A Functionality Coverage Analysis of Industrially used Ontology Languages

Olof Johansson, Adrian Pop, Peter Fritzson
Programming Environments Laboratory, Dept. of Computer and Information Science, Linköping University

Ontology development for engineering applications and domains is a time consuming negotiation and development process that takes years to complete, involving many domain experts and tool vendors that must agree. Once agreement is reached, an ontology serves as a common language, that allows engineers and machines to share data and knowledge. The long term goal with this work is to share and reuse engineering ontologies amongst different programming languages and tools, and thus facilitate engineering system integration and automated sharing of huge amounts of engineering knowledge and product data.
The paper presents and compares ontology functionality using UML diagrams for the software design language UML 1.5, simulation modeling language Modelica 2.1, and e-business datadictionary RosettanNet technical dictionary 3.2. The conclusion is that static, structural ontologies and product data can be shared amonst these languages using fully automated processes, however UML TaggedValues and Modelica Annotations or CommentStrings must be used in a standardized way for full roundtrips.

Background on Compared Ontology Languages

A static ontology is here defined as classes with multiple inheritance and attributes of standard data types.

OMG UML

The latest adopted specification of OMG UML is version 1.5 [1]. Static ontology functionality is available in the core package and extension mechanisms. See the overview UML diagrams in Figure 2-5 "Core Package - Backbone", p 2-13, Figure 2-6 "Core Package - Relationships" , Figure 2-8 "Core Package - Classifiers" and Figure 2-10 "Extension Mechanisms" p 2-76 in [1].
The probably most widely supported vendor neutral interchange file format for UML-models is XMI version 1.1 [2], with UML1.3 as meta-model. UML 1.5 is backwards compatible and adds an action semantics package for modelling the run-time semantics of executable actions and procedures, which is not used in this study.

Modelica

Modelica is a an object-oriented modelling language, designed to allow convenient, component-oriented modelling of complex physical systems, and used for behavioural simulation. The language is developed by Modelica Association [3], and the latest released language specification is Modelica 2.1 [4]. Modelica is accompanied with a free Modelica Standard Library for many engineering domains, which can be seen as ontologies for various engineering simulation domains. For an in-depth reference on Modelica, see [5].
Major differences between a Modelica- and an UML-model is that Modelica models specify a static structure of instances which remains fixed throughout a simulation, and that declarative mathematical knowledge can be expressed as equations, whose behavioural execution order is determined by the Modelica compiler.
Diagram 1 shows an UML-meta-model of Modelica 2.1. This is work under development and action semantics for algorithms, equation expressions, modification expressions etc are modeled as parse trees, where the meta-attribute ParseNode.nodeType determines the meta-class.
The predominant interchange format for Modelica models are Modelica source code files and packages. Recently an XML-based interchange format called ModelicaXML has been developed [6]. Modelica source code files can hence be transformed to xml-files, and used for tool import with an XML parser.

RosettaNet Technical Dictionary

RosettaNet is a non-profit consortium of more than 500 organizations working to create, implement and promote open e-business standards and services [7]. RosettaNet was founded in February 1998 by 40 information technology companies, and quickly grew to include the electronic components, semiconductor manufacturing and telecommunications industries as well as the logistics and solution provider communities.
The latest release of RosettaNet technical dictionary is RNTD 3.2 [8], and is mainly used for specifying product data in automated e-business transactions, using RosettaNet Partner Interface Processes (PIPs) which define standardised business processes between trading partners.
The dictionary contains 943 Classes and 3748 CharacteristicDefinitions which in UML correspond to a reusable well documented datatype. RNTD is distributed in an XML-file specified by a DTD and [9]. An UML information model of RNTD is presented in Diagram 2. This is also work in progress.

Functionality Coverage analysis

Table 1 lists how static ontology functionality for engineering data exchange and applications can be mapped to UML, Modelica and RNTD.

Table 1

Ontology Function	**UML Means**	**Modelica Means**	**RNTD Means**
Product data exchange			
Class	Class	Class	Class, PropertyDefinitionSet
"Multiple inheritance"	Generalization	Class Extension, multiple components of reused classes.	Multiple reused PropertyDefinitionSets
Attribute sets for reuse in multiple classes	Class with attributes defining the attibute set.	Class with record restriction, defining the attribute set. [4] p 43	PropertyDefinitionSets
Single-value Attribute	Attribute with type that is a subtype of DataType	Component variable of a PredefinedType or subType.	CharacteristicDefinition included in propDefs of a Class
Standard datatypes	Subclasses of DataType, ProgrammingLanguageDataType.expression TypeExpression	Subclasses of PredefinedType	Type owned by Domain.
Unit on continous Attributes	Possible with: TagDefinition, TaggedValue	Real.unit, less strong with PredefinedType.quantity	IntMeasureType.unit and RealMeasureType.unit
Documented discrete attribute value choices	Enumeration with EnumerationLiterals having Comments or TaggedValues	Enumeration,EnumerationLiteral CommentStrings.	ValueDomain with DicValues refering to TermDefinitions
Ontology documentation			
Names and multiple synonymes	ModelElement.name, TagDefinition, TaggedValue	ModelObject.identifier, Annotations may own a class_modification that can store arbitrary "tagged value"s. [4] p 15, 106 ff	Names (inherited by all DictionaryEntries), Synonymous.name
Element definitions, and documentation references.	Possible with: ModelElement.comment, TaggedValue, TagDefinition	Possible with: CommentStrings, Annotation (see above)	CommonElements, TermDefinitions, AppSpecific
Global Identification and Version tracking			
Globally unique identification	Possible with: TagDefinitions and TaggedValues on objects	Possible with: Annotations	Identifiers.code & majRev
Version tracking	See above + version maintenance procedures	See above + version maintenance procedures	Identifiers.code , majRev, minRev + [10]

Conclusions and Future Work

Table 1 shows that the studied languages can represent static ontology functionality. Work towards full round-trip experiments has started with refining the UML-models, generate their relational database implementations using SQL-based MDA-related techniques [11], create XML-importers/exporters for XMI, ModelicaXML and RNTD. Declarative ontology transformation descriptions and their implementation for model transformations between different meta-models have evolved since [12] but need more work. Until then, simpler but less generic methods can be used.

References

[1] OMG, OMG Unified Modeling Language Specification, Version 1.5, March 2003, http://www.omg.org/cgi-bin/doc?formal/03-03-01

[2] OMG, OMG XML Metadata Interchange (XMI) Specification , Version 1.1, November 2000, http://www.omg.org/cgi-bin/doc?formal/00-11-02

[3] Modelica Association, http://www.modelica.org

[4] Modelica Association, Modelica - A Unified Object-Oriented Language for Physical Systems Modeling: Language Specification, Version 2.1, January 30, 2004, http://www.modelica.org

[5] P. Fritzson, Principles of Object-Oriented Modeling and Simulation with Modelica 2.1, IEEE Press, 2004, ISBN 0-471-47163-1

[6] A. Pop, P. Fritzson, ModelicaXML: A Modelica XML Representation with Applications, Proc of 3rd International Modelica Conference, November 3-4, 2003, http://www.modelica.org/Conference2003/papers.shtml

[7] RosettaNet , http://www.rosettanet.org

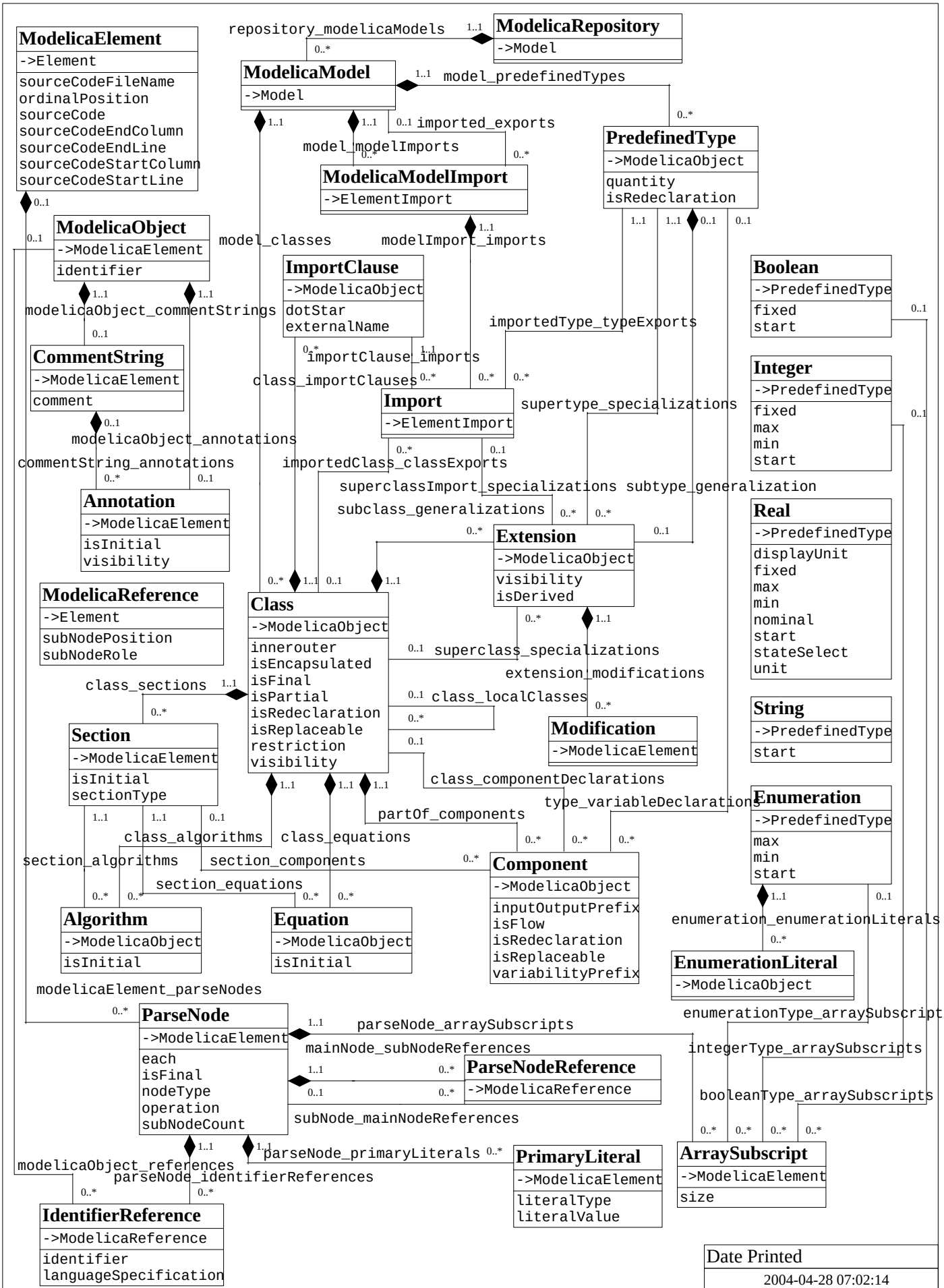
[8] RosettaNet, RosettaNet Techical Dictionary v3.2, http://www.rosettanet.org/technicaldictionary

[9] RosettaNet, RNTD Content Coding Specification, Version 1.01, November 2001, in [8]

[10] RosettaNet, RNTD Maintenance Specification, Version 1.01, Nov 2001, in [8]

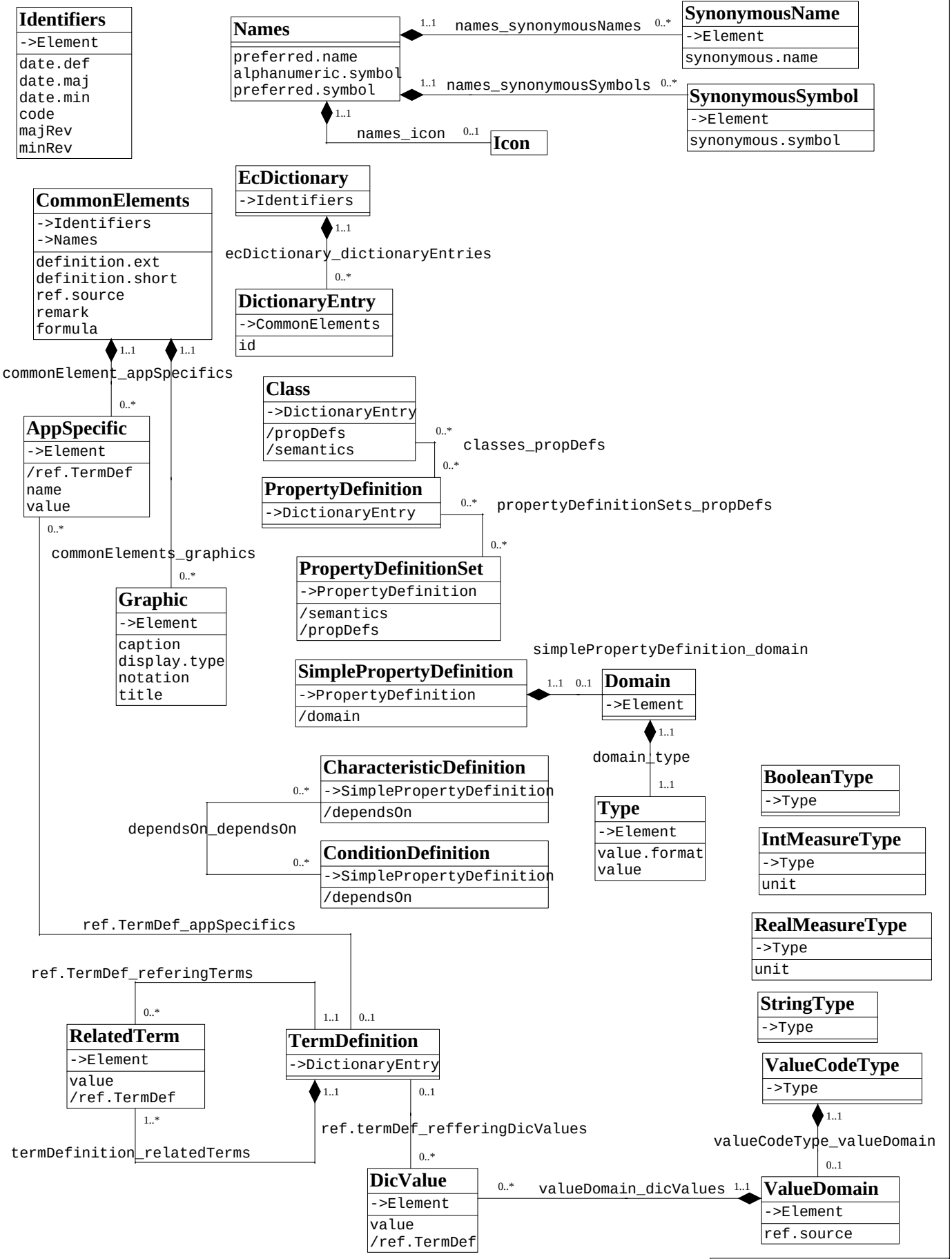
[11] O. Johansson, Development Environments for Complex Product Models, 1996, ISBN 91-7871-855-4

[12] O. Johansson, Using a meta-database for integration planning and documentation of product modeling systems, Proc. of Produktmodeller-98, Nov 10-11, 1998, ISBN 91-7219-364-6



Date Printed
2004-04-28 07:02:14

	Domain Model	Filename	Date Added	Date Modified	#Cls	#Rel	#Att
DM.	modelica-02h	modelica-02h.odm	2003-03-04 08:03:07	2004-04-28 06:57:16	42	42	124
	Modelica database design for ModelicaXML information exchange. The goal is a repository that can be round-tripped with Modelica source code.						
Dgm	MDAFA2004: Modelica Package		2003-03-04 09:33:10	2004-04-28 06:57:16	29	40	-
	Diagram 1: UML Model of Modelica 2.1. (Note that "->ModelicaObject", means inherits from ModelicaObject)						



Date Printed
2004-04-28 07:02:46

	Domain Model	Filename	Date Added	Date Modified	#Cls	#Rel	#Att
DM.	RNTD32d	RNTD32d.odm	2004-04-25 21:06:38	2004-04-28 06:02:35	43	22	112
	RosettaNet Technical Dictionary, according to the RNTD content coding specification V1.01, Nov 2001, and rntd.dtd in the [8] distribution.						
Dgm	2) RNTD		2004-04-25 21:09:56	2004-04-28 06:02:35	27	17	-
	Diagram 2: UML model of ontology subset of RNTD. (Note that "/ref.TermDef" represents an IDREF XML attribute that implements a relationship)						