Contextual Ontology Definition Metamodel

Ma. Laura Caliusco[1], César Maidana[1], Ma. Rosa Galli[1,2], and Omar Chiotti[1,2]

[1] GIDSATD – UTN – FRSF, Lavaisse 610 – 3000 Santa Fe - Argentina
{mcaliusc, cmaidana}@frsf.utn.edu.ar
[2] INGAR-CONICET, Avellaneda 3657 – 3000 Santa Fe – Argentina
{mrgalli, chiotti}@ceride.gov.ar

Abstract. An effective collaborative B2B relationship requires the right modeling of collaborative processes and business information needed to support these processes. Business information modeling implies modeling the data syntax and semantics. In order to model information semantics there are some ontology specification languages. However, from a B2B perspective, the main disadvantage of these languages is that they are mostly based on logic formalisms to support machine reasoning. This makes the language syntax unfamiliar to business analysts who model the business information to be exchanged within collaborative processes. The objective of this paper is to present a metamodel for modeling explicit and formal contextual ontologies, for human processing, associated to business documents.

Keywords. Contextual ontologies, B2B e-commerce, Metamodel, UML, XML.

1 Introduction

Today, global customers demand more quantity, better quality, a better service, more choices and more innovation. For getting all of these, the global customers neither like to spend much time in getting it, nor do they take any risk but prefer to pay a reasonable price. In this scenario, technologies associated to collaborative B2B relationships are the most vital tools to meet these challenges.

A collaborative B2B relationship implies business information interchange between business partners, and cross-company business interactions are still a problem. When information passes between companies, errors, inconsistencies and misunderstandings occur very often, leading to wasted work efforts. To solve this problem XML-based specifications were defined to exchange business information between partners. These specifications propose to use the same vocabulary, that is, the same collection of terms.

However, this is unrealistic since in the same department of different enterprises, people could use the same term to define different concepts. This problem is known as semantic interoperability problem. In order to overcome this problem we have to explicitly define the meaning of the terms, i.e., its semantics [5].

A common approach to represent semantics is to use an ontology. However, there is an emerging approach that combines an ontology with their context definition [2]. That is, a concept is true or false depending on its context. So, if we explicitly define the context, we avoid misunderstanding. The resulting structure is called contextual ontology.

To process a contextual ontology at run time, it has to be expressed in a machine processable language. Recently, some languages have appeared, i.e, C-OWL [2]. However, from B2B perspective, the main disadvantage of contextual ontology specification languages is that they are mostly based on logic formalisms to support machine reasoning. This makes the language syntax unfamiliar to business analysts who model the business documents to be exchanged. Furthermore, the collaborative processes and the business documents have to be implemented by software engineers who have to interpret the information model.

A model consists of sets of elements that describe some physical, abstract, or hypothetical reality. Good models serve as means of communications [13]. In order to overcome the gap between people involved in the business documents definition and ontology specification languages, an ontology modeling language is needed.

A metamodel is simply a model of a modeling language. It defines the structure, semantics, and constraints for a family of models. A model is captured by a particular metamodel [13].

Our goal is to define a metamodel that assists business analysts in the modeling of contextual ontologies. The objective of this paper is to present a metamodel for modeling explicit and formal contextual ontologies, for human processing, associated to business documents. Firstly, we present a metamodel derived from UML 2.0 infrastructure. Then, we analyze the relationship between XML specifications and ontologies in order to add formal and explicit semantics to business documents. Finally, we present our conclusions and future work.

2. Contextual Ontology Definition Metamodel.

In order to overcome the gap between people involved in the business documents definition and ontology specification languages, there are proposals for using UML as ontology modeling language [7]. But, UML itself does not satisfy needs for representation of ontology concepts that are borrowed from Descriptive Logic and that are included in ontology specification languages [8], like the Web Ontology Language (OWL). The Ontology Working Group is defining the Ontology Definition Metamodel (ODM) [14].

The ODM 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. OWL is a semantic markup language for publishing and sharing ontologies on the World Wide Web [12]. So, the ODM is driven by the OWL language. Furthermore, in this metamodel the context definition is supported by using annotations. But, some concepts are true or false depending on their context. It is well known a human being does not reason without context.

Our goal is to define a metamodel that assists business analysts in the modeling of contextual ontologies. The typical role of a metamodel is to define the semantics for how a model element in a model gets instantiated. In order to define the proposed metamodel we have imported some elements of the Core::Abstractions and Core::PrimitiveTypes Packages of the "UML 2.0: Infrastructure" specification [16]. The Core::Abstractions contains a set of metaclasses, most of which are abstracts, to be specialized when defining new metamodels compliant with MOF2. The Core::PrimitiveTypes simply contains a number of predefined types that are commonly used when metamodeling.

The main design principles of the metamodel are: (1) easy to use in rapid development of contextual ontologies from business documents, (2) modularity and (3) high independence degree of conextual ontology specification languages.

In order to fill the modularity design principles, the metamodel constructs were grouped into packages. The main package is the Kernel Package which imports the reused elements from Infrastructure::Core Package. All metamodel elements are derived from Kernel elements.

Each element of the metamodel is described by using a *description*, *constraints* and *semantics*. The description includes an informal definition of the element. The constraints are the well-formedness rules which must be satisfied by all instances of a metaclass for the model to be meaningful. Semantics is the meaning of a well-formed construct and it is defined using natural language.

Following we define the main packages of the proposed metamodel.

2.1 Ontology Package

This package contains classes and associations that can be used to define an ontology. The ontology term has been widely discussed in the AI area [6]. An ontology represents an explicit specification of domain conceptualization. A domain conceptualization names and describes the entities that may exist in that domain and the relationships among those entities. It therefore provides a vocabulary for representing and communicating knowledge about the domain [9].

An ontology can be defined as a set of *terms* and *relations* between them. Furthermore, it is suitable to add *properties* and *axioms* to enrich the ontology. The set of properties

defines the characteristics of *terms*. *Axioms* are properties of the relations. For example, *PurchaseOrder* is a sub-class of *Order*, and this relation is not a symmetric one (axiom).

Figure 1 represents the classes and associations of the Ontology Package. The main component is *Ontology* class that includes definition of concepts used to describe and represent a domain. This class is associated with the *OntologyElements* abstract metaclass, which groups the objects of an ontology metamodel. If an ontology is removed, so are the elements owned by it. The association *imports* represent that an ontology could contain definitions whose meanings are defined in other ontologies. The association *prior_Version* identifies the referred ontology as a prior version of one ontology. Each ontology element could be described by a comment, represented by the *Documentation* class which derived the *Comment* abstract class. The *Body* atribute specifies a string that is the comment. This class intend to model, for example, the *xsd:annotations* elements from XML-based documents.

The *Feature* class intend to represents the characterisctics of an ontology such as, creation date, version and so on.

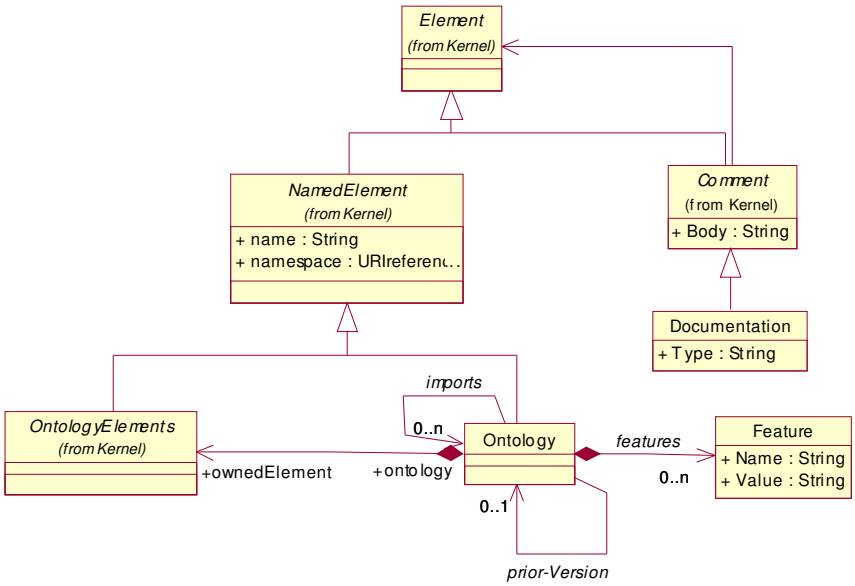

Figure 1. Elements defined in the Ontology package.

2.2 Properties and Terms Package

This package contains classes and associations that can be used to model terms and their properties. Terms represent the set of concepts that a business analyst wants to represent in an ontology. A term can be simple or complex. Simple terms have literal of some kind as their values. Complex terms are composed by simple terms.

Properties describe the features of a term. For example, allowed values, the number of the values, and other features of the values that a simple or complex term could take.

The metamodel that represents the relation between *Properties* and *Terms* is presented in Figure 2. In the proposed metamodel, the class *Properties* defines the features of a term so if a term is removed the properties owned by it have to be removed also. However, in an ontology specification language one property could exist independently. That is represented by the 0..1 cardinality in the *+property* association.

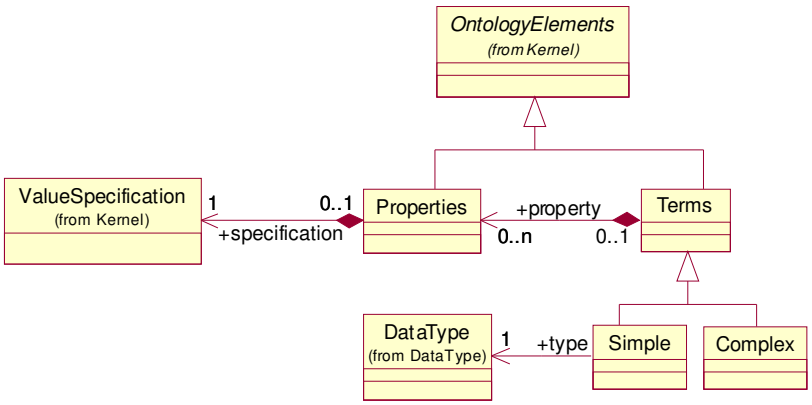

Figure 2. Elements defined in the Properties and Terms Package.

2.3 Relations Package

This package contains classes and associations that can be used to model relations between terms belonging to an ontology. Relations can be divided into hierarchical relations (is-a, part-of and inst-of relations), conceptual relations (synonym and antonym) and particular relations (defined by the modeler).

The relations' metamodel is presented in Figure 3. *Terms* and *Relations* classes are associated via the *RelationEnd* class. An instance of *Relations* class has to be associated at least with two instances of *RelationEnd* class, it is indicated with the label 2..n. *RelationEnd* class is associated with one *Terms* class and contains the information about

cardinality and the role of terms. Furthermore, this class has the *Navigable* attribute to represent the direction of the relation.

One important requirement for ontologies is the ability to structure the relations into hierarchies, i.e., to define sub-relations of a relation. Furthermore, it is suitable to define equivalent relations and inverse relations. Subrelationof, inverseof and equivalentto relations model these characteristics.

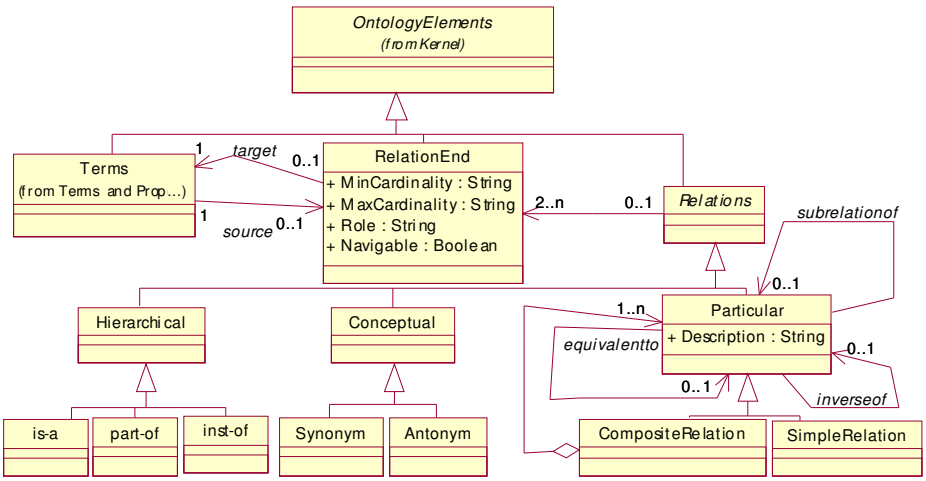

Figure 3. Elements defined in the Relations Package.

2.4 Axioms Package

This package contains classes and associations that can be used to describe axioms about relations. Axioms are properties of relations that help to constrain interpretation of concepts. Furthermore, they provide guidelines for automated reasoning. In the knowledge engineering area, axioms have been represented using logic languages. In the UML class diagram, axioms could be expressed by OCL [16]. For example, OCL constraints have to be used to declare a transitive property of a relation between terms. However, describing such constraints may involve writing moderately complex OCL expressions that are not immediately understandable to a human reader. In addition, there may be several different expressions encoding the same constraint. An interesting issue is to represent axioms as objects [15].

Axioms can be divided into two subsets: the set of axioms for relational algebra and the set of particular axioms. That is, the axioms defined by users. We include symmetric, reflexive, transitive and functional axioms as relational algebra axioms.

Figure 4 represents the metamodel for modeling axioms and their association with the class *Relations*.

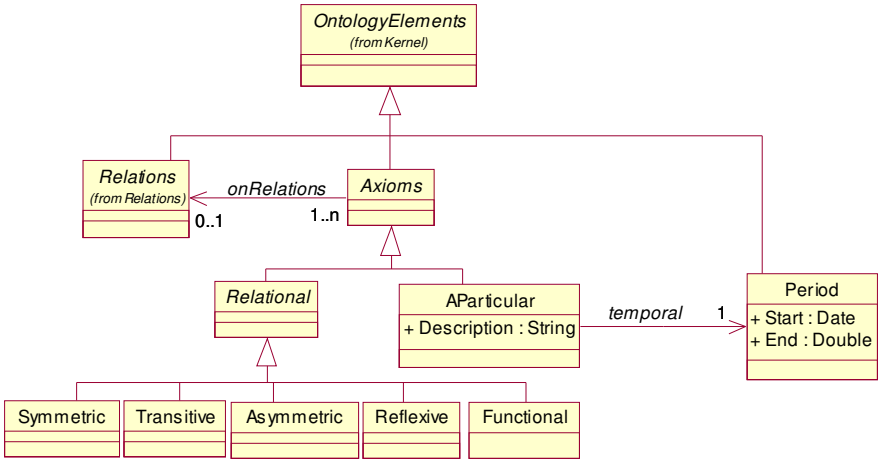

Figure 4. Elements defined in the Axioms Package.

2.5 Context Package

This package contains classes and associations that can be used to describe a context. In the area of AI there is a large amount of discussion about context [10][3]. From B2B perspective, a context can be defined by a collection of relevant assumptions that make a situation unique and composed by the real content. The assumptions are the characteristics or attributes that define the context, i.e., a description of a context; and the real content is the ontology. Following, we present the definition of a context and then the class diagrams that represent a context.

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, D_j, $O_{i,j}$>, where c is the unique identifier of the context j, D_j is a set of assumptions about context j and O_i represents the ontology i within the context j.

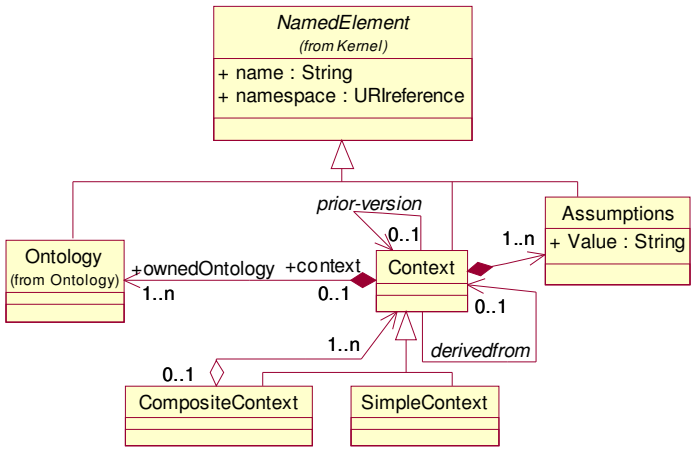

Figure 5. Elements defined in the Context Package.

In the class diagram of Figure 5, we associate the *Context* class with one or more *Ontology* classes and with one or more *Assumptions* classes. In [1] the following components are defined as assumptions: the owner of the context, the group in which the context has been developed, the security information and information on how a context was generated. But, we preferred to define the class in general to allow a user to define their own *Assumptions*. In addition, a context could be derived from other context and this is modeled by the *derivedfrom* association.

Furthermore, a context could be a simple one or a complex one. That is, a context could be formed by other contexts. For example, the context of supplier could be composed by different contexts that represent different domains that compose an enterprise, such as Forecasting, Planning, Scheduling, Product Design, and so on.

2.6 Context Mapping Package

This package contains classes and associations that can be used to define the mappings between terms that belong to different contexts. A *context mapping* allows us to state that a certain property holds between elements defined in different contexts. A context mapping is defined by *bridge rules* as linking rules between contexts. Following we define context mappings 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 bridges rules that map an element from source context to elements of the target context.

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

Definition 3. A bridge rule *br* 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 the relation between elements.

A bridge rule from context *s* to context *t* is a statement of one of the following forms, where e_i and e_j are elements of context c_s and c_t respectively.

$c_s : e_i \xrightarrow{\equiv} c_t : e_j$ $c_s : e_i \xrightarrow{\perp} c_t : e_j$ $c_s : e_i \xrightarrow{*} c_t : e_j$ $c_s : e_i \xrightarrow{\subseteq} c_t : e_j$ $c_s : e_i \xrightarrow{\supseteq} c_t : e_j$

(1) (2) (3) (4) (5)

Rule (1) means that e_i is similar to e_j. For example, Forecasting:Item $\xrightarrow{\equiv}$ Scheduling:Item.

Rule (2) means that both elements are disjointed. For example, Forecasting:Forecast $\xrightarrow{\perp}$ Scheduling:Employee.

Rule (3) means that e_i and e_j are compatible elements. For example, Forecasting:Forecast $\xrightarrow{*}$ Scheduling:Schedule (an Schedule derives from a Forecast).

Rule (4) means that e_i is less general than e_j and

Rule (5) means that e_i is more general than e_j. For example, Forecasting:Bucket $\xrightarrow{\supseteq}$ Schedudulling:Date (bucket: valid forecast time period).

Finally, we have to define the container of all the above defined elements. This container is the domain space concept. That is, the contexts and the rules that relate the elements between context and context mapping are contained in a domain, which in our example is the B2B collaborative domain.

Definition 4. A domain space is defined as a duple that contains all contexts C_j $\forall j \in J$ and the set of mappings $M_{s,t}$ $\forall s, t \in J \wedge s \neq t$.

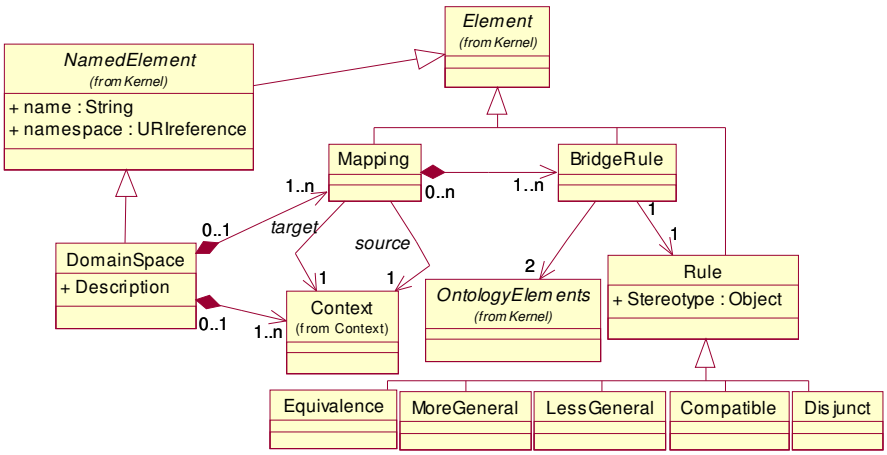

Figure 6. Elements defined in the Context Mapping Package.

Figure 6 represents classes and associations to model the context mapping, bridge rules and domain space. The *DomainSpace* class is associated with one or more *Context* and *Mapping* classes. A *BrigdeRule* class associated with *Rule* class could be one of the five rules defined above.

3 A Case Study

The main characteristic of a collaborative B2B relationship is that a customer and a supplier define business processes to be jointly executed, such as Forecasting and Scheduling Processes, and business information within them, such as, product requirement information.

In order to integrate this information to their own information systems, both parties have to know what the information exactly means. For example, the meaning of the "requirement information" depends on the business collaborative process. That is, at Forecasting Process level "requirement information" means a demand forecast sent from a customer to a supplier. At Scheduling Process level, "requirement information" means a supply schedule sent from a supplier to a customer. To inform this information between a customer and a supplier the structure of the business document could be the same, i.e. as shown in Figure 7; but the semantics changes.

```
1) <xs:complexType name="ReplenishmentOrder">
2)     <xsd:simpleType name = "Description">
3)       <xsd:restriction base = "xsd:string">
4)             <xsd:maxLength value = "40"/>
5)       </xsd:restriction>
6)     </xsd:simpleType>
7)     <xsd:element name = "Item" type = "Items"/>
8)       <xs:annotation>
9)     <xs:documentation> Customer's   actual requested amount of the item  </xs:documentation>
11)      </xs:annotation>
12)   </xs:element>
13)   <xsd:complexType name = "LocalAddress">
14)       <xsd:complexContent>
15)       <xsd:extension base = "Address">
16)          <xsd:element name= "zip" type = "xsd:string"/>
17)       </xsd:extension>
18)       </xsd:complexContent>
19)   </xsd:complexType>
20) <xs:complexType/>
```

Figure 7. A fragment of the business document: Replenishment Order

If business analysts do not define in an unambiguous way the semantics associated to the information, a problem arises when the software engineer wants to integrate business documents information to its own information system [5]. Furthermore, the information stored in trading partners information systems have to be mapped into business documents.

First of all, trading partners have to define the context and the characteristics that make the context unique. For example, they could define the "Forecasting" context and decide the schemas of business documents that will be interchanged during this collaborative process.

Following we present some rules to model terms belonging to a XML-based business document:

1. The *xsd:complexType* element has to be modeled by the class *Complex*. For example, from the document defined in Figure 7, line 1 and line 13, we can model **ReplenishmentOrder** and **LocalAddress** terms as complex terms.
2. The *xsd:simpleType* element has to be modeled by the class *Simple*. For example, from the document presented in Figure 7, line 2, we can model **Description** element as a simple term.
3. Then the elements defined by *xsd:element* tag could be simple or complex depending on its type definition. That is, if they are defined as a base xsd type, they

are simple terms. For example, on line 16 the **zip** element is defined as *xsd:string*. So, this element has to be modelled as a simple term. Then, on line 7, the **Item** element is defined as *Items*. In order to determine whether it is a simple or complex term we have to analyse if *Items* has been defined as a simple or complex term. This definition is not in this document. So, we suppose that *Items* was defined as a complex term.

4. Furthermore, the xsd:restriction element has to be modelled by *Properties* class. For example, on line 3 the **Description** element is restricted by using *xsd:restriction* definition so this characteristic has to be modelled as a property of the **Description** term.

In addition, the relationships between concepts can be derived from an XML document as follows:

1. The *xsd:extension* element represents the *is-a* relationship between terms. For example, in Figure 7 line 15, the <xsd:extension base = "**Address**"> definition states that the previously defined element (**LocalAddress**) *is-a* **Address**.
2. The combination of *complexType* and *element* primitives represents the *part-of* relationship between terms. For example, all elements defined between the *complexType* primitives that define the **ReplenishmentOrder** term are related to it by the *part-of* relation. That is, **Date**, **Description**, **Items** and **LocalAddress** are *part-of* **ReplenishmentOrder**.
3. The *element* primitive represents the *Inst-of* relation between terms.

Figure 8 presents the sintactic and semantics models of the terms belonging to the XML-based document shown in Figure 7. The semantics model was obtained by applying the rules defined above. The notation used to graphically represent this model is based on the UML notation by using UML stereotypes. However, we are working on the definition of a graphical notation for the metamodel presented in this paper. The tool, called VCODe Tool, is based on agent technology [4] and it is still in progress.

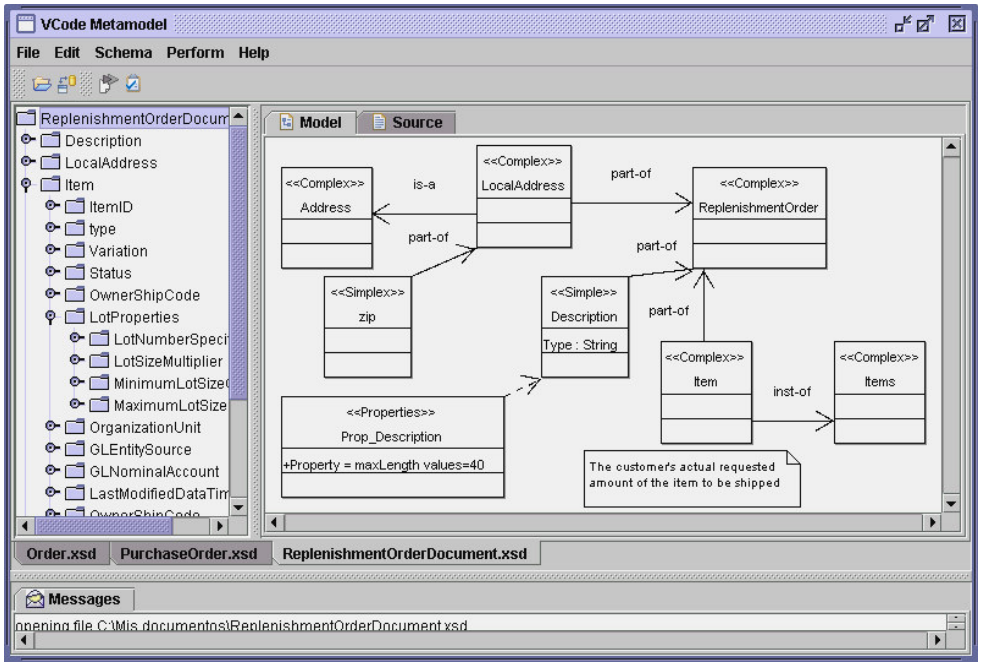

Fig. 8. Sintactic and semantics models of an XML-based document.

4 Conclusions and future work

An effective collaborative B2B relationship requires the right modeling of collaborative processes and each message within these processes. Each message contains business information that may be defined by a vocabulary that is shared by the parties engaged in the B2B relationship. XML (eXtensible Markup Language) is becoming widely used for defining specifications that define both the process and the business documents. However, these specifications are not easy to implement due to the heterogeneity problem at semantic level because XML does not express semantics by itself.

In order to fill the gap between people involved in the business documents definition and ontology specification languages we have defined a Contextual Ontology Definition Meta-model. This metamodel allows business analysts to explicitly model the contextual ontologies. Ontologies and contexts have to be integrated in order to solve the semantic heterogeneity problem and this is the main characteristic of the proposed metamodel. On the one hand, ontologies define concepts and relationships between them. On the other hand, contexts are useful tools to model concepts since the latter are true or false according to their contexts.

Furthermore, in this paper we present an approach to automatically generate an ontology model from a XML-based document. This approach is being implementing in a visual tool.

The future directions will be focus on the mapping between the contextual ontology definition language presented in this paper and one contextual ontology specification language.

REFERENCES

[1] Bouquet, P, Dona, A, Serafini, L and Zanobini, S. Contextualized local ontologies specification via CTXML. AAAI-02 Workshop on Meaning Negotiation (MeaN-02) July 28, 2002, Edmonton, Alberta, Canada.

[2] Bouquet, P; Giunchiglia, F.; van Hamerlen, F.; Serafini, L and Stuckenshmidt. C-OWL: Contextualizing ontologies. In the Proceeding of the Second International Semantic Web Conference, 2003. 164-179.

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

[4] Caliusco, Ma. L.; Maidana, C.; Chiotti, O. and Galli, Ma. R. Propuesta de un Sistema Multiagente para Asistir al Modelado de Documentos de Negocio. Accepted to be presented in Argentine Symposium on Information Systems (ASIS 2004). September 20-24. Argentina.

[5] Caliusco, Ma. Laura, Galli, Ma. Rosa and Chiotti, Omar. Ontology and XML-based specifications for collaborative B2B relationships. In Proceeding of III Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería de Conocimiento (JIISIC). Valdivia (Chile). November 2003.

[6] 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.

[7] Cranefield, S., Haustein, S. and Purvis, M.. UML as an ontology modelling language. In Proceedings of the Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents, Montreal, Canada, 2001.

[8] Duric, D.; Gasevic, D; Devedzic, V. A MDA-based Approach to the Ontology Definition Metamodel. Proceedings of a 4TH Workshop On Computational Intelligence And Information Technologies. October 13, 2003, Faculty of Electronics, Niš, Serbia.

[9] Farquhar A, Fikes R, Rice J (1997) "The Ontolingua Server: A Tool for Collaborative Ontology Construction". International Journal of Human Computer Studies 46(6):707-727

[10] Giunchiglia, F. and Bouquet, P. "Introduction to contextual reasoning. An Artificial Intelligence Perspective", in B. Kokinov (ed.), Perspectives on Cognitive Science, 3, NBU Press, Sofia (Bulgaria) 1997. http://dit.unitn.it/~bouquet/pers-publ-engl.html

[11] Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M., Smith, J. UML for Ontology Development, Knowledge Engineering Review Journal Special Issue on Ontologies in Agent Systems, 2002 Vol. 17.

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

[13] Mellor, S; Scott, K; Uhl, A. and Weise, D. MDA Distelled – Principles of Model-Driven Architecture. ADDISON-WESLEY, 2004.

[14] Ontology Definition Metamodel (ODM). Initial Submission to OMG. August, 2003.

[15] Staab, S; Mädche, A.. Axioms are objects, too - Ontology Engineering Beyond the Modeling of Concepts and Relations. In: V.R. Benjamins, A. Gomez-Perez, N. Guarino (eds.). Proceedings of the ECAI 2000 Workshop on Ontologies and Problem-Solving Methods. Berlin, August 21-22, 2000.

[16] UML 2.0 Infrastructure – Final Adopted Specification. September, 2003.