves the problem presented in previous works.

2.3 Data Modeling and Data Semantics

By this point, we have discussed the technologies that nowadays drive the data modeling process. These technologies fit into this process as it is represented, with white boxes, in Figure 2. XML Schema fits into Physical Schema level. UML Profile for XML Schema fits into Logical Schema level. And, UML class Diagrams fit into conceptual data modeling level.

If we talk about information integration, even small discrepancies in the way an XML Schema is defined could cause serious problems during information sharing among software applications. This means that, in a XML Schema, the elements are delimited by start and end tags. We can create our own markup, e.g. <ItemQuantity> to refer to a purchase order item. These tags carry some *implicit semantics* for people. However, from a computational perspective, tags like <Item> carry as much semantics as a tag like <H1>. A computer simply does not know what an *Item* is and how the concept *Item* is related to e.g. the concept *Purchase Order*. Furthermore, different modelers could use different terms to refer to the same concept, which is reflected in the applications. For example, one XML Schema employs the tag <Item> instead of the tag <Article>. Moreover, two XML Schemas could use two concepts with the same name but

different meaning. For example, <Item> in a purchase order refers to the item to be required; but <Item> in a forecast refers to a forecasted item.

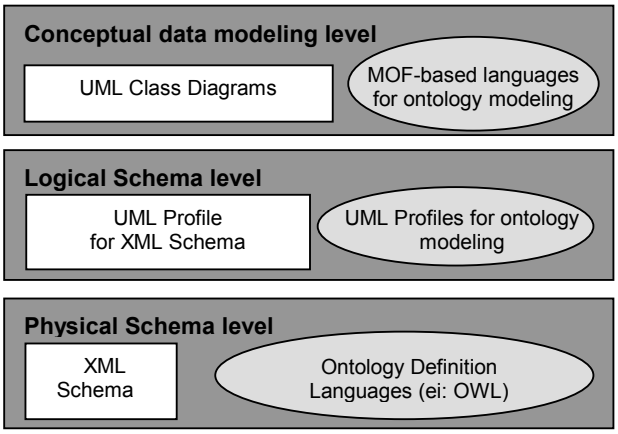

Figure 2. Technologies associated to data modeling process.

Consequently, being able to explicitly model the meaning of information, its semantics, promises to drive information integration technology to a new level of flexibility and automation.

The main approach for data semantics specification is based on the ontology definition. Ontologies are intended to be used at run-time of applications, and for that reasons several machine processable languages for ontology definition were defined [9]. The most recently specified one is the Web Ontology language (OWL) [18]. OWL is a semantic markup language for publishing and sharing ontologies on the World Wide Web. Ontology definition languages fit into the physical schema level as shown in Figure 2. On the one hand, some tools were developed in order to facilitate the ontology definition. Examples of these tools are Protégé [20], WebODE [10] and OntoEdit [27]. Although these tools provide the functionalities that are necessary and enough for defining a ell-conformed ontology, they have not been incorporated into the development of every information systems as expected. One of the reasons for this is that these tools use Artificial Intelligence techniques for ontology creation and they do not integrate with previously defined data modeling techniques that are used when developing every information systems. Nowadays, there are many proposals to use software engineering techniques, especially the UML since it is the most accepted software engineering standard, in order to bring ontology development process closer to an increased practitioners' population. However, UML is based on the object oriented paradigm and has some limitations regarding ontology development. These limitations can be overcome using UML profiles as well as other OMG's standards (i.e. *Model Object Facilities - MOF*). These modeling languages fit the level of Conceptual Data Modeling as shown in Figure 2. Currently, there is an initiative within the OMG whose aim is to define a suitable language for modeling Semantic Web ontology languages. This language, called Ontology Definition Metamodel (ODM) [22], is a MOF2 (Meta Object Facilities) compliant metamodel that allows a user to define ontology models using the same terminology and concepts as those defined in OWL. So, the ODM is driven by the OWL language. Even though ODM is a great advance in the area of semantics modelling, it lacks a good means for context definition. The context definition during the semantics definition is important if we consider that some concepts are true or false depending on their context. Furthermore, it is well known that a human being does not reason without context.

3 Data Modeling at Syntactic and Semantic Level

Taking into account the development stream of data model technologies, we have identified some issues influencing the data modeling phase during an information system development. These issues have to be considered when looking for a tool to integrate data semantic definition with data modeling technologies. They are:

- Nowadays, UML is a widely recognized and used standard for information system development.
- A lot of business standards based on XML for electronic commerce have been defined. Generally, these standards have to be customized to be used by enterprises.

- UML Profiles for XML Schema manipulation improve the understandability and integration of XML Schema in UML-based software development processes.
- The inclusion of ontologies into the information system development is constrained by the lack of a supporting tool based on an appropriate methodology.
- Tools for supporting semantics data model have been developed separately from the most usually used technologies for data modeling. So, there is no bridge between technologies such as object-oriented, XML, and ontology.
- Even though, the need for data models with richer semantics and the later mapping into object-oriented and XML-based applications is widely recognized, no single approach has won general acceptance.
- Initiatives for defining a suitable language for modeling semantics at conceptual level lack a good means for context definition.

Taking the aforementioned points by reference, we have firstly defined a language for semantic modeling based on MOF [11]. This language, called C-OML, fits the level of conceptual data modeling as shown in Figure 3; and incorporates elements for contextual ontology definition. This is the main difference between C-OML and other MOF-based initiatives for modeling data semantics.

Then, we defined an UML Profile for the conceptual modeling of an XML Schema. The main design principle of this UML Profile was to be a bridge between XML Schemas and C-OML language.

Finally, we have implemented a tool that integrates all of these technologies. This tool supports the modeling process represented in black arrows in Figure 3. In this process we assume that a modeler starts from an XML Schema document in order to customize it. This customization is carried out from a syntactic point of view, using a UML profile (1), and from a semantic point of view through the C-OML modeling language (2). Once this modeling process is finished, the semantics is represented by a machine-processable language, such as OWL (3). We present this tool in the next Section.

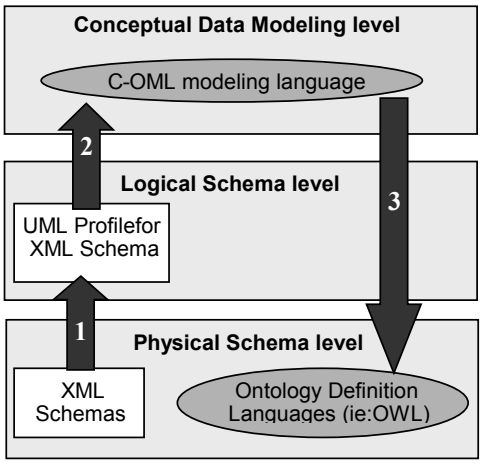

Figure 3. Data Modeling at syntactic and semantics level.

3.1 C-OML: Contextual Ontology Modeling Language

In order to visualize the advantage of contextualizing an ontology, we analyze the following example. Suppose that two enterprises, OilCo and WarehouseA, want to establish a B2B relationship. Each enterprise has its own ontology, as it is shown in Figure 4. In OilCo ontology the term *agent* represents employees who temporarily work for the enterprise. In contrast, in WarehouseA ontology the term *agent* represents employees who are part of the enterprise since they permanently work for the enterprise. Clearly, this term has different meanings in different contexts. In addition, the term used to represent permanent workers within OilCo is *employee*. So, in different contexts there exist different terms to represent the same concept. If we wish not to lose the meaning of each term belonging to different ontologies, a solution is explicit the context of each ontology and then create relations between the terms. For example, at the bottom of Figure 4 we represent that the terms *employee* in the OilCo and *agent* in the WarehouseA are equivalent.

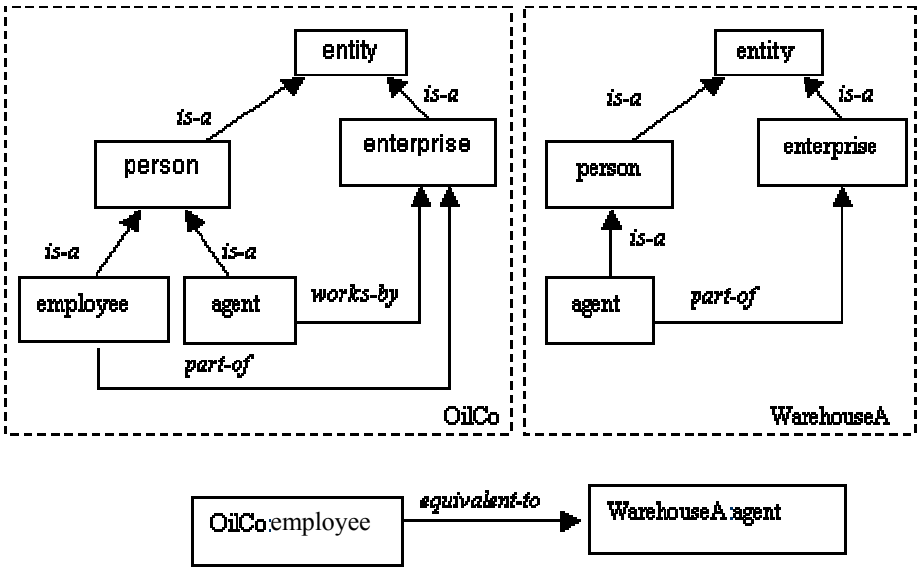

Figure 4. Contextual ontologies.

Taking into account that context specification is an important factor to explicitly define the meaning of concepts in an ambiguous way, we have defined the Contextual Ontology Modeling Language (C-OML) for adding context to semantics at conceptual modeling level. Following, we briefly define contextual ontology concepts that are the basis of our language.

3.1.1. The Notion of Contextual Ontology

Considering different definitions of context [13] [3] [29], we define a context as a collection of relevant conditions or assumptions that make a situation or entity unique and comprehensible. That entity or situation depends on the domain we are. The context definition is formalized as:

Definition 1. *Let J be a set of indexes j , a context* $C_j \ \forall j \in J$ *can be defined as a 3-tuple* $<c_j, D_j, O_{i,j}>$*, where:* c_j *is the unique identifier of context j,* D_j *is a set of assumptions about context j, and* $O_{i,j}$ *represents ontology i within context j.*

Ontology mapping is a well known problem in knowledge engineering [17]. However, if we represent the semantics using a contextual ontology, the mapping has to be made between contexts. That is called *context mapping*.

A context mapping allows us to state that a certain property holds between elements belonging to different ontologies defined in different contexts [5]. Then, a context mapping is defined by *bridge rules* as linking rules between contexts [3]. Following, we formally define both context mapping and bridge rules.

Definition 2*. A context mapping* $M_{s,t}$ *can be defined as a 3-tuple* $<c_s, c_t, BR>$ *where:* c_s *identifies the context source,* c_t *identifies the context target and BR represents the set of bridge rules that map an element from a source context to elements of a target context.*

Mappings are directional, i.e. $M_{s,t}$ is not the inverse of $M_{t,s}$. Mapping $M_{s,t}$ might be empty [2]. That means that there is no relation between both contexts.

Definition 3*. A bridge rule can be defined as a 3-tupple* $< e_s, e_t, R >$ *where:* e_s *is an element from source context,* e_t *is an element from target context, and R is a rule operator between elements.*

A bridge rule is formed by $c_s{:}e_s \xrightarrow{R} c_t{:}e_t$ where e_i and e_j are elements from a source context (c_s) and a target context (c_t) respectively. Examples of bridge rules are:

1) office-equipment:pen $\xrightarrow{\equiv}$ *school-equipment:pen* means that the concept *pen* within *office-equipment* context is

similar to the concept *pen* in *school-equipment* context.

2) informatics:mouse $\xrightarrow{\perp}$ *biology:mouse* means that concept *mouse* in *informatics* context is disjoint from the concept *mouse* within *biology* context.

3) e-commerce:businessdocument $\xrightarrow{*}$ *business:transaction* means that a *businessdocument* in *e-commerce* context, is a compatible concept with *transaction* in *business* context.

4) entertainment:artist $\xrightarrow{\supseteq}$ *television:actor* means that the concept *artist* in *entertainment* context is more general than the concept *actor* in *television* context.

Based on these concepts, we have developed the contextual ontology modeling language, C-OML. In order to define it, we have imported some elements from the Core::Abstractions and Core::PrimitiveTypes Packages of the UML 2.0 specification [30]. Furthermore, our metamodel is improved with OCL constraints with specific invariants and that have to be fulfilled by all models that instantiate C-OML.
The metamodel of the C-OML language has been architected with the modularity design principles in mind. So, the metamodel constructs were grouped into packages according to the elements needed to define a contextual ontology. More details about C-OML metamodel can be found in [11].

3.2 An UML Profile for XML Schema

The Extensible Markup Language, XML, defined by W3C consortium [31] was originally designed to meet the challenges of large-scale electronic publishing, but it is now playing an important role in different applications. XML Schema is becoming the most common method for defining and validating highly structured XML documents due to its expressiveness. An example of an XML Schema document is shown in Figure 5. This document represents a catalog structure defined by the OAGIS standard [21].

```
<?xml version = "1.0" encoding = "utf-8"?>
    <xs:schema xmlns = "D:/BODs/oagis/modificados/pruebas"  version = "0.02"
    elementFormDefault = "qualified" attributeFormDefault = "unqualified">
<xs:simpleType name = "Description">
          <xs:restriction base = "xs:string">
               <xs:maxLength value="40"/>
       </xs:restriction>
</xs:simpleType>
<xs:element name = "Item" type="xs:string">
    <xs:annotation>
        <xs:documentation> Customer's actual requested item amount </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:complexType name = "ItemLine">
    <xs:sequence>
        <xs:element ref = "Item"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name = "CatalogLine">
    <xs:complexContent>
        <xs:extension base="ItemLine">
            <xs:sequence>
                <xs:element name="LineNumber" type="xs:nonNegativeInteger"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name = "Catalog">
    <xs:sequence>
        <xs:element name = "ProductDescription" type = "Description"/>
        <xs:element name = "Line" type = "CatalogLine"/>
    </xs:sequence>
</xs:complexType>
<xs:element name = "CatalogDocument" type = "Catalog"/>
</xs:schema>
```

Figure 5. An XML Schema document.

A number of approaches relating XML Schemas and conceptual models have been described in other works, which are based on the Unified Modeling Language (UML) that has proven to be valuable for data modeling. However, most of them try to solve the problem of how mapping data stored in relational and object oriented databases into XML documents [14] [19].

The discussion about the relationship between syntax and semantics is not new. Several approaches have been made for adding semantics to XML documents. The XML semantics definition Language (XSDL) [28] defines XML semantics by mapping XML to ontology. Recently, Patel-Schneider and Simeón have proposed the idea of Ying/Yang Web, in which XML XQuery 1.0 and XPath 2.0 Data Model are regarded as a unified model for both XML and RDF [4]. Furthermore, the BECHAMEL Project [2] is trying to apply knowledge representation technologies to the modeling of meaning and relationship expressed by XML markup. However, all of these approaches deal with the problem at low-level implementation. But, one of our motivations is to provide a graphically designing XML Schema without exposing designers to low-level implementation.

Consequently we have designed an UML Profile for XML Schema manipulation [11]. This is not a new idea because some effort has been made in defining UML profiles for XML Schema [7] [24] [6]. However, one weak point of these approaches is the fact that they do not take into account, as a design requirement, that each element defined in the XML document represents the domain semantics. That is, in an XML document, tags should be interpreted as a semantically meaningful unit of the domain they are representing. This assumption imposes some limitations during the design of the UML profile for an XML Schema.

Figure 6 shows the application of our UML Profile for modeling an XML Schema. In this figure, we can view that each element is defined as a stereotyped class. Consequently, this profile has to be able to represent elements and attributes in an independent way to manipulate them as an ontology concept. And, this is the main difference between our UML Profile and previous approaches. More details about this profile can be found in [12].

```
<xs:complexType name = "Catalog">
  <xs:sequence>
    <xs:element name = "ProductDescription"
                                type="Description"/>
     <xs:element name = "Line"  type ="CatalogLine"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name = "CatalogDocument"
                                type = "Catalog"/>
```

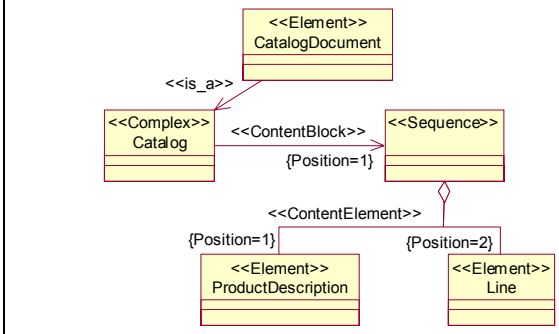

Figure 6. An example of UML Profile for XML Schema.

4 D@SS-MODELER TOOL

The main objective of this section is to briefly outline our D@SS-Modeler Tool prototype. This tool implements the previously defined modeling languages.

D@SS-Modeler Tool was designed to support the semantic definition of XML-based documents. These documents and their semantics are grouped into a context. That means that, the elements defined in an XML-based document are interpreted in a particular way within a certain context, and could be interpreted in a different way within another context. In addition, a set of contexts is represented by a project. So, when we run the tool for the first time, a window for selecting or creating a new project appears. After selecting a project, we can create or select a context; or, if there are at least two contexts defined within the project, we can define bridge rules between them.

Figure 7 shows a screenshot of the D@SS-Modeler Tool which shows a project called *B2B.prj* with two defined contexts, *SalesContext* and *SchedulingContext*. Each context is composed of two main windows: the XML Schemas window and the ontology definition window. We will describe this windows following. For that purpose, we consider the *catalog.xsd* XML Schema which is presented in Figure 5.

4.1 XML Schemas Windows

We can view a schema in three different forms. These forms are: a hierarchy of elements (Tree Tab), a plain text (Source Tab) and a graph according to the UML Profile for XML Schema modeling discussed in Section 3.2 (Graph Tab).

On the left side of Figure 7, we can view the catalog.xsd document as a tree. Each document element is represented as a tree node.

In addition, within each context we can define various business documents based on XML. In this example, catalog.xsd and order.xsd belong to SalesContext context. These XML documents must not to have terms with the same name due to within a context each element has to be unique. One solution to solve this problem is to consider each document as a context. This requirement will be taken into account in future tool improvements.

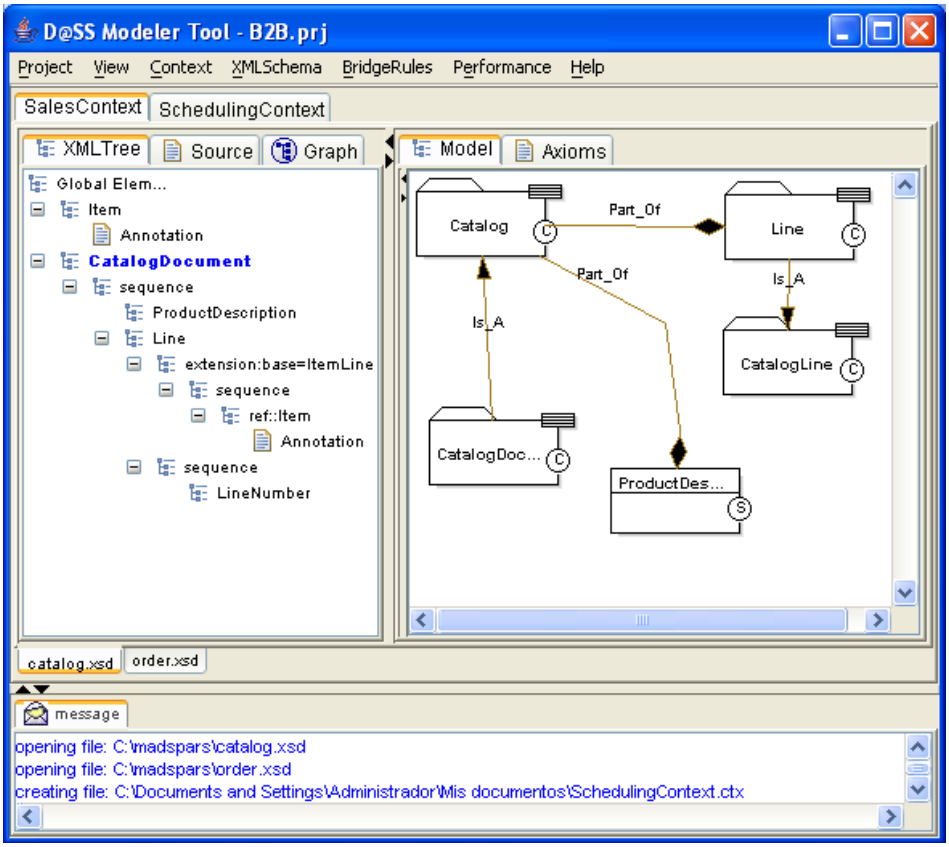

Figure 7. A screenshot of D@SS-Modeler Tool.

4.2 Ontology Definition Window

On the right side of Figure 7, the semantics associated to each element of the catalog.xsd document is represented. This representation is based on our C-OML graphical language defined in Section 3.1. In this figure, we can see that the *Catalog* term is a complex concept composed of *ProductDescription* and *Line*. This relationship is represented by the *Part-of* association.

When a modeler adds a new XML Schema, a basic semantic model that processes the document is automatically generated. In addition, D@SS Modeler Tool supports the definition of different characteristics belonging to an ontology, such us relationships definition, terms definition, and Properties definition.

Furthermore, this tool supports the definition of axioms by clicking on Axiom tab. Axioms are properties of relations; they help to constrain interpretation of concepts and they provide guidelines for automated reasoning. Although there are different kinds of axiom, in this prototype a user could define non-relational and relational axioms.

4.3 Bridge Rules Definition

After defining contexts and elements that compose them, the tool allows a user to define mapping rules between contexts by using bridge rules as the one presented in Section 3.1.1. An example of bridge rule definition is shown in Figure 8. This figure shows on the left side the terms of PlanningContext, on the right side the terms of SalesContext and on the middle the bridge rules. The tool supports up to now the definition of mapping rules between terms, without supporting the definition of mapping rules between properties or restrictions.

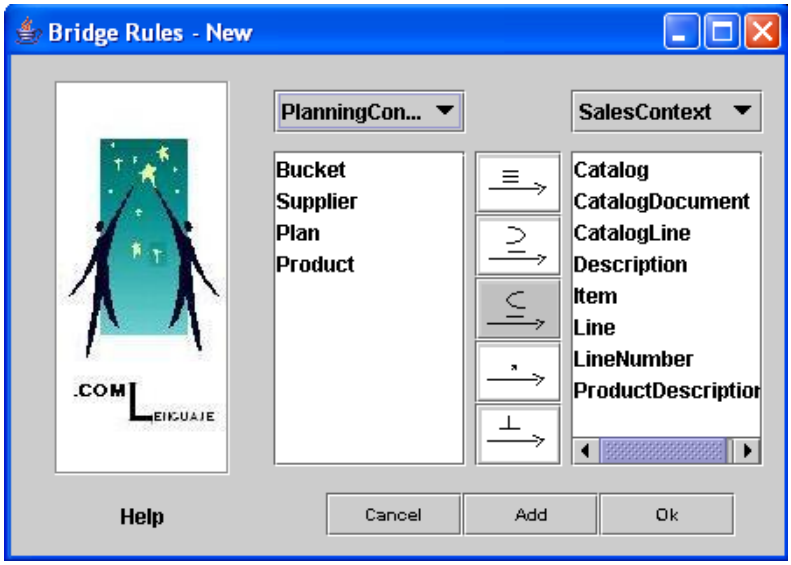

Figure 8. A screenshot of bridge rules definition.

5 Conclusions and Future Work

Methodologies and tools for information system development should assist the developer in making decisions about those aspects of the analysis, design and implementation that are crucial for the system. In this work, we have presented a tool that integrates different technologies, such as XML, object-oriented and ontologies.

This tool, called D@SS-Modeler, is intended to be a bridge between design models and the existing implementation languages. Differently from other applications this tool supports the data semantic definition without a base on Artificial Intelligence techniques in order to facilitate its incorporation into an information system development.

This is just the first step and further research remains to be done. In this paper, we show a simple scenario, we will study the tool difficulties applying it to more complex real cases. Furthermore, we will focus on studying conversion rules that allow us to establish appropriately mapping between different contextual ontologies.

References

[1] Alexiev, V.; Breu, M.; de Bruijn, J.; Fensel, D.; Lara, R.; Lausen, H. *Information integration with Ontologies – Experiences from an Industrial Showcase*. John Wiley and Sons. 2005.

[2] Allen, R., Dubin, D., Sperberg-McQueen, C.M., Huitfeldt, C.: *Towards a semantics for XML markup*. In: the 2002 ACM Symposium on Document Engineering, 119–126.

[3] Brézillon (1999) Context in problem solving: A survey. The Knowledge Engineering Review, 14 (1), 1-34.

[4] Beckett, Dave. RDF/XML Syntax Specification. W3C Recommendation. Februay, 2004. Disponible on-line http://www.w3.org/TR/rdf-syntax-grammar/

[5] Bouquet, P, Dona, A, Serafini, L and Zanobini, S. (2002) *Contextualized local ontologies specification via CTXML*. In Proceedings of AAAI-02 Workshop on Meaning Negotiation (MeaN-02) Edmonton, Canada.

[6] Bernauer, M.; Kappel, G.; Kramler, G. *Representing XML Schema in UML - A UML Profile for XML Schema.* Technical Report, November 2003. Business Informatics Group. Institute of Software Technology and Interactive Systems, Vienna University of Technology

[7] Carlson, D. *Modeling XML Applications with UML*. Addison-Wesley, 2001.

[8] Chen, P.S. The entity-relationship model: Towards a unified view of data. ACM Transactions on Database Systems, Vol. 1:9-36, 1976.

[9] Corcho, O., Fernández-López, M., Gómez-Pérez, A. Methodologies, tools and languages for building ontologies. Where is their meeting point?. Data & Knowledge Engineering 46 (2003) 41–64.

[10] Corcho O, Fernández-López M, Gómez-Pérez A, Vicente O. WebODE: an integrated workbench for ontology representation, reasoning and exchange. Lecture Notes on Artificial Intelligence Vol 2473. 13th International Conference on Knowledge Engineering an Knowledge Management. Springer-Verlag. pp: 138-153. October 2002.

[11] Caliusco, M. L., Maidana, C., Galli, M.R. and Chiotti O. (2006) Contextual Ontology Modeling Language to facilitate the use of enabling Semantic Web Technologies. *Web Semantics and Ontology*. Idea Group Inc, pp. 68-89.

[12] Caliusco, Ma. Laura, Maidana, César, Patiño, Martín, Galli, Ma. Rosa, Chiotti, Omar. *An UML Profile for XML Schema*. Segundo Simposio Argentino de Sistemas de Información (ASIS) - 34 Jornadas Argentinas de Informática e Investigación Operativa (JAIIO) Rosario - Argentina.

[13] Dey, A. (2001). Understanding and using context. Personal and Ubiquitous Computing Journal, Volume 5 (1), 4-7.

[14] Fong, J.; Pang, F.; Bloor, C. *Converting Relational Database into XML Document*. In Proc. of the 12th International Workshop on Database and Expert Systems Applications. pp 61 – 65. 2001.

[15] Farias L´osio, B; Salgado, A; Rego Galvao, L. Conceptual Modeling of XML Schemas. 5th ACM International Workshop on Web Information and Data Management, New Orleans, Louisiana, November 2003.

[16] Fallside, D.C. and Walmsley, P. (October, 2004) *XML Schema Part 0: Primer Second Edition*. W3C Recommendation. Retrieved from http://www.w3.org/TR/xmlschema-0/

[17] Kalfoglou, Y. and Schorlemmer, M. (2003 January) *Ontology mapping: the state of the art.* The Knowledge Engineering Review 18(1):1.

[18] McGuinness, D and van Harmelen, F. OWL Ontology Web Language - Overview. W3C Recomended Propesed. December 2003. http://www.w3.org/TR/owl-features/.

[19] Mlynkova, I., Pokorny, J. *XML in the World of (Object-)Relational Database Systems*. In Proc. of the XIII International Conference (ISD 2004), Lithuania, September 2004.

[20] Noy, N. F. M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, & M. A. Musen. (2001) Creating Semantic Web Contents with Protege-2000. IEEE Intelligent Systems 16(2):60-71.

[21] The Open Applications Group XML Project Team. OAGIS Release 8.0. (May, 2002). http://www.openapplications.org/downloads/oagidownloads.htm

[22] Ontology Definition Metamodel Request for Proposal, OMG Document: ad/2003-03-40, http://www.omg.org/cgi-bin/doc?ad/2003-03-40, 2003.

[23] Psaila, G. *From XML DTDs to Entity-Relationship Schemas*, pp. 378-389, In: *Conceptual Modeling for Novel Application Domains, ER 2003 Workshop Proceedings*, Manfred A. Jeusfeld, Óscar Pastor (Editors), Chicago, Illinois, Springer-Verlag, October 2003.

[24] Routledge, N., Bird, L., and Goodchild, A. *UML and XML Schema.* In Proc. of the 13th Australasian conference on Database technologies (ADC2002) - Volume 5. Melbourne, Victoria, Australia pp: 157 – 166, January/February 2002.

[25] RosettaNet (2003). http://www.rosettanet.org

[26] Sure, Y., Angele, J., Staab, S. OntoEdit: Multifaceted Inferencing for Ontology Engineering *Journal on Data Semantics* 1 (1): 128-152. November 2003. LNCS 2800, Springer.

[27] Sure, Y; Angele, Jand Staab, S. (2003) OntoEdit: Multifaceted Inferencing for Ontology Engineering. Journal on Data Semantics 1(1):128-152.

[28] Shengping Liu, Jing Mei, Anbu Yue and Zuoquan Lin: XSDL: Making XML Semantics Explicit. Semantic Web and Databases: Second International Workshop, SWDB 2004, Toronto, Canada, August 29-30, 2004, Revised Selected Papers, LNCS 3372.

[29] Theodorakis, M. and Spyratos, N. (2002) *Context in Artificial Intelligent and Information Modeling*. In Proc. of 2nd Hellenic Conf. on Artificial Intelligence (2002).

[30] UML 2.0. The Unified Modeling Language Superstructure version 2.0 – OMG Final Adopted Specification. August 2003. http://www.omg.org

[31] W3C (2000): Extensible Markup Language (XML) 1.0. W3C XML Working Group, http://www.w3c.org/TR/REC-xml.

[32] Waldt, D. and Drummond, R. *EbXML - The Global Standard for Electronic Business*. http://www.ebxml.org