

**Equivalence of UML semantic data models and
RDF content models**

January 2003

IST-2001-32429

ICONS

Intelligent Content Management System

www.icons.rodan.pl

Project Partners

Rodan Systems (PL)

The Polish Academy of Sciences (PL)

Centro di Ingegneria Economica e Sociale (IT)

InfoVide (PL)

SchlumbergerSema (BE)

University Paris 9 Dauphine (FR)

University of Ulster (UK)



Equivalence of UML semantic data models and RDF content models

Project name	Intelligent Content Management System
Acronym	ICONS
Workpackage	WP2
Task	T2.2
Document type	report
Title	Equivalence of UML semantic data models and RDF content models
Subtitle	
Document acronym	D08
Author(s)	Yaxin Bi, David Bell, Hui Wang, Kieran Greer, Guongde Guo
Reviewer(s)	Pasquale Rullo, Robert Kowalczyk, Bartosz Nowicki
Accepting	Witold Staniszkis
Location	ICONS WP2 T2 D08 0104.doc
Version	1.05
Date	January 2003
Status	final version
Distribution	public

January 2003

History of changes

Date	Version	Author	Change description
10.01.2003	1.03	Yaxin Bi, David Bell, Hui Wang, Kieran Greer, Guongde Guo	Reformat and made some corrections
10.01.2003	1.0	Yaxin Bi, David Bell, Hui Wang, Kieran Greer, Guongde Guo	Revised version based on the Bartosz Nowicki, Robert Kowalczyk and Pasquale Rullo comments
2.01.2003	0.8	Yaxin Bi, David Bell, Hui Wang, Kieran Greer, Guongde Guo	The first draft, adding more content based on Dr. Witold Staniskzi's comments.
15.11.2002	0.5	Yaxin Bi, David Bell, Hui Wang, Kieran Greer, Guongde Guo	Revised some sections according to the comments from Annette Bleeker
30.09.02	0.2	Yaxin Bi, David Bell, Hui Wang, Kieran Greer, Guongde Guo	Revised extended abstract, and it has been transferred to the template format
1.1.2002	1.01	Yaxin Bi, David Bell, Hui Wang, Kieran Greer, Guongde Guo	document creation

Executive summary

This report provides an overview of developing transformation from object models in UML to XML DTD / Schema and RDF models. Starting with defining potential content objects to be stored in the ICONS content base, we investigate approaches for representing content objects in UML, equivalence of UML semantic data models and RDF content models, and approaches of mapping UML to RDF. In this report, we also give the description of the state of the art of research initiatives in this aspect of work in some details, freeware software tools and commercial products in supporting the functionality of converting UML models to XML DTDs / Schemas and RDFS models. Finally we identify some outstanding issues and suggest the ways to deal with these issues.

Table of contents

History of changes.....	3
Executive summary	4
Table of contents	5
List of figures	6
List of tables	6
1. Introduction	7
1.1 Objectives	7
1.2 Scope	7
1.3 Relations to Other Documents.....	7
1.4 Intended Audience.....	7
1.5 Usage Guidelines.....	7
1.6 Notation Conventions.....	8
2. Preliminary – UML, DTD, Schema and RDF	9
2.1 UML	9
2.2 DTD.....	9
2.3 XML Schema.....	9
2.4 RDF	10
3. Content model	12
3.1 Relational data model.....	12
3.2 Object model	12
3.3 Mapping relational models to object models	13
4. From object models to XML DTDs.....	16
5. From XML DTDs to XML schemas	17
6. Ontology and RDF	22
7. Approaches to mapping UML to RDF	23
8. Summary	26
Bibliography.....	27
Dictionary.....	28

List of figures

Figure 2.2: A DTD definition	9
Figure 2.3: An XML Schema	10
Figure 2.4: A RDF example	10
Figure 2.5: A RDF example with syntax	11
Figure 2.6: Dublin Core in RDF format	11
Figure 3.1: The relationship of the relational tables in the ER diagrams.....	14
Figure 3.2: The representation of the relation tables in the UML diagrams.....	14
Figure 3.3: The representation of the relation tables in the UML diagrams.....	15
Figure 4.1: A DTD definition corresponding to the class definition in Figure 3.1	16
Figure 5.1: The XML Schema of the relational table Orders	17
Figure 5.2: The extension of Figure 5.1	17
Figure 5.3: The Schema generated by EditML.....	18
Figure 5.4: An UML diagram.....	20
Figure 5.5: A fragment of the corresponding XML schema.....	21
Figure 6.1: A fragment of the corresponding XML schema.....	22

List of tables

Table 3-1: correspondence between relational model and object model	13
Table 3-2: Relational tables.....	14
Table 5-1: The proposed extension of stereotypes	19
Table 7-1: Mapping from UML to RDF.....	24
Table 7-2: Mapping from UML to RDF.....	24
Table 8-1: Feature summary of the current products.....	26

1. Introduction

Task 2.2 is a constituent of Work Package 2, part of the multi-paradigm integrated knowledge schema. The objective of the task is aiming at investigating the equivalence of UML (Unified Modeling Language) semantic data model and the RDF (Resource Description of Framework) content model. The research effort will lead approaches to mapping object models in UML to XML DTD (Document Type Definition) / Schema complex content data models and mapping UML semantic data models (SDM) to RDF object relationship models. These will be used in defining the knowledge bases that consist of principal parts – the content base comprising repositories of diverse data sources such as relational data, multimedia information objects and ontology base representing formal knowledge pertaining to the corresponding to application domains – data sources. In this report, we clearly make distinction between the representation of content models and ontologies in UML, this is not only for considering practical applications in the ICONS context, to be necessary to distinguish two fundamental concepts, but also making our discussion more precise.

The progress of this task presents four aspects. The first is to examine a content data model, mainly in an object-oriented notation and how it can be specified in UML. The second aspect is to investigate feasible methodologies for translating the object models to be associated with data sources in UML to XML DTD / Schemas complex content model. The third aspect is to develop domain ontologies based on the RDF metadata that describe the content base semantics, and then to examine how the ontologies can be represented in UML and mapped to RDF objects. We also need to identify an approach to complement the ontology base since the semantics derived from the Content Base may not be sufficient to achieve semantic interoperability required by the ICONS prototype system.

In the next sections, we give a brief overview of UML and XML DTD / Schema, together with some descriptions of the RDF, and then describe different types of mapping rules involved in mappings from relational models to object models, from XML DTDs to Schemas, and so forth. In particular, these are focused in the context of knowledge representation. We also present more details about the specific description of the content model for storing meta-information associated with heterogeneous data sources in the Content Base, the ontology roles in ensuring the consistent use of terminology in the content model in order to cope with semantic heterogeneity reflected in the integration of heterogeneous data sources, and the mapping rules from the UML model to RDF.

1.1 Objectives

The major objective is aiming at investigating the equivalence of UML semantic data model and the RDF content model.

1.2 Scope

This report is closely related to deliverables [ICONS D06] which describe knowledge representation structures. Some new information and suggestions however are included. This deliverable is also of relevance to the ICONS architecture [ICONS D16].

1.3 Relations to Other Documents

The deliverable is not self-contained, for full understanding, it may be necessary to refer to [ICONS D01, ICONS D06].

1.4 Intended Audience

The intended audience includes all members of the ICONS project consortium as well as the representatives of the European Commission monitoring and evaluating the progress of the project research and development work.

1.5 Usage Guidelines

This deliverable is a suggestion for the equivalence of UML semantic data models and RDF content models, and approaches of mapping UML to RDF which is relevant to the structure of the multi-paradigm integrated knowledge schema. Knowledge of the other deliverables listed above would be useful when reading this.

1.6 Notation Conventions

None.

2. Preliminary – UML, DTD, Schema and RDF

2.1 UML

UML is becoming widely used for software modelling, and the current version was adopted by the Object Management Group (OMG) as a standard language for object-oriented analysis and design. UML includes diagrams for use cases, static structures (class and object diagrams), behaviour (state-chart, activity, sequence and collaborations diagrams) and implementation (component and deployment diagrams). For data modelling purpose, UML uses class diagrams, to which constraints in a textual description may be added. Although class diagrams include implementation details (e.g. navigation and visibility indicators), it is possible to use them for analysis by omitting such as details. When used in this way, class diagrams essentially provide an extended Entity Relationship (ER) notation. Therefore the use of UML to design data models for databases is rapidly growing in the recent years.

2.2 DTD

A DTD defines element types and a syntax like a relational schema. It provides a logic structure for XML documents, and restrictions that how elements can be related to each other. More specifically, a DTD is composed of element types and sub-element types, attributes, terminal strings such as ENTITY, PCDATA, and CDATA, along with constraints on element and sub-element types, including '*': set with zero or more elements, '+': set with one or more elements, '?': optional, and '|': or. There is a special attribute, ID, which can occur once for each element. The ID attribute uniquely identifies an element within a document and can be referenced through an IDREF field in another element. IDREFs are untyped. Finally, there is no concept of a root of a DTD – an XML document conforming to a DTD can be rooted at any element defined in the DTD if a root element in DOCTYPE is not specified. Figure 2.1 present a simple DTD.

```
<!ELEMENT address      (name, place) >
<!ELEMENT place        (street, town) >
<!ELEMENT name         (#PCDATA) >
<!ELEMENT street       (#PCDATA) >
<!ELEMENT town         (#PCDATA) >
```

Figure 2.1: A DTD definition

2.3 XML Schema

An XML Schema provides a powerful means by which to define element types and grammars for XML documents. Itself is in form of a well-formed XML document. Being compared to DTDs, XML Schemas provide an object oriented approach to defining the format of XML documents. The underlying difference between XML-schemas and DTDs is that the former provide a set of basic types that are much wider ranging than the types of PCDATA and CDATA defined in the latter. XML-Schemas include most basic programming types such as integer, byte, string and floating point numbers, but they also expand into Internet data types such as ISO country and language codes. Majority of these types cannot be defined with DTDs. Due to the limitations of the simple structuring and typing mechanisms in DTDs, numerous XML validation, structuring, and typing systems have been created in XML Schemas. Figure 2.2 illustrates a simple example of XML schema.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema>
<xs:element name="Address" type="AddressType"/>
  <xs:complexType type="AddressType ">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="street" type="xs:string"/>
      <xs:element name="town" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element></xs:schema>
```

Figure 2.2: An XML Schema

2.4 RDF

RDF is a model for representing metadata, and one possible syntax (using XML) for expressing it. At the core of RDF are *nodes*, and attached *attribute/value pairs*. Nodes can be any web resources or other instances of metadata. Attributes are named as properties of the nodes, and their values are either atomic or other resources or metadata instances. These often are modelled as triplets of subject, predicate and object. The instances of RDF data models, the RDF Schemas, are roughly comparable with ER diagrams, and two RDF expressions are equivalent if and only if their data model representations are the same. RDF in itself does not contain any predefined vocabularies for authoring metadata, but its point of view is broad enough to be applicable to most (if not all) metadata information. A RDF model can be represented in a graph and also be formulised in a well-defined RDF syntax.

As a simple example, let us see how to model the sentences "The individual referred to by employee id 85740 is named Ora Lassila and has the email address lassila@w3.org. The resource <http://www.w3.org/Home/Lassila> was created by this individual" in a RDF graph model:

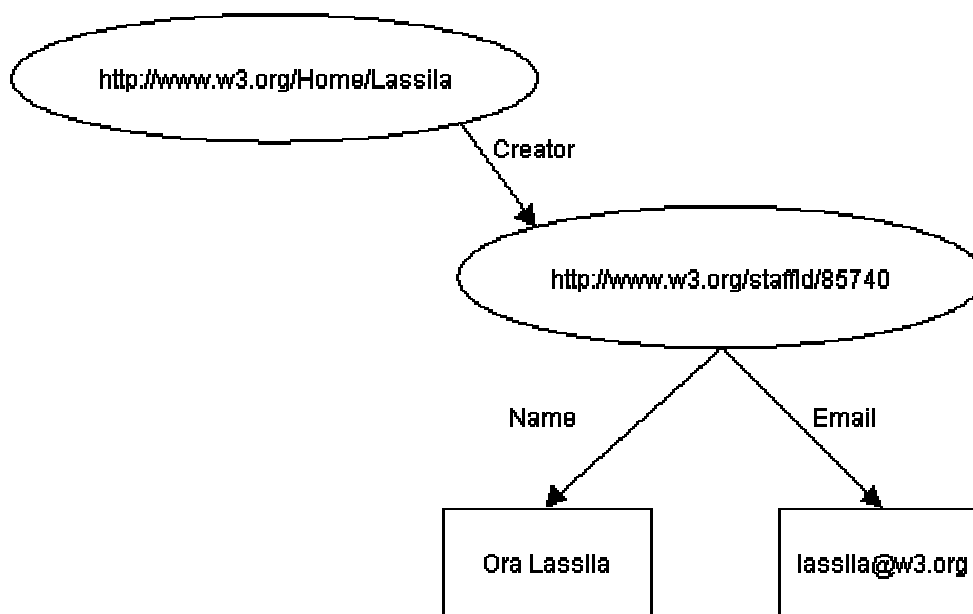


Figure 2.3: A RDF example

The corresponding triple of the first sentence is as follows:

Subject (Resource)	http://www.w3.org/Home/Lassila
Predicate (Property)	Creator
Object (literal)	"Ora Lassila"

and the above graph can be expressed with the RDF syntax below:

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator rdf:resource="http://www.w3.org/staffId/85740"/>
  </rdf:Description>

  <rdf:Description about="http://www.w3.org/staffId/85740">
    <v:Name>Ora Lassila</v:Name>
    <v:Email>lassila@w3.org</v:Email>
  </rdf:Description>
</rdf:RDF>
```

Figure 2.4: A RDF example with syntax

Here is another example that illustrates how a bibliographical metadata – Dublin Core is represented with the RDF syntax. The following Dublin Core properties are used: "Title", "Description", "Publisher", "Date", "Subject", "Type", "Format", and "Language".

```
<rdf:RDF
  xmlns:rdf="http://w3.org/TR/1999/PR-rdf-syntax-19990105#"
  xmlns:dc="http://purl.org/metadata/dublin_core#">
  <rdf:Description about="http://www.dlib.org"
    <dc:Title> D-Lib Program - Research in Digital Libraries</dc:Title>
    <dc:Description> The D-Lib program supports the community of people
      with research interests in digital libraries and electronic
      publishing.</dc:Description>
    <dc:Publisher> Corporation For National Research
      Initiatives</dc:Publisher>
    <dc>Date>1995-01-07</dc>Date>
    <dc:Subject>
      <rdf:Bag>
        <rdf:li>Research; statistical methods</rdf:li>
        <rdf:li>Education, research, related topics</rdf:li>
        <rdf:li>Library use Studies</rdf:li>
      </rdf:Bag>
    </dc:Subject>
    <dc>Type>World Wide Web Home Page</dc>Type>
    <dc:Format>text/html</dc:Format>
    <dc:Language>en</dc:Language>
  </rdf:Description>
</rdf:RDF>
```

Figure 2.5: Dublin Core in RDF format

3. Content model

In ICONS, the Content Base holds partial definitions of the content object models, content object methods, definitions of the content object relationships, as well as the content object taxonomies [ICONS D01]. However, the relation of these definitions to concrete data sources has not been well defined in the ICONS reports. In general, the relevant content may include multimedia information, text documents, relational data, and Web-based data repositories, etc. These rich internal and external data sources and a variety of access methods could be abstractly modelled as content objects and inference methods of triples denoted by $\langle R, M, F \rangle$ [ICONS D01], and specified in UML. Each model defined in UML are suggested to be converted as XML DTDs / Schemas which will be associated with these data sources.

To simplify our discussion, we assume relational databases to be the major data sources of the ICONS prototype system, and for the purpose of data source discovery and publishing point, these data sources may be represented in some standard metadata such as Dublin Core in RDF and they will be published in the Web-based repositories.

To begin with modelling external data sources in UML, it will be necessary to consider translation between relational and object models, and how transformation between two models being carried out since UML is an object-oriented approach and particularly suited to modelling data source as a range of objects.

Considerable research work exists in developing transformation between object and relational models [Fussell 2002, ORMMapping 2002], which are mainly carried out in database research area and object-oriented programming community. An object-relational modelling is concerned with methods and techniques of transforming object to relational models. It can be regarded as a way for doing the reverse translation, but such an approach may not be the best one in effectively fitting to the translation scenario from relational models to object oriented models as required in the ICONS project. The reason for this is that mapping rules from object to relational models defined is a set of single and tight-coupled constraint, much changing needs to be made in order to generalize these rules to another case. To deal with this translation issue in the ICONS context, there needs to investigate a possible methodology for mapping relational models to object models and this methodology will form a ground towards examining equivalence between UML / XML DTD / Schema.

In fact, relational modelling and object modelling have such different concerns that they are extremely compatible. Relational modelling is concerned with modelling entities into structured data and their relationships, while object techniques are concerned with (primarily) entities' attributes and behavior along with their structures. Mapping between the two models requires deciding how the two models can refer to each other. We will first describe briefly the characteristics of two types of the models and then explore how relational models can be transformed to object oriented models.

3.1 Relational data model

A relational database storing data may consist of one or more tables with rows and columns. The rows correspond to records (tuples); the columns correspond to attributes (fields in the record). Each column has a data type (e.g., date). The types of data that can be stored are confined to a very limited (six or so) number of data types -- for example, character, string, time, date, numbers (fixed and floating point), and currency. Any attribute (field) of a record can store a single value or a set of values. Variable length fields may be not supported. Relationships are not explicit, but rather implied by values in specific fields, foreign keys in one table (e.g., departments) that match those of records in a second table (e.g., employees). Many-to-many relationships typically require an intermediate table that contains just the relationships.

3.2 Object model

Object modelling describes systems as built out of objects: programming abstractions that have identity, behavior, and state. Objects are an abstraction beyond abstract data type, where data and variables are merged into a single unifying concept. As such object modelling includes many other concepts: abstraction, similarity, encapsulation, inheritance, modularity, and so on. For sake of simplicity, we are only concerned the basic object concepts of identity, association and class.

3.3 Mapping relational models to object models

Relational-object mapping can be treated as a process of transforming between relational and object models. The task of relational-object mapping requires a solid understanding of object modelling and relational modelling, how they are similar, and how they are different as described above. Table 3.1 gives a correspondence between the different concepts from the two types of the models.

Relational concepts	Object concepts
Attribute	Attribute
Tuple	Instance
Relational table	Class
Key	Identity
Foreign key	*
One-many relationship	Association
*	Inheritance

Table 3.1: correspondence between relational model and object model

Based on the above correspondences, we define a mapping process for translating relational tables to objects as follows:

- Mapping relational table definitions to object classes. The table definitions map directly attribute definitions in classes except for foreign keys. These keys can be replaced with relationships.
- Mapping inheritance based on a table or multiple tables. There may be type codes in a given table used to identify types of entities, such as types of Orders or types of Items. Those types can be turned into an inheritance structure at the object application level. Similarly, there may be separate tables for different types of entities. Again, this can be turned into an inheritance structure.
- Mapping tuple retrieval, keys and relational joins to relationships for object navigation. Object navigation replaces successive searches based on keys.
- Mapping "intersection tables" to object relationships. An example of an intersection table is the "OrderNum ItemsNum" as shown in Table 3.2 (d). This table exists because the relational model does not support many-to-many relationships. The relationship between "Order" and "Items" is a many-to-many relationship because one Order can include more or one item and one item can appear in more or one Order as demonstrated in Table 3.2.

Difficulties may occur when we deal with the real data sources implementation and relational-object model mapping using the above approach. These data sources may have individual implementations that are incomplete or inconsistent with each other. The reason for this is not abnormal because in most situations external data sources may be heterogeneous which were developed by different organizations. Although they might follow certain standards when they were developed, such standards might have been customized according to their specific requirements. Therefore it is difficult to develop a generic approach which can cope with a range of variations of standards to satisfy all of the requirements. On the other hand, object modelling in UML is not entirely standardized, so each programming environment may implement its own variation. In such a sense, it may be necessary to investigate some common aspects that would be useful in establishing equivalence between UML and XML DTD / schema, and can be applied to potential data sources in the ICONS project.

As an example, we take three simple relational tables as presented in Table 3.2, which are assumed to represent major characteristics of external data sources, to see how they can be represented in an object-oriented model and how they can be mapped to XML DTD / Schema.

Orders (a)

OrdersNum	Date	CustNum

Items (b)

OrdersNum	ItemsNum	Quantity	Parts

Parts (c)

PartsNum	Price
-----------------	--------------

Orders-Items (d)

OrdersNum	ItemsNum
------------------	-----------------

Table 3.2: Relational tables

Looking at the preceding tables (a, b), the relationships between Orders and Items is a many-to-many relationship. Because relational database could support this kind of relationship, the relationship has to be normalized into an intersection table as shown in Table 3.2 (d) containing two one-many relationships as depicted in the following diagram. The relationship between Items and Parts can be treated as one-to-one relationship.

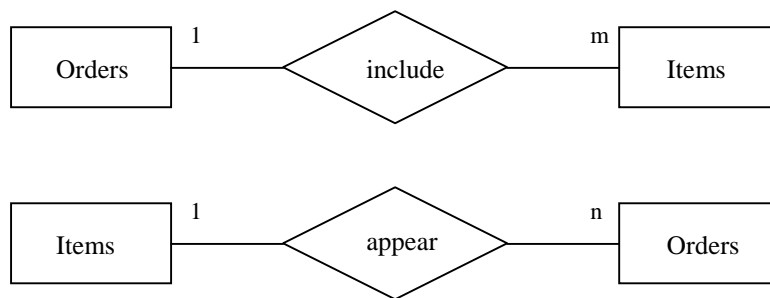


Figure 3.1: The relationship of the relational tables in the ER diagrams

The above three relational tables can be represented in UML below, it seems that the mapping between the two types of the models would be straightforward. In fact, as we see that some features embedded in relational tables such as relationships between the tables can not be depicted explicitly in the UML diagrams. Such relationships may be refereed as one kind of semantics in the ICONS context, but the current version of UML does not provide broader stereotypes to support a wide range of relationships that are reflected in the relational models. Thus the UML tool is a less ideal approach for representing complicate relational models. The recognition of this deficit has led to a proposal to extend the stereotypes as described in Table 5.1.

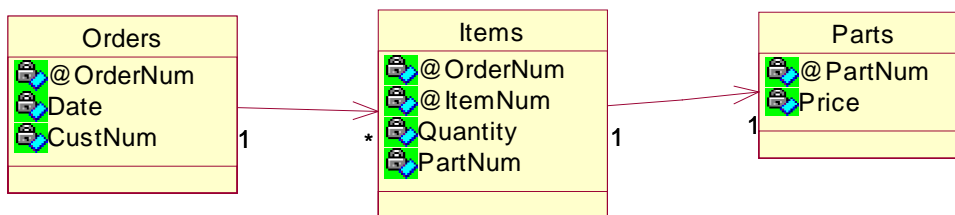


Figure 3.2: The representation of the relation tables in the UML diagrams

The above diagram gives a logical specification of the objects, which are corresponding to the three tables shown in Table 3.2. Based on this diagram, we can define the classes for each relational table in the Java convention below.

```
Class Orders {  
  String OrderNum;  
  String Date;  
  String CustNum;  
  Items [] I;  
}
```

```
Class Items {  
  String OrderNum;  
  String ItemNum;  
  Integer Qunatity;  
  Parts P;  
}
```

```
Class Parts {  
  String PartNum;  
  Double Price;  
}
```

Figure 3.3: The representation of the relation tables in the UML diagrams

This figure demonstrates that how the relationships can be defined in the classes along with mapping primary/foreign keys of relational tables to attributes of object models by the class reference mechanism. It only shows a one-way relation mapping of from Orders to Items. Such a relationship reflects a hierarchical relation between the classes even if no inheritance exists from Order to Items. However, the reverse relation may not be desirable in dealing with model mappings, we disregard it in the whole mapping processing.

4. From object models to XML DTDs

The above approach constitutes a basis ensuring the generation of object models from relational models. One particular benefit from this is to allow us to examine equivalence and implementation between alternative models for the same applications and to generate XML DTDs based on the object models in UML. In fact, these object models not only can be used to derive the equivalent DTD definitions for the external data sources, but also can be incorporated to generate XML schemas / RDF. Since XML DTDs and schemas are roughly compatible with respect to syntaxes for XML documents, our approach will focus on how the object models can be transformed into DTD representations, which preserve underlying structural characteristics of external data sources, and then compare XML DTDs and XML schemas, and possible translations. First let us look at some characteristics of DTDs.

Figure 3.2 and 3 depict explicitly the object models in UML and class notations. Now we use these models to translate to an equivalent DTD. The DTD can be generated by starting from a single class, and gradually including element types from the other classes. As an example, we see how the translation from the above object models to the DTD can be properly done.

In the first step, we generate an element type for the class of Orders and then generate PCDATA-only elements for the attributes (Date and CustNum) and add references to these elements to the content model of the Orders element. Thus this works out the following result.

```
<!ELEMENT Orders (Date, CustNum)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT CustNum (#PCDATA)>
```

Repeating the above procedure, we create a PCDATA-only element for the primary key (OrderNum) and add a reference to it to the content model:

```
<!ELEMENT Orders (Date, CustNum, OrderNum)>
<!ELEMENT OrderNum (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT CustNum (#PCDATA)>
```

and then add an element for the table (Items) to which the primary key is exported, as well as a reference to it in the content model:

```
<!ELEMENT Orders (Date, CustNum, OrderNum, Items*)>
<!ELEMENT OrderNum (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT CustNum (#PCDATA)>

<!ELEMENT Items()>
```

Figure 4.1: A DTD definition corresponding to the class definition in Figure 3.1

Following the same mapping process, we can construct the element type definition for the class **Items** as well.

In the above example, we describe a process of mapping from the object models to the DTD. This process demonstrates a feasible way to modelling external data sources as object models in UML, and then converting them into DTDs. More recently, there are a variety of activities in developing a range of tools from OMG and W3C communities to support extraction of DTDs and XML schemas from relational databases. These include XMLSpy [XMLSpy 2002], EditML [NetBryx 2002], Oracle 9i [Oracle 2002], etc. However, in the context of the ICONS project, our interest is related to the investigation of equivalence between UML and XML DTD / Schema / RDF, these techniques and methods can not fit in our requirements. Thus the approach which we are taking provides a roadmap to fulfil our designated objectives.

5. From XML DTDs to XML schemas

An XML Schema provides a powerful means by which to define element types and grammars for XML documents. We use these core types, along with various operators and modifiers, to create complex types of their own. In urn, these complex types can be used to define elements in XML Documents.

As a simple example, let us see an XML Schema for defining the Orders defined as the DTD as illustrated in Figure 4.1. Firstly, we must declare this as an XSD Document, and then add other constructs to it:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema>
<xs:element name="Orders" type="OrdersType"/>
  <xs:complexType type="OrdersType ">
    <xs:sequence>
      <xs:element name="OrderNum" type="xs:int"/>
      <xs:element name="Date" type="xs:string"/>
      <xs:element name="CustNum" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:element></xs:schema>
```

Figure 5.1: The XML Schema of the relational table Orders

As we see from this example, we define the element, Orders, which is equivalent to the element in the DTD as shown in Figure 4.1. In addition, we also specify its type as OrdersType, which is not a standard type, and so we have to provide a definition of that type.

We then define a complexType with OrdersType, in turn comprising a sequence of OrderNum, Date and CustNum. The "OrderNum" type is an xsd:int, "Date" type is an xsd:string, and "CustNum" is type of an xsd:int. These types have been defined by the XML Schema Namespace. Figure 5.1 the definition of all of elements in the relational table of Orders.

This example gives a simple definition for the relational table of Orders, it does not include any relationships with other relational tables such as Items and Parts, which are reflected in the DTD as the nesting relations between different the elements. Such relationships can be easily defined in an XML schema. We continue the above example to see how the Items element can be defined.

To add the relations into the Schema, we need to add the Items element: <xs:element name="Items " type="ItemsType" minOccurs="1 "/> as the sequence part as shown in Figure 5.2. There are two different constituents in this definition, ItemType and minOccurs = "1". The former specifies that the ItemType is not standard type of XML schemas, which needs to be further defined. The minOccurs means there must be at least one element at all times. As maxOccurs is not defined, there no upper limit to the number of elements that might be included. If we had specified neither, the default would be exactly one instance, as is used in the OrdersNum element. These actually are playing the same role as the constraints in DTDs. Next, we define the schema for the ItemType.

```
<xsd:complexType name="ItemType">
  <xsd:element name="OrderNum" type="xsd:int"/>
  <xsd:element name="ItemsNum" type="xsd:int"/>
  <xsd:element name="Quantity" type="xsd:int"/>
  <xsd:element name="Parts" type="PartsType" minOccurs="0"/>
</xsd:complexType>
```

Figure 5.2: The extension of Figure 5.1

This is all similar to the declaration of the OrdersType, but it notes that we have to re-define name element within the scope of this type.

As mentioned before, such an XML Schema can be generated by XML-aware tools. The following figure presents an XML schema generated by EditML Pro 2.0.

```
<?xml version="1.0" ?>
- <!-- Generated 11/01/2003 12:54:51 by EditML Pro v3.0-->
<=
  <xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:RecordSet" xmlns:nb="urn:RecordSet">
    <xsd:element name="RecordSet" type="nb:RecordSetType" minOccurs="1"
      maxOccurs="unbounded" />
    <= <xsd:complexType name="RecordSetType">
      <= <xsd:sequence>
        <xsd:element name="Record" type="nb:RecordType" minOccurs="1"
          maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:complexType>
    <= <xsd:complexType name="RecordType">
      <= <xsd:sequence>
        <xsd:element name="OrderNum" type="xsd:int" minOccurs="1"
          maxOccurs="unbounded" />
        <xsd:element name="Date" type="xsd:string" minOccurs="1"
          maxOccurs="unbounded" />
        <xsd:element name="CustomerNum" type="xsd:int" minOccurs="1"
          maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:schema>
```

Figure 5.3: The Schema generated by EditML

This example presents an XML schema for a single table of Orders, it does not include any relationships with the other relational table. Although the schema includes some overhead, basically the schema is the same as the schema defined in Figure 5.1.

As mentioned previously that the role of which DTDs plays basically is the same as that of XML schemas in defining grammars for XML documents, but XML schemas provide strong data typing, modularization and reuse mechanisms not available in XML DTDs. These result in the implementation of the automatic transformation from a DTD to XML schemas are more difficult than the reverse process of XML schemas to DTDs. Currently there are several working groups from W3C, OMG and enterprise organizations, working on the proposals for transforming DTDs to XML schemas and reverse process. Some also suggests completely to replace DTDs with XML schemas. Some conversion tools from DTD to XML Schema are available, such as DTD2XSD, Rational Rose 2.0, etc. [DTD2XSD 2001, Sumeet 2001]. These tools provide potential facilities for implementing possible transformation between UML and XML schemas by using the approaches for generating DTDs based on the object models in UML.

Obviously there is a direct way to transform UML models to XML Schemas. Some initial work has been presented in [Booch 1999, eBTWG 2001]. The idea of these work is to develop methodologies – production rules – that will be used in the whole transformation process of UML to XML schemas, also including DTDs. Booch et al. suggested to extend the current functionality of UML, in particular, to support more stereotypes for expressing XML schema constructs as defined in the recommendations of the XML schemas [Davidson 1999]. The following table lay out possible stereotypes to be added in the UML.

Stereotype	UML construct	Schema meaning
<<sox>>	Package	Indicates a full Schema
<<elementtype>>	Class	Element type definition
<<sequence>>	Nested class	Sequence group from a content model
<<choice>>	Nest class	Choice group from a content model

Stereotype	UML construct	Schema meaning
<<enumeration>>	Class may be nested	Enumeration datatype – can be UML Enumeration
<<scalar>>	Class – may be nested	Scalar datatype
<<varchar>>	Class- may be nested	Varchar datatype
<<implied>>	Attribute or Unidirectional association	Indicate an implied attribute
<<default>>	Attribute or Unidirectional association	Indicate a required attribute
<<fixed>>	Attribute or Unidirectional association	Indicate a fixed association
<<content>>	Attribute or aggregation	Indicates an atom in a content model

Table 5.1: The proposed extension of stereotypes

In the above table, there are essentially four new types of class stereotype:

- Element types. This includes the <<elementtype>> stereotype.
- Model groups. These are the <<sequence>> and <<choice>> stereotypes.
- Various datatype constructors corresponding to the datatype constructors found in XML Schema. These are the <<enumeration>>, <<scalar>> and <<varchar>> stereotypes.
- Stereotypes associate with XML attributes (<<implied>>, <<required>>, <<default>>, <<fixed>>) and content models (<<content>>).
- A <<sox>> stereotype to declare a Package to be a XML SCHEMA schema.

Some of these also apply to associations:

- The <<content>> stereotype applies to aggregation associations for parts of XML Schema content models.
- The XML attribute stereotypes can apply to a unidirectional association to delineate XML attributes.

The following figure presents a trivial example to shown UML to XML schema mapping. Figure 5.4 is an UML diagram and Figure 5.5 is the corresponding XML schema.

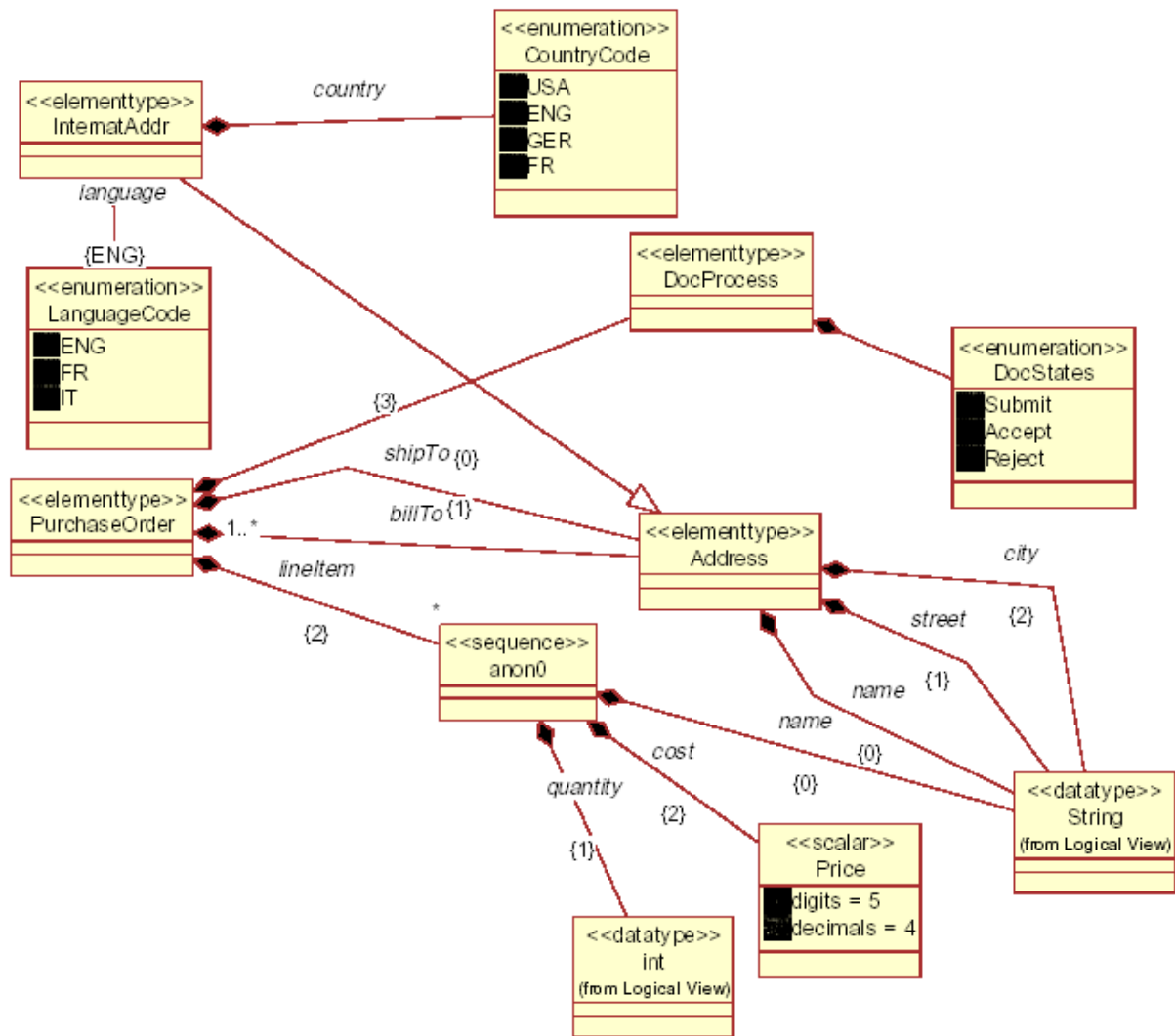


Figure 5.4: An UML diagram

```

<datatype name = "CountryCode">
  <enumeration datatype = "NMTOKEN">
    <option>USA</option>
    <option>ENG</option>
    <option>GBR</option>
    ...
  </enumeration>
</datatype>

<datatype name = "Price">
  <scalar digits = "5" decimals = "4"/>
</datatype>

<elementtype name = "DocProcess">
  <model>
    <string datatype = "DocStates"/>
  </model>
</elementtype>

<elementtype name = "Address">
  <model>
    <sequence>
      <element name = "name" type = "string"/>
      <element name = "quantity" type = "int"/>
      <element name = "cost" type = "Price"/>
    </sequence>
  </model>
</elementtype>

<elementtype name = "InternatAddress">
  <extends type = "address">
    <append>
      <element name = "country" type = "CountryCode"/>
    </append>
    <attrdef name = "language" type = "LanguageCode">
      <default>ENG</default>
    </attrdef>
  </extends>
</elementtype>

...

<elementtype name = "PurchaseOrder">
  <model>
    <sequence>
      <element name = "shipTo" type = "Address"/>
      <element name = "billTo" type = "Address"/>
      <sequence name = "lineItem" occurs = "+">
        <element name = "name" type = "string"/>
        <element name = "quantity" type = "int"/>
        <element name = "cost" type = "Price"/>
      </sequence>
    </sequence>
  </model>
</elementtype>

...

```

Figure 5.5: A fragment of the corresponding XML schema

The UML to XML Design Rules Project was approved as an activity under the UN/CEFACT Electronic Business Transition Working Group [eBTWG 2001]. The purpose of the UML2XML project is "to produce a set of formal syntax production rules, describing in a very detailed and strict way how to convert standardized business messages, which are defined in UMM-compliant UML class diagrams, into physical XML representations. The project has produced a technical specification describing the set of formal design rules needed to convert the UML class diagram representing a standardized business message into a physical XML representation. This technical specification will also include specifications for the UML representation of a standardized business message. For further details of proposed production rules can be found in [eBTWG 2001].

6. Ontology and RDF

Ontologies are critical for the integration of heterogeneous data sources. ICONS takes a widely accepted definition, that is, an ontology is defined as a shared formal conceptualization of a particular domain [Gruber 1993]. It can be used to specify what concepts represent and how they are related. In ICONS, most of terminology defined in the domain ontologies will derive from the Content Base, and the complementary terms may import from the classification servers or RDF repositories which are evolving on the Internet. With respect to the representation of the ontologies, XML DTDs and XML Schemas are sufficient for exchanging ontologies between data sources which have agreed to definitions beforehand. They fulfil the universal expressive power requirement because anything for which a grammar can be defined can be encoded in XML. It also fulfils the syntactic interoperability requirements because an XML parser can parse any XML data, and is usually a reusable component. However, XML has disadvantages in handling semantic interoperability [Berners-lee 2001]. We recommend that XML DTDs / Schemas would be only employed in modelling content models for the Content Base in ICONS.

Obviously, lack of semantics prevents machines from reliably performing exchanging tasks given new terminology. RDF and RDF Schema (RDFS) begin to approach this problem by allowing simple semantics to be associated with terms, defining mappings between terms within the data, which requires content analysis. With RDFS, one can define classes that may have multiple subclasses and super classes, and can define properties, which may have sub-properties, domains, and ranges, which are used to define either the closed taxonomies based on a specified list of categories, or open taxonomies based on an arbitrary value (s) of content object properties. In this sense, RDFS is a simple ontology language. However, in order to achieve interoperation between numerous, autonomously developed and managed schemas, richer semantic interoperability is needed [Heflin 2002].

Taking advantage of the existing work, it is suggested that ICONS would use RDF / RDFS as a principle way to represent the domain ontologies. In connection with the issue of mapping UML semantic data model to XML DTD / RDFS content and models, it is envisaged that the mapping from UML semantic data model to RDF/RDFS would be only applied to the construction of domain ontologies, which explicitly specify high level concepts, terminology, and semantics to be derived from content object repositories. The relationship between content objects and domain ontologies is depicted in Figure 6.1.

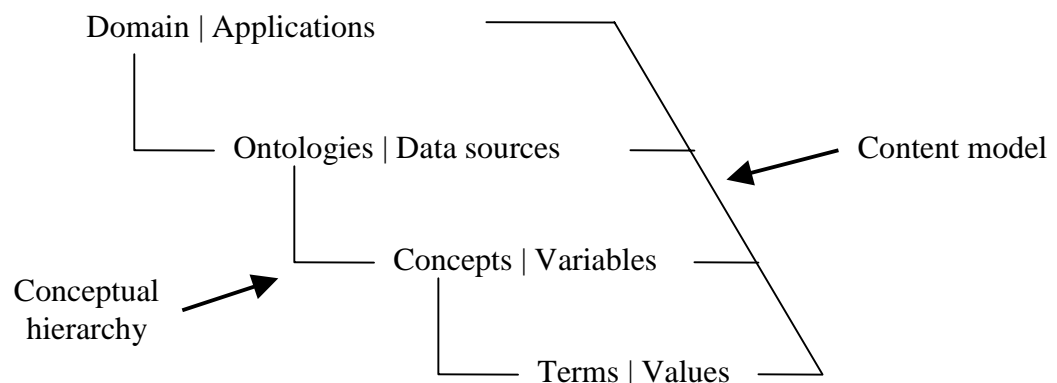


Figure 6.1: A fragment of the corresponding XML schema

7. Approaches to mapping UML to RDF

A variety of different research projects and commercial initiatives have been developing technologies for converting between UML and RDFS, and applying UML for ontology representation. The approaches taken in these efforts vary in a number of different ways, which are mainly carried out in OMG and W3C communities [Baclawski 2001]:

UML provides a round trip engineering between visual models and XML/DTD specifications – allows users to fully model, generate and reverse-engineer XML/DTDs.

UML is used for XML Schema mapping specification

UML is used directly as an ontology representation and as a graphical front-end for another ontology representation language of RDFS (DAML+OIL).

UML has been applied to a variety of ontology related tasks (e.g., ontology mapping and consistency checking).

The first two activities are mainly going on in the OMG community. In relation to the XML Schema mapping with UML, there exists much effort in the database research community, and in Web-based applications which are inspired by e-commerce as mentioned previously. In this section, we confine our interest to that of using UML to define the semantic data model for the Ontology Bases and these models can be translated to the corresponding RDF, as well as ensure the semantics will not be lost during conversion. Therefore, we will examine the last two, concentrating on what approaches have been developed in these areas and what issues are outstanding there.

Currently, there are two major systematic techniques for constructing mappings (transformations) between modeling languages UML and RDF – from visual model (graphs) to representation of textual description, and from visual model to Directed Labeled Graph (DLG) model. Baclawski et al. [Baclawski 2001] presents a representative work based on the first of these, a project of the UML Based Ontology Tool-set (UBOT) project, which is building ontology engineering tools and natural language processing-based text annotation tools for RDFS. In this work, they describe explicitly the reasoning deficiency in RDF, and propose to extend from RDF to a extended version RDFS, called DAML, and describe the correspondences between UML and DAML. They also identify weakness of UML in supporting transformations between the corresponding elements, e.g. the UML associations to RDFS elements. The approach proposed is to extend UML by defining a prototype UML profile for RDFS which maps UML stereotypes to RDFS-specific elements [Baclawski 2001]. The following table shows a partial comparison between the different approaches, which is considered as work in progress. A full comparison can be seen in [UBOT 2002].

UML	RDF
Class	Class (RDFS)
InstanceOf	Type (RDF)
Type of modelElement	type (RDF)
Attribute	Property (RDF)
Association	Property (RDF)
Generalization	subClassOf (RDFS)
Stereotyped dependency between 2 associations *	subPropertyOf (RDFS)
Generalization between stereotyped classes * (named DAMLproperty)	subPropertyOf (RDFS)
Note	comment (RDFS)
Name	Label (RDFS)
Tagged value on a class and association *	seeAlso (RDFS)
Tagged value on a class and association *	isDefinedBy (RDFS)
Attribute type: string	Literal (RDFS)
Attribute value value (RDF)	value (RDF)
Initial value for an attribute	Default
class containing the attribute	domain (RDFS)
Source class of an association	domain (RDFS)

UML	RDF
Attribute type (primitive or class)	range (RDFS)
Target class of an association (RDFS)	range (RDFS)

Table 7.1: Mapping from UML to RDF

The second approach for mapping UML to RDF is from visual model to DLG model [Note-RDF-UML 1998]. The idea of this approach is to map two types of different graphs, since most of classes, properties and semantics represented in UML can be equivalently represented using DLG model. If we use RDFS instead of RDF, the RDFS DLG can be shown to be isomorphic to a subgraph of the UML diagrams, because RDFS elements can map directly into UML class model elements. The table below presents a representative comparison of the models, showing that class schema representations using RDFS and the subset of UML class model are equivalent in some extent [NoteRDFUML1998].

Step	Description
1.	Show that RDFS and UML classes map between each other.
1a.	Show that single and multiple class inheritance constructs in each system map to each other.
1b.	Show that the class inheritance DAGs for RDFS and UML (using RDF-Schema subClassOf and UML subclass relationships respectively) are isomorphic.
1c.	Note that RDFS does not explicitly support the notion of operations or methods, We ignore this for now. When RDFS supports this, map these methods into UML operations.
2.	Show that RDFS properties and UML attributes can be directly mapped between each other. Extensions can be added later to support this if necessary.
3.	Show that any UML associations can be expressed as RDFS properties.
3a.	Show that UML multiplicity constraints on associations are equivalent to a subset of RDFS cardinality constraints
3b.	Show that property naming and reification in RDFS maps to UML association names and attributes.
4.	Show that while currently missing from RDFS, a generalized constraint model can be created by extending RDFS using reification allowing it to map into the general UML Class constraint model.
5.	Having accomplished 1,2, and the portion of 3 relevant to the RDFS, we can then show that the full RDFS DLG can map into a proper subgraph of the UML class model DLG.

Table 7.2: Mapping from UML to RDF

Although there are a number of ways to do translations between two significantly different approaches of UML and RDF, some crucial issues have been pointed out in distinct contexts and applications. The common issues are related to semantic representation and transformations in two different approaches since there are serious concerns about significant semantic mismatches between UML and RDF which result from the OMG and W3C approaches, leading to semantic interoperability being hard to achieve. Some of these significant mismatches are:

The OMG approach does not have a first-class concept of an "association" (analogous to a "property" in W3C terminology) [Baclawski 2001]. Associations in UML can only exist in the context of two or more classes. Properties in RDF and DAML are first-class elements that can be defined in an ontology without reference to classes. If an RDF schema or DAML ontology states that "company owns vehicle" and "person owns dog", "owns" is the same property whereas they would be different associations in a UML model.

The W3C approach does not have a clean layered architecture. This leads to confusing situations like a class being an instance of another class [why? Both rdf:type (for instances) and rdf:subClassOf exist.. Or do you mean

that there can be cycles in this `rdf:type` relation?], which can lead to a Russel paradox. The OMG follows the software engineering philosophy of strict separation between classes and instances.

8. Summary

We describe approaches for converting UML models to the corresponding XML DTDs / Schemas and RDF. The issues identified are mainly related to capturing relationships from the relational models and coping with the transformation of different relationships from UML to XML-based models. The ongoing effort from OMG and W3C communities is to bring the approaches derived two communities closer together to tackle these issues reflected in the conversion between UML and RDF, and more commercial products (as presented in Table 8.1) are becoming available. Even if we assume that it is not intended to develop such a software component for translation to be integrated into the ICONS prototype system, we suggest the use of the UML semantic data model as the principle platform for specifying the content model in the Content Base and domain ontologies in the Ontology Base.

Feature	XMLSpy	EditML	UBOT	ArgoUML	Rational Rose
Conversion between DTD & Schema dialects	*	*			
Generation of DTD/Schema from use-cases	*	*			
Generation of DTD/Schema from database	*	*			
Generation of XML instance documents based on DTD/Schema	*	*			
Conversion between DTD & Schema dialects	*				
Conversion between UML and XML DTD			*	*	
Conversion between UML and XML Schema			*	*	*
Conversion between UML and RDF			*		*

Table 8.1: Feature summary of the current products

Bibliography

- [Berners-lee 2001] Tim Berners-lee, James Hendler and Ora Lissila (2001), Semantic Web Scientific American, May 2001.
- [Horrocks 2002] Ian Horrocks. DAML+OIL: a description logic for the semantic web. *IEEE Bull. of the Technical Committee on Data Engineering*, 25(1):4-9, March 2002.
- [Heflin 2002] Jeff Heflin, Raphael Volz and Jonathan Dale (Edts). Requirements for a Web Ontology Language. <http://www.w3.org/TR/webont-req/>
- [NoteRDFUML1998]A Discussion of the Relationship Between RDF-Schema and UML <http://www.w3.org/TR/NOTE-rdf-uml/>
- [XMI 1999] XMI, <http://www.alphaworks.ibm.com/tech/xmitoolkit>
- [Baclawski 2001] Kenneth Baclawski, Mieczyslaw Kokar, Paul Kogut, Lewis Hart, Jeffrey Smith, William Holmes, Jerzy Letkowski, Mike Aronson, Extending UML to Support Ontology Engineering for the Semantic Web at the Fourth International Conference on UML (UML 2001), Toronto, October 1-5, 2001.
- [ORMapping 2002] Object-Relational Mapping. <http://www.object-relational.com/> (2002).
- [Fussell 2002] Mark L. Fussell. Foundations of Object-Relational Mapping <http://www.chimu.com/publications/> object Relational/
- [XMLSpy 2002] XMLSpy. <http://www.xmlspy.co.uk>
- [UBOT 2002] UBOT. http://ubot.lockheedmartin.com/ubot/details/uml_to_daml.html (October 2002).
- [ArgoUML 2002] ArgoUML. <http://argouml.tigris.org/> (October 2002).
- [ICONS D01] The IST-2001-32429 ICONS Consortium, Intelligent Content Management System. Project Presentation, www.icons.rodan.pl, April 2002
- [NetBryx 2002] NetBryx, <http://www.netbryx.com/default.asp>, 2002.
- [Oracle 2002] Oracle, <http://technet.oracle.com/tech/xml/content.html>
- [ICONS D06] The IST-2001-32429 ICONS Consortium, Analysis and selection of the ICONS project research base, www.icons.rodan.pl, June 2002
- [ICONS D16] The IST-2001-32429 ICONS Consortium, Specification of the ICONS Architecture, www.icons.rodan.pl, December 2002
- [Gruber 1993] Gruber, T. (1993). A translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*. Vol 5.
- [DTD2XSD 2001]. DTD2XSD, http://www.w3.org/2000/04/schema_hack/ (2001).
- [Sumeet 2001] Sumeet Toprani and Shirish Nilekar. (2001). XML DTD to O-R Schema Mapping Tool. <http://www.cs.wpi.edu/~cs542/f01w/projects.htm>.
- [Booch 1999] Crady Booch, Magnus Christerson, Matthew Fuchs and Jari Koistinen, (1999). UML for XML Schema Mapping specification. http://www.rational.com/media/uml/resources/media/uml_xmlschem a33.pdf
- [eBTWG 2001] UN/CEFACT's Electronic Business Transition ad-hoc Working Group (eBTWG). UML2XML: the second requirement addresses support for W3C Schema features. <http://xml.coverpages.org/UML2XMLReqv1-day2.pdf>
- [Davidson 1999] Andrew Davidson, Matthew Fuchs, Mette Hedin, Mudita Jain, Jari Koistinen, Chris Lloyd, Murray Maloney, Kelly Schwarzhof, (1999). Schema for Object-Oriented XML 2.0. <http://www.w3.org/TR/NOTE-SOX/>
- [RDFDB 2001] rdfDB, <http://www.guha.com/rdfdb/>
- [Brickley 2001] Dan Brickley, Libby Miller, RDF, SQL and the Semantic Web - a case study, <http://ilrt.org/discovery/2000/10/swsq1/>

Dictionary

Notion	Meaning
UML	Unified Modelling Language
OMG	Object Management Group
XML	eXtensible Markup Language
DTD	Document Type Definition
XML Schema	Specify a syntax of domain specific for validating XML documents
RDF	Resource Description of Framework
RDFS	RDF schema (an instance of RDF)
ER	Entity Relationship