System Entity Structure For XML Meta Data Modeling; Application to the US Climate Normals

Saehoon Cheon, Doohwan Kim,
RTSync Corp
cheon,dhkim@rtsync.com
Bernard P Zeigler
Arizona Center for Integrative Modeling and Simulation
The University of Arizona
Tucson, AZ
zeigler@ece.arizona.edu

Abstract

Data engineering becomes increasingly important with the increasing popularity of service oriented architecture (SOA) and web services. Such data engineering requires a design methodology within an ontology framework. In this paper, the System Entity Structure (SES) serves as an abstract ontology framework for world state descriptions particularly involving dynamics in space and time. It is implemented within a software tool, the SESBuilder, employing the extended markup language XML. Pruned Entity Structures (PES) represent the logically possible set of world state descriptions consistent with the SES. At the implementation level, an SES is represented by a schema or DTD whose instance documents represent possible prunings. The SESBuilder supports convenient specification of SESs, pruning to create PESs, and transformation to XML representations, all through natural language and graphical interfaces. Thus the software provides a capable data engineering work space. In this paper, we illustrate its application to data modeling of US Climate Normals for the purpose of automated generation of weather simulation models.

Key Words- SES, PES, SESBuilder, XML, Data Modeling

1. Introduction

Data Engineering has become more necessary and critical activity for business, engineering, and scientific organizations as the move to service oriented architecture (SOA) and web services become more popular. In these web based technologies, data must be encoded into a format to be sent from a producer to a consumer. The Extensible Markup Language (XML), the standard format on the web, is employed to encode such data sets.

The paper introduces a well-defined formalism, the System Entity Structure (SES), and its role as an ontology framework. The SES automates the creation of XML schemata using a data model that reflects system engineering concepts of hierarchical decomposition and specialization. The SES may be represented as an instance of a particular DTD and then transformed into an XML DTD or schema. The instances of the latter represent the pruned entity structures (PES) of the SES. These basic concepts of SES and PES are introduced in section 2, and constructing and representing the SES and the PES in computational form are introduced in section 3.

Software named SESBuilder has been developing in collaboration between RTSync Corp (www.rtsync.com) and ACIMS (www.acims.arizona.edu), the University of Arizona. The software provides data engineering work space based on XML for syntax and validity check. Moreover, it supports natural language input for an SES definition. Users can define an SES by following predefined syntax for the SES key components. All of the XML metadata including DTD and schema are automatically generated. A graphical tree view helps users prune an SES. Further details of the SESBuilder are presented in section 4.

A real application employing the US Climate Normal time-indexed weather data is presented in section 5. Encoding the US Climate Normals into XML metadata and query creation to extract data subsets of interest from a large database are automatically generated in the SESBuilder workspace. Such data sets are intended to support automated generation of weather simulation models at various desired levels of resolution (see [4] for details). For the future work, data harmonization issues, and merging operations to create larger composites are discussed.

2. System Entity Structure and Pruned Entity Structure

2.1. Introduction to the System Entity Structure (SES)

The basic idea of system entity structure is that a system entity represents the real system enclosed within a certain choice of system boundary. Figure 1 shows a simple view of entities and their relationship in a system entity structure (SES).

The followings are the key components consisting of system entity structure [2]:

Entity

An entity is intended to represent a real world object which either can be independently identified or is postulated as a component in some decomposition or a real world object.

Aspect

An aspect represents decomposition out of many possible of an entity. The children of an aspect are entities representing components in a decomposition of its parents.

Specialization

A specialization is a mode of classifying entities and is used to express alternative choices for components in the system being modeled. The children of a specialization are entities representing variants of its parent.

Multi-Aspect

A multi-aspect is aspect for which the components are all of one kind.

Variables

A variable is a slot attached to an entity that can be assigned a value from a given range set. It denotes a property, quality, or attribute of an entity to which it is attached.

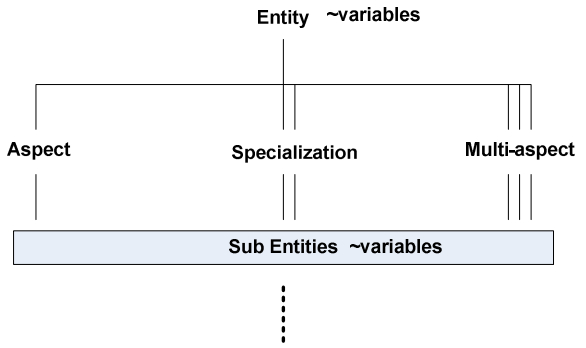

Figure 1 Overview of SES Items and Relationships

2.2. Pruned Entity Structure(PES) and Its Inheritance Representation

The process of pruning the SES is to construct a desired entity structure to meet particular application objectives. More specifically, a specialization may have several entities to select from representing different ways of specializing an entity. Ultimately, in a completely pruned entity structure, every specialization has exactly on entity. In other words, the process of pruning is that of reducing the SES by making selections in all its specializations, allowing for multi-aspect expansions.

PES Inheritance is defined so that the parent and any child of a specialization combine their individual variables, aspects and remaining specializations when pruning is activated. Figure 2 illustrates how the inheritance is processed in the case of multiple decompositions.

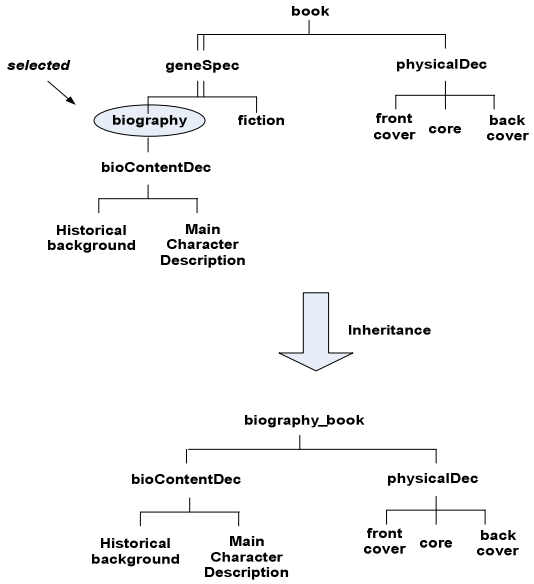

Figure 2 SES and PES Inheritance

The SES framework is especially applicable to simulation modeling of dynamical systems. Comparison with other ontology frameworks is presented in [2].

3. Computational Representation of SES and PES

In this section, we discuss a methodology to construct and represent the SES in computational form.

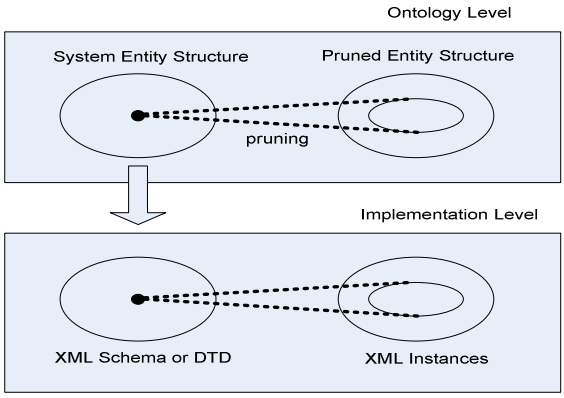

Figure 3 Relating SES and XML through Ontology and Implementation Level

As depicted in Figure 3, a designer creates an SES to describe a given domain at the ontology level. The designed SES is implemented by XML Schema or document type definition (DTD) at the implementation level. The Document Object Models (DOM) class is the core class for representing XML documents in computer memory. We employ Sun's implementation of the DOM specification and Java [6] for the implementation language and is illustrated in Figure 4.

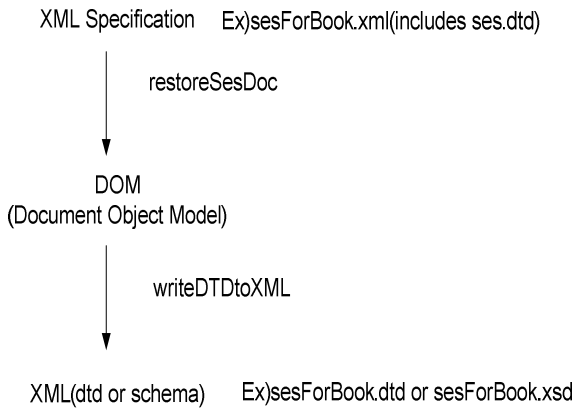

Figure 4 Implementing SES using DOM specification

4. Introduction to SESBuilder

The SESBuilder (http://www.sesbuilder.com) is an integrated tool to provide a data modeling framework supported by SES and PES concepts at the ontology level and implemented in XML schema and XML instances. Besides general data modeling, it supports features that are useful for the development of simulation models expressed in the Discrete Event Specification (DEVS) formalism [1]. The software and extensive tutorials can be downloaded from [5].

4.1. SES Design by Natural Language; syntax and semantics

This section explain the methodology to define a SES using natural language, and show its XML representation with real example in SESBuilder. First natural language forms for specification of SES elements are presented:

Aspect

From VIEW perspective, THING is made of COMPONENT1, COMPONENT2, and COMPONENT3!
Ex) From the structure perspective, a bank is made of a payment and a debt!

Results in:
entity: THING *(bank)*
--aspect: THING-VIEWDec *(bank-structureDec)*
----entity: COMPONENT1 *(payment)*
----entity: COMPONENT2 *(debt)*
Violations in "and" clause result in "entity: unknown"

Specialization

THING can be VARIANT1, VARIANT2, or VARIANT3 in CLASSFAMILY!
Ex) A payment can be credit, cash, or check in form!

Results in:
entity: THING *(payment)*
--specialization: THING-*CLASSFAMILYSpec (payment-formSpec)*
----entity: VARIANT1 *(credit)*
----entity: VARIANT2 *(cash)*
----entity: VARIANT3 *(check)*

MultiAspect

From multiple perspective, THINGS are made of more than one THING !

Ex) From the structure perspective, credits are made of more than one credit!

Results in:
entity: THINGS *(credits)*
--multiAspect: THINGS-multipleMultiAsp *(credits-structureMultiAsp)*
----numComponentsVar: numTHINGS, min: 0, max: 10,
----entity: THING *(credit)*

Variables

THING has VAR1, VAR2, and VAR3!
Ex) A check has a signature, a bank, and an authorizedState!
The range of check's signature is string!
The range of check's bank is string!
The range of check's authorizedState is boolean!

Results in:
entity: THING *(check)*
--var: VAR1*(authorizedState),* rangeSpec: *(boolean)*
--var: VAR3*(bank)*, rangeSpec: *(string)*
--var: VAR2*(signature)* ,rangeSpec: *(string)*
Restriction: THING must be an entity in some other sentence.

4.2. SESBuilder View

The SESBuilder graphical user interface has several tabs. Given some natural language input as in Figure 5, its DTD, Schema, and XML output data are automatically generated. The "TreeView" button opens a new window showing a tree representation and allowing the user to prune the SES. The inherited structure explained in section 2 is shown in "PESInheritance" tab with XML format. In addition, the SESBuilder currently is able to generate both DTD and Schema output, and allows users to set the values for variables in the "Tree View" window. For more information, please visit http://www.sesbuilder.com and download the software and tutorial. In the next section, a real example illustrates how the SESBuilder supports model selections and their coupling.

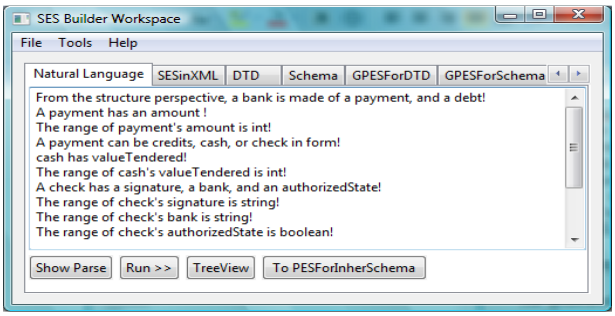

Figure 5 SESBuilder View

5. Case Study ; Automatic Model Generation for Simulation

5.1. Project Introduction and Purpose

Creating generators to provide input trajectories to a system model is a critical task in modeling and simulation. A methodology to automatically create DEVS generator models was developed in [4] and briefly described here. Event information consisting of time and value pairs is derived from the time-indexed trajectories of variables. This enables a DEVS model to be created from the given event set. To enable meaningful model structuring based on a model developer's request, a search engine extracts the specified data from the data base, and then a model creation engine is launched to create the DEVS model.

The source data need to be organized within a metadata format to support the flexible and efficient data handling required for model creation. Therefore, the source data designed by the SES contains only valuable information with the XML format. A parsing engine is required to retrieve the XML metadata containing the critical event set.

5.2. US Climate Normals

Climate is an important factor in agriculture, commerce, industry, and transportation. The National Oceanic and Atmospheric Administration's (NOAA's) National Climatic Data Center (NCDC) has a responsibility to establish and record the climatic conditions of the United States. The average value of a meteorological element over 30 years is defined as a climatological normal. These normals are summarized in daily, monthly, divisional, and supplementary normals products. This data we experimented with includes daily 1971-2000 normal maximum, minimum, and mean temperature (degrees F), heating and cooling degree days (base 65 degrees F), and precipitation (inches) [7].

5.3. Metadata Representation for Source Data; US Climate Normals

There are almost 6000 stations to observe the weather in the United States. All of these station data have a same format and these data are daily recorded, one data set per day. The SES representation, illustrated in Figure 6, is a simplified description and one of many possible definitions to describe the weather station data. Indeed, the choice represents a designer's ontological commitment and should be driven by the application in mind.

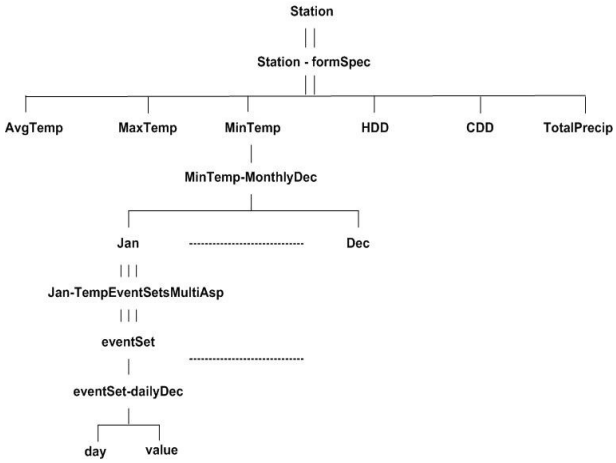

Figure 6 SES for Weather Station

Figure 7 illustrates the natural language input to describe the SES shown in Figure 6. Note that to save space both figures concentrate only on the minimum temperature normal for January.

```
Station can be AvgTemp, MaxTemp, MinTemp, HDD,CDD, or TotalPrecip in form!
From the monthly perspective, MinTemp is made of Jan, Feb, Mar, Apr, June, July, Aug,
Sept, Oct, Nov, and Dec in name!
From the TempEventSets perspective, Jan is made of more than one eventSet!
From the daily perspective, eventSet is made of day and value!
```

Figure 7 Natural Language Input for Weather Station

The XML output for the input of Figure 7, is illustrated in Figure 8. The PES inheritance output already explained in section 2 illustrates that the minimum temperature and January are selected, and these XML output is automatically generated in the SESBuilder. The grayed box focuses on the inherited entities. In this output, the number of the event set is only 2, but it can have multiple event sets as many as we want to add. Even though they have same decomposition, eventSet-dailyDec, each event set is unique. Using these predefined tags, unique tag and temperature in each day can be filled. The SESBuilder has been developing to support all this capability in the "TreeView".

```
<?xml version="1.0" encoding="UTF-8"?>
<MinTemp_Station xmlns:xsi =  "http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="xmlinSCH.xsd">
<aspectsOfMinTemp>
<MinTemp-monthlyDec  coupling = "">
<Feb/>
<Mar/>
<Apr/>
<June/>
      <Jan>
            <aspectsOfJan>
                  <Jan-TempEventSetsMultiAsp  numContainedInJan = "1">
                        <eventSet>
                              <aspectsOfeventSet>
                                    <eventSet-dailyDec  coupling = "">
                                    <value/>
                                    <day/>
                                    </eventSet-dailyDec>
                              </aspectsOfeventSet>
                        </eventSet>
                  </Jan-TempEventSetsMultiAsp>
            </aspectsOfJan>
      </Jan>
<July/>
<Oct/>
<Nov/>
<Sept/>
<Aug/>
<Decinname/>
</MinTemp-monthlyDec>
</aspectsOfMinTemp>
</MinTemp_Station>
```

Figure 8 XML Schema Metadata Output for Weather Station

5.4. Query Generation for US Climate Normals

The query format to request a data is also developed using the SES. The query in the example supports both time and geographical spaces, and up to 6 kinds of weather variables at once. The spaces have three different types of specification. The Region entity has reflects the regional division used by US Census Bureau. GeographicalLocation was defined to support geographical coordinates, both latitude and longitude. By pruning the SES, users can specify a desired data set. In addition This example shows that an SES specifies a family of hierarchical modular structures. Once constructed, such structures may be treated as components to be used in creating larger composites. This task is supported by a merge operation. These entities such as regions, geographical locations, and states are merged by their sub entities, and then merged to form one large structure. Not all of the sub entities have been shown in figure due to the space limitation. Please refer to [4, 8-10]

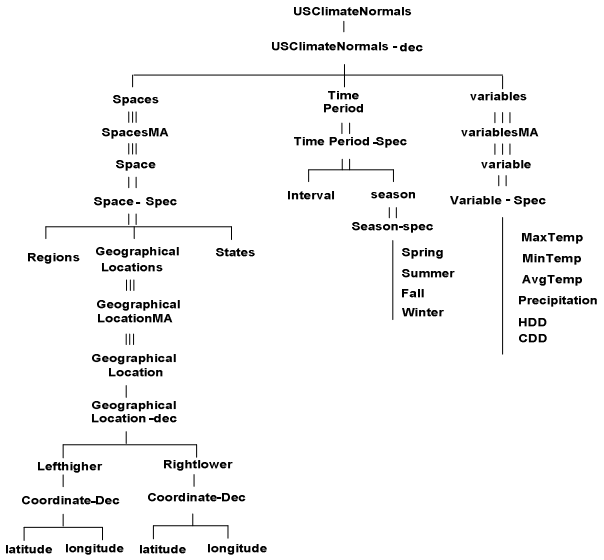

Figure 9 SES for Query

for more detailed information. In addition to composition, decomposition is also available. The ability to compose and decompose hierarchical modular structure is the most important feature to manage the complexity of data representation.

As before, an XML schema is auto-generated from the input of Figure 10 and Figure 11 illustrates a PES to specify a data set to support DEVS model generation. Note that it describes just a few of all the possible selections based on the SES in Figures 9 and 10.

6. Summary

The Unified Modeling Language (UML [UML]) is a software development language and environment that has found application in data engineering. The Object Modeling Group (OMG) has initiated a process for developing an Ontology Definition Metamodel (ODM) for modeling Semantic Web ontology languages within the context of OMG's Model-Driven Architecture (MDA). A proposal for such a definition is presented in [11] which develop an Ontology UML Profile [12-14]. Unfortunately, although UML was developed to support interoperability of tools, few can actually exchange UML models without information loss.

Further, the System Entity Structure (SES) ontology framework provides unique support for data engineering in the context of simulation modeling. Implemented within a software tool, the SESBuilder, it supports convenient specification of SESs, pruning to create PESs, and transformation to XML representations, all through natural language and graphical interfaces. Thus the software provides a capable data engineering work space.

A full discussion of the strengths of the SES ontology framework in relation to UML developments is provided in [2].

From the form perspective, USClimateNormals is made of Spaces, TimePeriod, and Variables!
From the mult perspective, Spaces is made of more than one Space!
Spaces can be Regions, GeographicalLocations, or States in type!
From the corner perspective, GeographicalLocations is made of LefthigherCorner and RightlowerCorner!
From the coordinate perspective, LefthigherCorner is made of latitude and longitude!
latitude has degrees, minutes, and seconds!
The range of latitude's degrees is int!
The range of latitude's minutes is int!
The range of latitude's seconds is int!
longitude has degrees, minutes, and seconds!
The range of longitude's degrees is int!
The range of longitude's minutes is int!
The range of longitude's seconds is int!
From the coordinate perspective, RightlowerCorner is made of latitude and longitude!
latitude has degrees, minutes, and seconds!
The range of latitude's degrees is int!
The range of latitude's minutes is int!
The range of latitude's seconds is int!
longitude has degrees, minutes, and seconds!
The range of longitude's degrees is int!
The range of longitude's minutes is int!
The range of longitude's seconds is int!

TimePeriod can be Interval or Season in timeForm!
Season can be Spring, Summer, Fall, or Winter in seasonForm!

From the multi perspective, Variables is made of more than one Variable!
Variable can be MaxTemp, MinTemp, AvgTemp, Precip, HDD, or CDD in form!

Figure 10 Natural Language Input for Query

```
<?xml version="1.0" encoding="UTF-8"?>
<USClimateNormals xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="xmlinSCH.xsd"
pruneName = "USClimateNormals">
<aspectsOfUSClimateNormals>
<USClimateNormals-formDec coupling = "">
    <Variables pruneName = "Variables">
        <aspectsOfVariables>
            <Variables-multiMultiAsp numContainedInVariables = "1">
                <MinTemp pruneName = "MinTemp_Variable">
                </MinTemp>                                   <-- MinTemp has been selected
            </Variables-multiMultiAsp>
        </aspectsOfVariables>
    </Variables>
    <GeographicalLocations pruneName = "GeographicalLocations_Spaces">   <-- GeographicalLocation has been selected
        <aspectsOfGeographicalLocations>
            <GeographicalLocations-cornerDec coupling = "">
                <RightlowerCorner pruneName = "RightlowerCorner">
                    <aspectsOfRightlowerCorner>
                        <RightlowerCorner-coordinateDec coupling = "">
                            <longitude degrees = "111" minutes = "41" pruneName = "longitude" seconds = "00">
                            </longitude>
                            <latitude degrees = "33" minutes = "17" pruneName = "latitude" seconds = "00">
                            </latitude>
                        </RightlowerCorner-coordinateDec>
                    </aspectsOfRightlowerCorner>
Coordinate Definition
                </RightlowerCorner>
                <LefthigherCorner pruneName = "LefthigherCorner">
                    <aspectsOfLefthigherCorner>
                        <LefthigherCorner-coordinateDec coupling = "">
                            <longitude degrees = "112" minutes = "06" pruneName = "longitude" seconds = "00">
                            </longitude>
                            <latitude degrees = "33" minutes = "38" pruneName = "latitude" seconds = "00">
                            </latitude>
                        </LefthigherCorner-coordinateDec>
                    </aspectsOfLefthigherCorner>
                </LefthigherCorner>
            </GeographicalLocations-cornerDec>
            <Spaces-multMultiAsp numContainedInSpaces = "1">
                <Space pruneName = "Space">
                </Space>
            </Spaces-multMultiAsp>
        </aspectsOfGeographicalLocations>
    </GeographicalLocations>
    <Spring pruneName = "Spring_Season_TimePeriod">
    </Spring>                                                <-- Spring Season has been selected
</USClimateNormals-formDec>
</aspectsOfUSClimateNormals>
</USClimateNormals>
```

Figure 11 PES Inheritance XML Output

7. References

[1] Zeigler B.P. "Theory of Modeling and Simulation" Wiley & Son, N.Y, 1976

[2] Zeigler, B.P. and Hammonds, P.E., "Modeling and Simulation-Based Data Engineering: Introducing Pragmatics into Ontologies for Net-Centric Information Exchange", Elsevier, 2007

[3] Zeigler B P, Herbert Praehofer, Tag Gon Kim, " Theory of Modeling and Simulation", Academic Press, 2000

[4] Saehoon Cheon, "Experimental Frame Structuring for Automated Model Construction: Application to Simulated Weather Generation", Doct. Diss. Dept. of ECE, U. Arizona, Tucson, AZ,2007

[5] SESBuilder, http://www.sesbuilder.com/

[6] Document Object Model (DOM), http://www.w3.org/DOM/

[7] National Climate Data Center, http://www.ncdc.noaa.gov/oa/ncdc.html

[8] Saehoon Cheon, Bernard P Zeigler, "Experimental Frame Structuring and Aggregation of Source Data: Application to US Climate Normals", SpringSim07, Norvolk, VA

[9] Saehoon Cheon, Zeigler. B.P "Web Service Oriented Architecture for DEVS Model Retrieval by System Entity Structure and Segment Decomposition", scs2006, Alabama.

[10] DEVSJAVA3.0, http://acims.arizona.edu

[11] [GAVE] Gašević, D., Djurić, D, Devedžić, V, "Model Driven Architecture and Ontology Development," Springer-Verlag, Berlin, 2006, ISBN: 3-540-32180-2 (in press).

[12] Unified Modeling Language (UML) Superstructure, version 2.0 http://www.omg.org/technology/documents/formal/uml.htm(accessed Nov. 2006)

[13] MOF 2.0 / XMI Mapping Specification, v2.1, http://www.omg.org/technology/documents/formal/xmi.htm/ (accessed Nov. 2006)

[14] Meta Object Facility (MOF) Core Specification, http://www.omg.org/technology/documents/formal/mof.htm/ (accessed Nov. 2006)