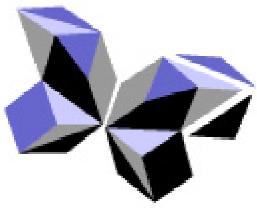

Digital Business Ecosystem

Contract n° 507953

Workpackage 15: DBE Business Modeling Language

Deliverable D15.4: Ontology Creator/Importer/Viewer

Information Society Technologies

Project funded by the European Community under the “Information Society Technology” Programme

Contract Number: 507953

Project Acronym: DBE

Title: Digital Business Ecosystem

Deliverable N°: D15.4

Due dates: 06/2006

Delivery Date: 07/2006

Short Description:

This document accompanies the software deliverable for the final release of the interactive visual ontology Creator/Importer/Viewer tool. This tool is being used for manipulating domain ontologies into DBE. In order to represent ontologies into DBE an Ontology Definition Metamodel (ODM) has been developed. The Ontology Editor is needed for giving the ability to domain experts to conceptualise (define ontologies for) business domains. Also, the Ontology Browser is needed in order to provide to the business analysts the facility to browse through the domain ontologies and view the defined concepts and their relationships. Finally, interoperability with existing ontologies is a valuable function for DBE in order to act as an open environment. ODM is able to represent (using MOF) existing ontologies described with the Ontology Web Language (OWL-DL). The main characteristics of the Creator/Importer/Viewer tool are:

- It allows visual modeling of domain ontologies from scratch. A UML-Like notation for the ODM has been designed and implemented. The modelled ontologies are stored either locally or into the DBE Knowledge Base.
- It allows viewing (browsing) of ontologies supported by the DBE Ontology Metamodel (ODM).
- It allows importing of OWL-DL ontologies from the semantic web community into the DBE Environment and vice versa (exporting DBE ODM ontologies as OWL-DL ontologies).

Author: Technical University of Crete (TUC)

Partners contributed: Technical University of Crete (TUC)

Made available to: Public

Versioning		
Version	**Date**	**Author, Organization**
1.0	30/05/2006	Panagiotis Kontogiannis, Dimitris Kafantaris, Editing by Fotis G. Kazasis, Technical University of Crete
2.0	14/07/2006	Revisions based on the reviewers' comments/suggestions/changes

Quality check:

1st Internal Reviewer: Javier Val, Miguel Angel Barcelona, ITA

2nd Internal Reviewer: Giulio Montanari, Soluta.net

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 2.5 License. To view a copy of this license, visit : http://creativecommons.org/licenses/by-nc-sa/2.5/ or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

creative commons

COMMONS DEED

Attribution-NonCommercial-ShareAlike 2.5

You are free:

- to copy, distribute, display, and perform the work
- to make derivative works

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor.

Noncommercial. You may not use this work for commercial purposes.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

Table of Contents

List of Figures

List of Tables

Executive Summary

The core of the DBE Knowledge Representation Framework (DBE KRF) consists of domain specific ontologies. These ontologies capture community accepted semantics (concepts and relationships between them) of a particular business domain and they are the reference point of every specific model or semantic description allowing real knowledge sharing and semantic interoperability among SMEs. That is, SMEs model and semantically describe their business and their offerings (services) by re-using community-wide accepted and understood concepts.

The DBE Ontology Definition Metamodel (ODM) has been developed in order to represent domain specific ontologies in the adopted framework, which is based on the MOF metadata architecture. ODM is able to describe new ontologies (that will be developed by regional catalysts or business experts) and at the same time it is also able to represent existing ontologies from the semantic web in order to make the Knowledge Representation Framework of the DBE KB, and thus the DBE itself, an open environment.

The common conceptualisation of a business domain is captured in domain specific ontologies defined with the Ontology Definition Metamodel. For the modelling and description of specific businesses and services specific business model and service ontologies were needed. However, such ontologies must be specific to business and service modelling. That is, the mechanisms that will be used for their definition needs to enforce the user to define concepts that have specific semantics under the prism of their purpose. For example, a business model ontology that defines a business model needs to make use of primitives that have a clear meaning and interpretation in business modelling and not in generic knowledge representation. The modelling and representation of such ontologies is made by using specific modelling languages, which provide the required modelling primitives to the business experts. The DBE Business Modelling Language (BML) [19] is virtually made of two languages; the Business Organizational Metamodel (BOM) and the Semantic Service Language (SSL). The first is related to the business representation from an organisational point of view, while the second represents the service from an interaction and business perspective. In addition the Service Description Language (SDL) [20] provides the technical description of the programmatic interface of a service. ODM supports both a domain ontology (i.e. the domain vocabulary in pure business terms) for utilizing semantics in the BML as well as an information system ontology for utilizing the semantic of the services and data used in the SDL [21].

This document accompanies the software deliverable for the final release of the interactive visual ontology Creator/Importer/Viewer tool. This tool is being used for manipulating the ODM ontologies into DBE. The Ontology Editor is needed for giving the ability to domain experts to conceptualise (define ontologies for) business domains. Also, the Ontology Browser is needed in order to provide to the business analysts the facility to browse through the domain ontologies and view the defined concepts and their relationships. Finally, interoperability with existing ontologies is a valuable function for DBE in order to act as an open environment. ODM is able to represent (using MOF) existing ontologies described with the Ontology Web Language (OWL-DL).

1. Introduction

The Ontology Analysis Tool, one of the core components of the DBE Studio [16], [22], provides a visual environment based on an UML-like graphical user interface that enables business analysts to deploy domain-specific ontologies in order to describe the business requirements of SMEs in the context of the DBE project. The DBE Studio is the DBE Integrated Development Environment (IDE) and it is implemented upon the Eclipse platform. In Eclipse, all tools that compose the platform are implemented as plugins and correspondingly all DBE Studio tools are implemented in the same way. Figure 1 depicts the logical synergies among the various DBE Studio components. The ontology modelling activities that are handled by the Ontology analysis tool are taking place at the initial stage of this logical workflow.

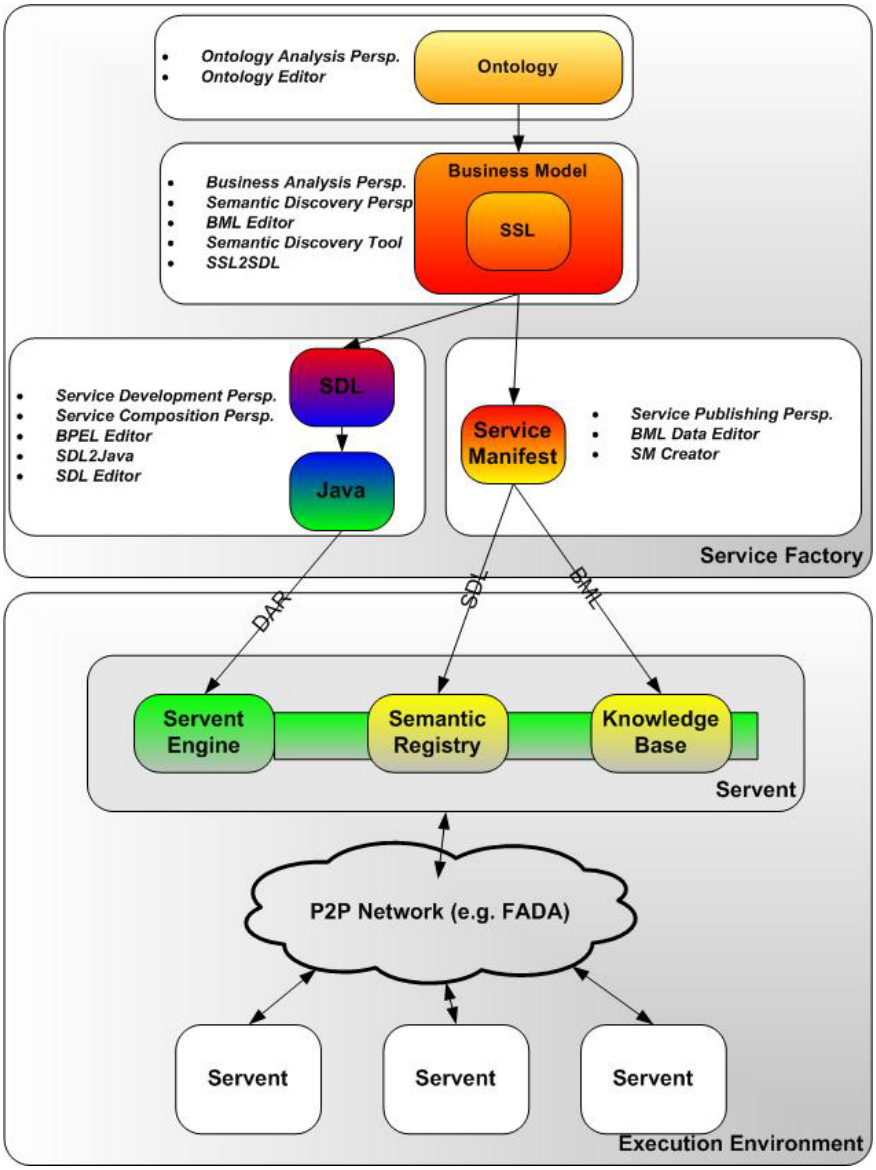

Figure 1: DBE Studio General Workflow (figure obtained by the DBE Deliverable D20.9: DBE Studio User Interface Evaluation produced by INTEL)

The ontologies definition is based on the ODM. The ODM (Ontology Definition Metamodel) is a MOF 1.4 model developed in DBE for ontology representation. It is compatible with the OWL [1] and it can be represented using XMI 1.2 (XML metadata interchange) format [17]. The ODM-based ontologies can be stored locally (in the local file system) or into the DBE Knowledge Base [15] using the JMI 1.0 (Java Metadata Interface) standard. In order to allow DBE to be able to handle existing ontologies described with OWL a mapping from ODM to OWL concepts and vice versa has been realized and the respective mechanisms that support this mapping have been developed.
The rest of chapter 1 provides a comparison analysis between the DBE Ontology Analysis Tool and a number of available systems and tools that offer similar functionality. Chapter 2 describes the functionality offered by the tool following a use-case based methodology. The architecture of the tool as well as its communication with other DBE modules/components is presented in chapter 3. Chapter 4 provides a user guide for navigating within the graphical user interface and utilizing the offered functionality. Finally, chapter 5 describes the mechanisms that have been developed in order to support the interoperability of DBE ontologies with ontologies of the Semantic web represented in the OWL language.

1.1. Related Systems and Development Tools

In the area of semantic web, many applications and tools have been developed for the design and deployment of domain-specific ontologies. Next we present the most representative ones:
Protégé (Stanford Medical Informatics, Stanford University) [6] is the most popular platform for the creation and management of domain-specific ontologies. It provides an extended environment using plug-ins (OWL, ezOWL etc.) that supports many description and storage formats (OWL, RDF, XML and HTML), visual components (diagrams, graphs and tables), multimedia data (audio, images and video), customised forms for data manipulation etc.
The OWL plug-in [7] enables users to create domain-specific ontologies using a form-based user interface. Users can also manage and visualize ontology classes, properties and SWRL (Semantic Web Rule Language) rules, define logical class characteristics as OWL expressions and edit OWL individuals for semantic web mark-up. The domain ontologies can be represented using OWL and RDF format. Finally, the OWL plug-in can integrate OWL reasoners for the validation and testing of the ontology models. The Protégé ezOWL plug-in [8] provides a visual editor of ontologies using diagrams, graphs and tables.
Construct (Network Inference) [9] is a commercial Microsoft Visio plug-in that supports a graphical environment for creating, managing and representing semantic models and ontologies. Ontologies can be presented in an UML-like graphical editor using one or more diagrams. The plug-in can be integrated with the Cerebra Server™ that provides reasoning, querying, error validation and consistency checking mechanisms. The users can collaborate and share their knowledge.
IsaViz (W3C - World Wide Web Consortium) [10] is a graphical environment for creating, editing and browsing RDF models that can be represented using diagrams. It supports a 2.5d graph-based workspace using rectangles, eclipses, boxes and arrows for the design and graph style sheets (GSS) for the visual representation of the models. The IsaViz environment enables users to import models using RDF/XML, Notation3 and N-triple languages, and export them to RDF/XML, Notation3, N-Triple, SVG (Scalable Vector Graphics) and PNG (Portable Network Graphics). The IsaViz does not support the OWL-based models.

KAON OI-modeler (FZI Research Center & AIFB Institute, University of Karlsruhe) [11] is an interactive graphical tool for the creation, editing and browsing of ontologies. The main advantage of KAON is its ability to manage very large ontologies using multi-level deployment. Ontologies can be stored in an Engineering Server that supports the concurrent and collaborative processing of ontologies by different users. The description language of the ontology models is a RDF extension. The tool also supports queries using the KAON Query language.

MR3 (Shizuoka University & AIST (National Institute of Advanced Industrial Science and Technology)) [12] is a tool for creating and editing RDF and RDF entities. It consists of a graphical editor for editing and processing RDF models and entities and supports validation checking mechanisms between the processing models and their corresponding meta-models. MR3 does not support OWL.

OntoTrack (University of Ulm Dept. for Artificial Intelligence, Germany) [13] is an interactive tool for creating, editing and browsing ontologies. It enables users to create ontologies in a graphical environment using only one diagram per ontology, it can manage and process very large ontologies and it uses OWL Lite as the models description language. Finally, OntoTrack can integrate an external reasoner (RACER Pro) in order to provide consistency and validation checking mechanisms to the ontology models.

RDFAuthor (Damian Steer) [14] is an interactive environment for the creation and editing of RDF models. It supports a graphical editor that enables users to create RDF models in a drag-and-drop way using only one diagram per model and a notation quite different from the UML notation.

1.2. Comparison Analysis

The following table presents the basic characteristics of the tools described above comparing to the ones provided by the DBE Ontology Analysis Tool.

Tools/ Characteristics	DBE Ontology Analysis Tool	Construct	Protege	IsaViz	KAON	MR3	OntoTrack	RDF Author
Description Language	ODM/OWL Lite import-export is supported	OWL	OWL	RDF, GSS	KAON, RDF Extension	RDF(S)	OWL Lite in RDFS notation	RDF
Database Support	XML/Relational	Relational	Relational	N	Relational	N	N	N
Web Support	N (Reference Ontologies by URI)	URI namespaces	Reference Ontologies by URI	URIs, XML namespaces	Y	URIs	RDF, URIs and namespaces	URIs (web links, remote RDF query)
Import / Export Formats	ODM/ODM, XML, OWL Lite	OWL, XML	RDF, RDFS, DAML+OIL, XML, OWL, Clips, UML, Notation3, N-Triple	RDF/XML, Notation3, N-Triples, SVG, PNG	RDFS, Protégé RDFS	RDF/XML, N-Triples, PNG	RDF/XML, N-Triple	RDF, XML
Graph View – Editing	Y	Y	Y	Y	Y	Y	Y	Y
Consistency Check	Y	Y	Y	Y	Y	Y	Y	Y

UML Notation	Y	Y	Y	N	Y	Y	Y	N
License	GPL-licensed Open Source	Commercial	Open Source	Open Source	Open Source	Open Source	Open Source	Open Source
Multiple Diagrams	Y	Y	N	N	N	Y	N	N
Eclipse Plug In	Y	N	N	N	N	N	N	N
Multi User Support	N	N	N	N	Y	N	N	N

Table 1: Comparison Characteristics between DBE Ontology Analysis Tool and related systems

The DBE Ontology Analysis Tool (at least by the time it was designed and developed) appears to be the first one that allows the complete management of ontologies represented in a MOF model (i.e. metamodel) complying to the requirements of the OMG's ODM RFP [23], it acts as an integral component of the DBE environment that follows the MDA approach and it is implemented as an eclipse plugin. In specific the most important advantages of the Ontology Analysis Tool regarding to the other systems are the following:

- It has been developed using the Eclipse platform as an Eclipse plug-in, according to the strict system requirements of the DBE project.
- It can be used as an autonomous tool that supports ontology models analysis as well as a component of the DBE Studio platform in full interoperability and collaboration with the other DBE components such as the Ontology Viewer, the ServENT, the Knowledge Base system etc.
- It is the only tool that supports the ODM, an Ontology Definition Metamodel. The ODM is based on the Model Driven Architecture (MDA) and the Meta Object Facility (MOF 1.4) approach as provided by the OMG group, and it is compatible with the OWL-DL ontology language.
- It can also connect with the DBE Knowledge Base Service, a MOF-based ontology models central database using the Java Metadata Interface (JMI 1.0), in order to store, retrieve and update ontology models.

2. User Requirements Analysis

The Ontology Analysis Tool is a graphical, UML-like editor allowing for managing business models. It supports the ODM, an Ontology Definition Metamodel compatible with the OWL-DL [1] for description and representation of the ontology models. The users can create, edit and manage their ontologies in a fully interactive graphical environment and save them in the local file system of the application or in the DBE Knowledge Base Service that provides the ontology models persistency layer.

In this section, the functionality and the user requirements of the Ontology Analysis Tool will be described using the UML 2.0 Use Cases diagrams. Use Cases is a software engineering methodology that provides a standard and comprehensive representation of the functionality of a system from a user's perspective. Each use case may contain one or more scenarios that convey how the system should interact with the end user or another system to achieve a specific business goal.

A UML Use Case diagram that describes a scenario is made up of three elements: **Actors**, **Use Cases**, and **Associations** between actors and use cases.

- **Actors**. The actors represent the users of the system and can be humans, other computers, pieces of hardware, or even other software systems. Each actor can participate in one or more use cases. The actors may be defined in generalization hierarchies, in which an abstract actor description is shared and augment by one or more specific actor descriptions. An actor is shown as a small stick person in a Use Case diagram
- **Use Cases**. The use case represents a single task that the system needs to carry out. Each use case is a coherent unit and expressed by sequences of messages exchanged by the system and one or more actors of the system. The purpose of a use case is to define a piece of coherent behaviour without revealing the internal structure of the system. A use case is shown as an ellipse in a Use Case diagram.
- **Use Cases and Actors Associations**. The associations are the links drawn between uses cases or actors and use cases in a use cases scenario. They represent which actors interact with the system to complete a task represented by a use case as well as the relationships between the use cases. The associations supported by the UML 2.0 use cases are the following:
    - **Include relationship**: Although each use case instance is independent, the description of a use case can be factored into other, simpler use cases. The use case can simply incorporate the behaviour of other use cases as fragments of its own behaviour. In this case, the new use case is not a special case of the original case and cannot substituted for it.
    - **Extend relationship**: A use case can be defines as an incremental extension to a base use case. Each use case may have several extensions that may all be applied together. The extensions to a base use case add to its semantics: it is the base use case that is instantiated, not the extension use cases.
    - **Use Case Generalization:** A use case can be specialized into one or more child use cases. Any child may be used in a situation in which the parent use case is expected.
    - **Actors Generalization:** The actors may be defined in generalization hierarchies, in which an abstract actor description is shared and augment by one or more specific actor descriptions.

The associations are displayed as arrows between the related use cases or actors.
In the following figures a list of use cases scenarios are presented in order to describe the Ontology Analysis Tool functionality from the user's point of view.

The basic functionality of the Ontology Analysis Tool

The basic functionality of the Ontology Analysis tool includes the following use cases: create a new ontology model, edit the ontology model, save an ontology model, open an existing ontology model, remove an ontology model and export an ontology model in other formats as shown in the figure below.

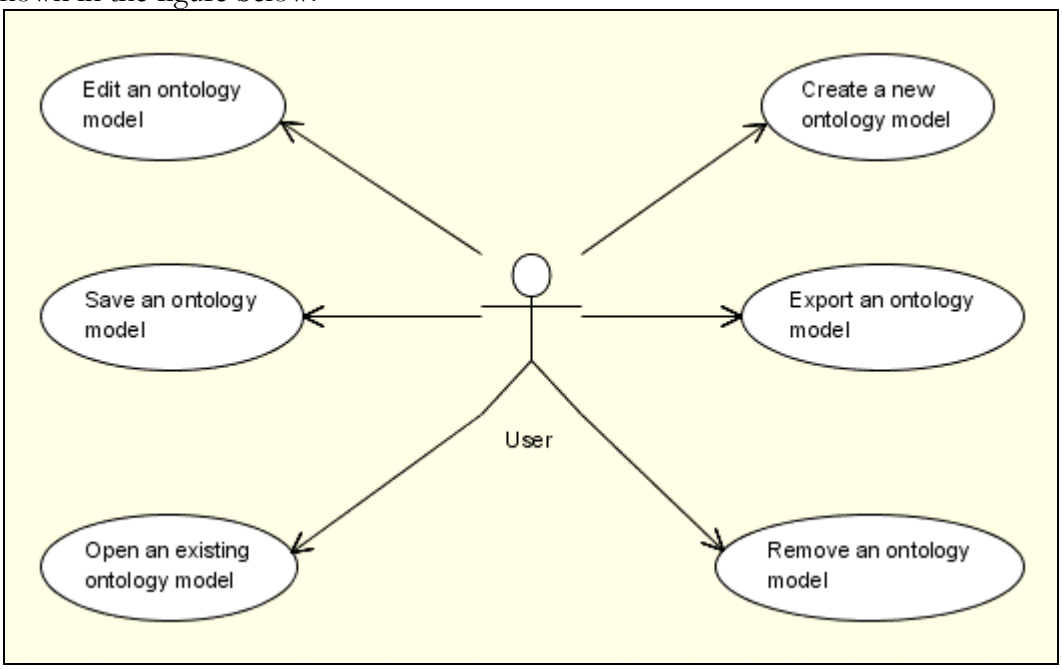

Figure 2: The main functionality of the Ontology Analysis Tool

Create a new ontology model

The "create a new ontology model" use case may be extended to simpler use cases such as create classes, diagrams, datatype properties, object properties, enumeration types, comments, individuals etc. as shown in the figures below.

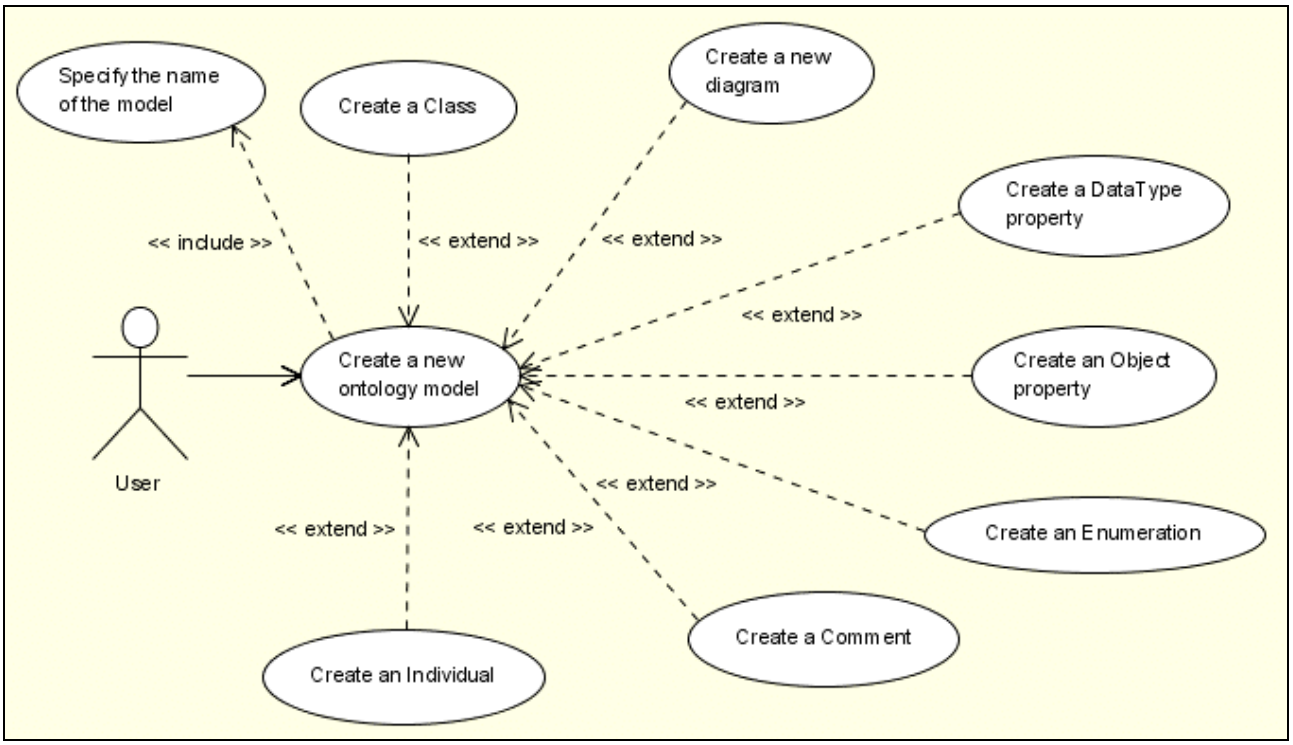

Figure 3: Create a new ontology model use case

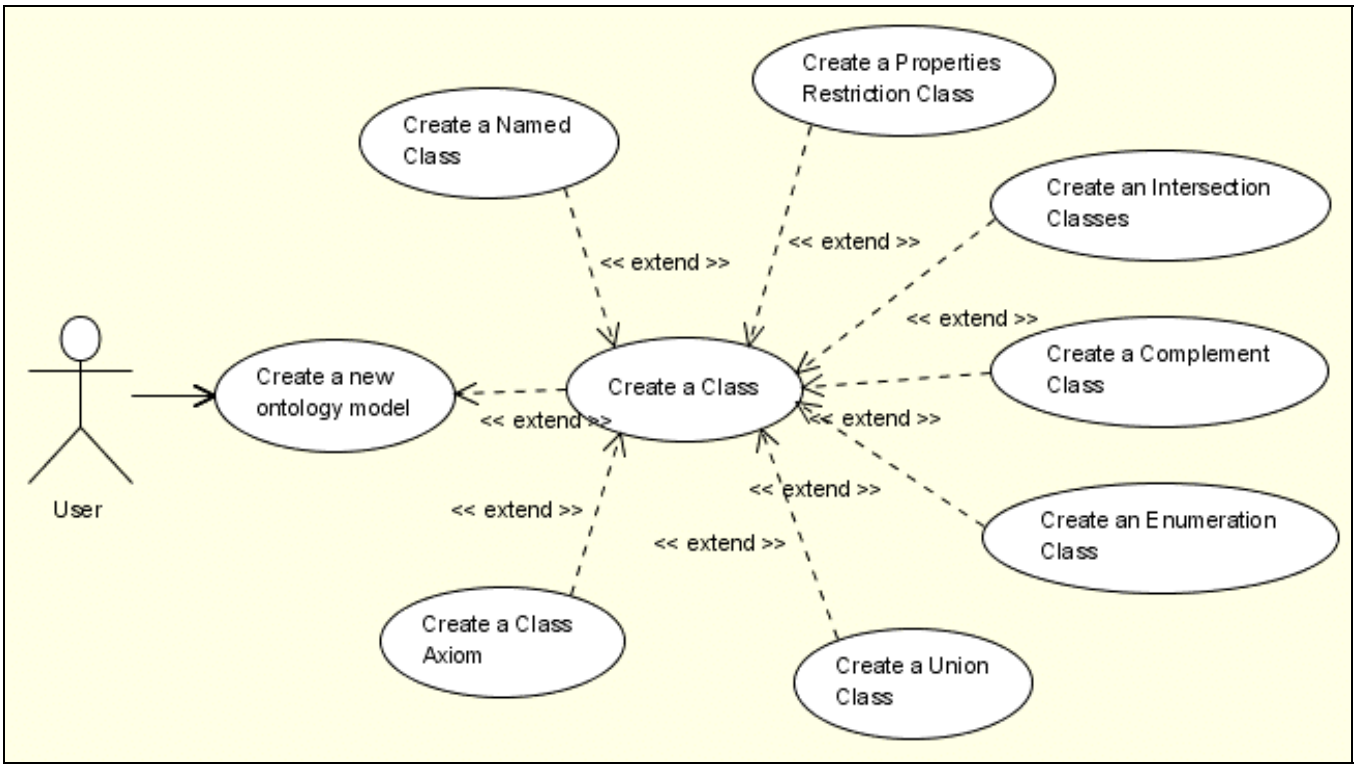

Figure 4: Create a Class use case

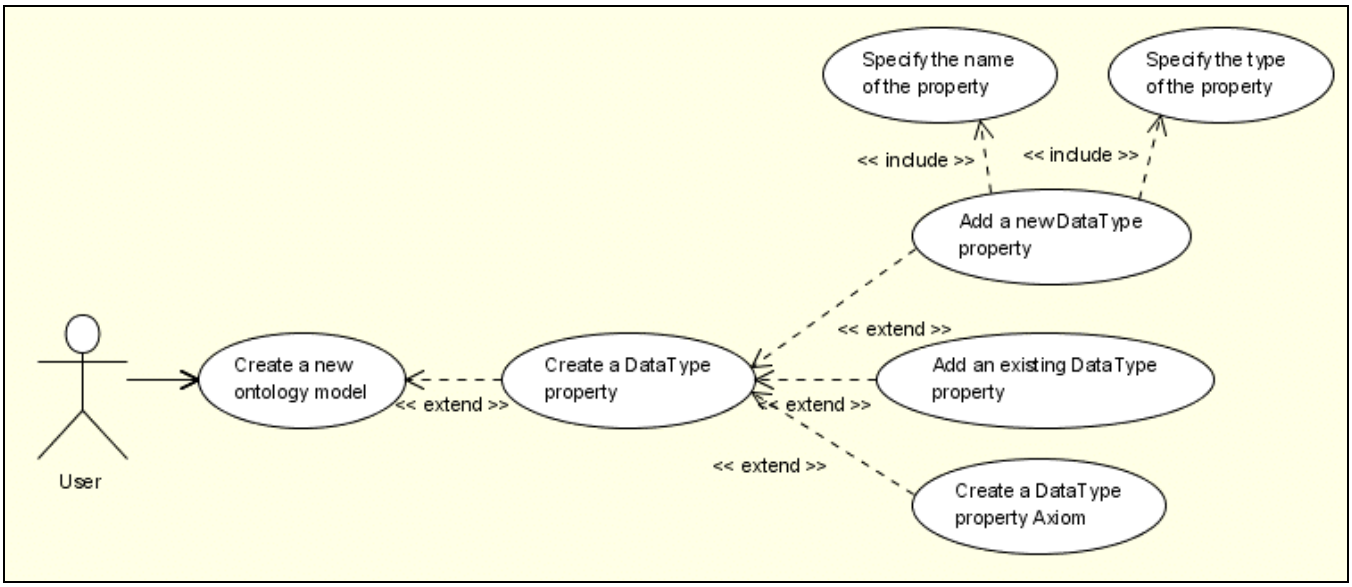

Figure 5: Create a DataType property use case

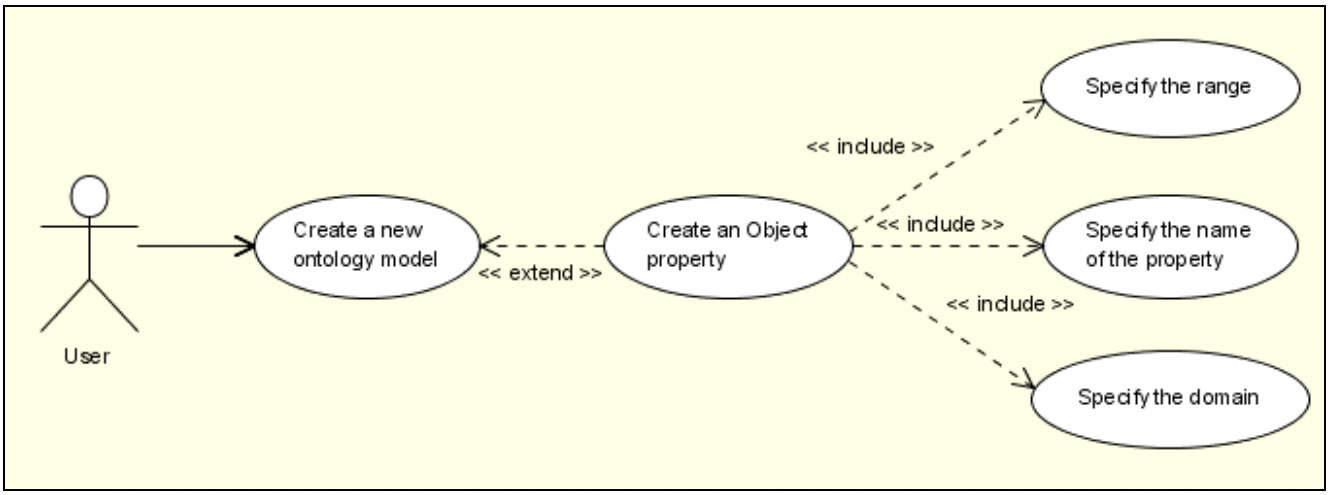

Figure 6: Create an object property use case

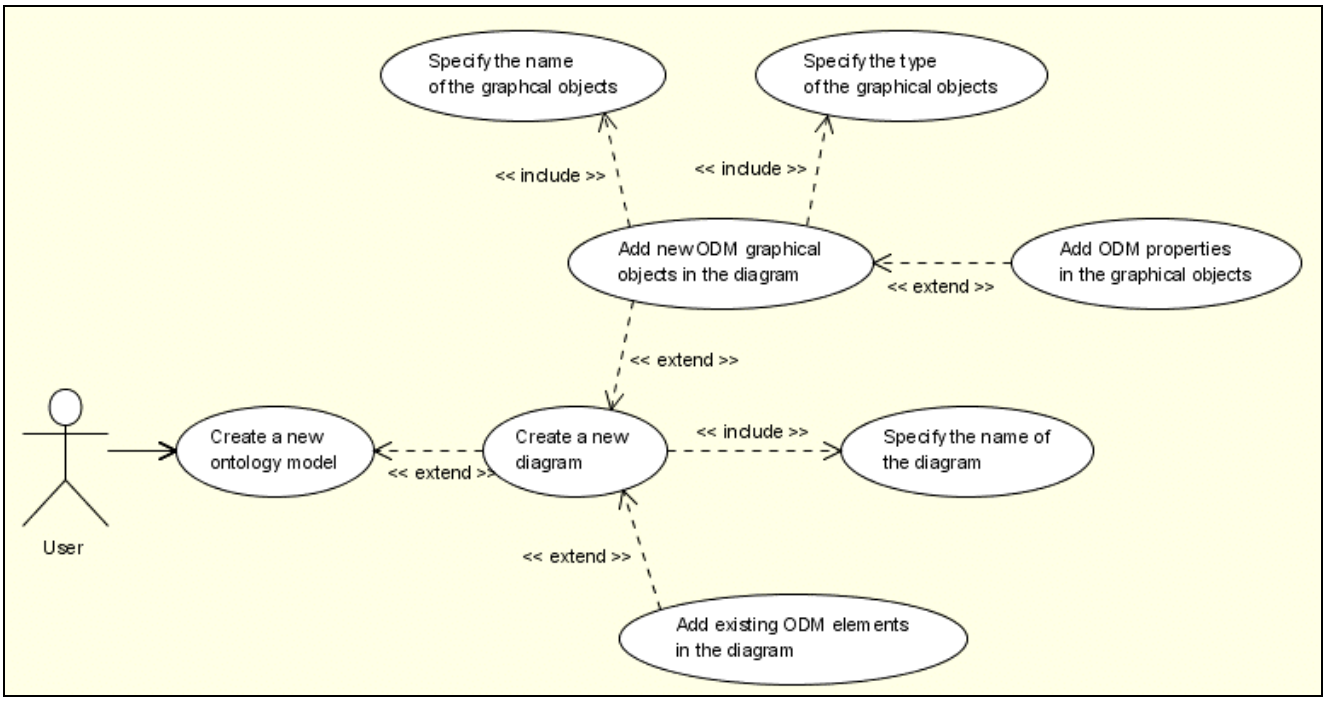

Figure 7: Create a diagram use case

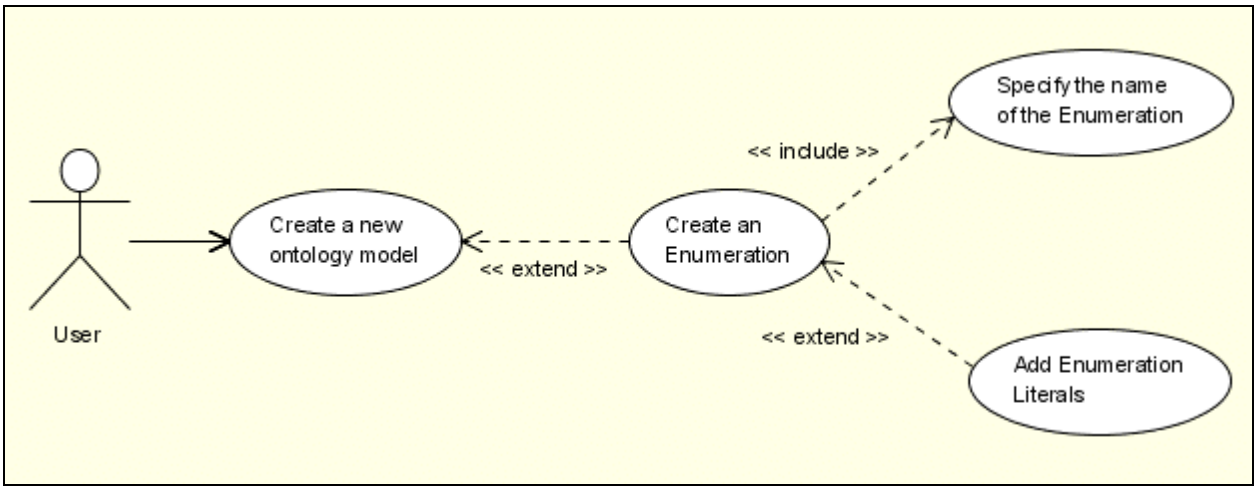

Figure 8: Create an Enumeration Type use case

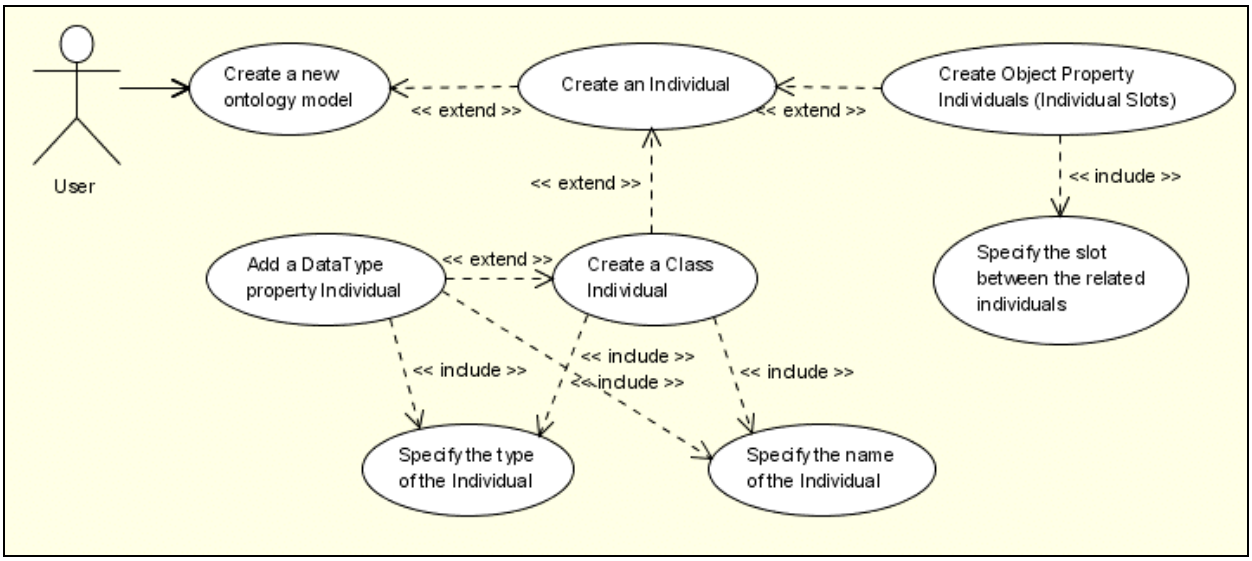

Figure 9: Create an individual use case

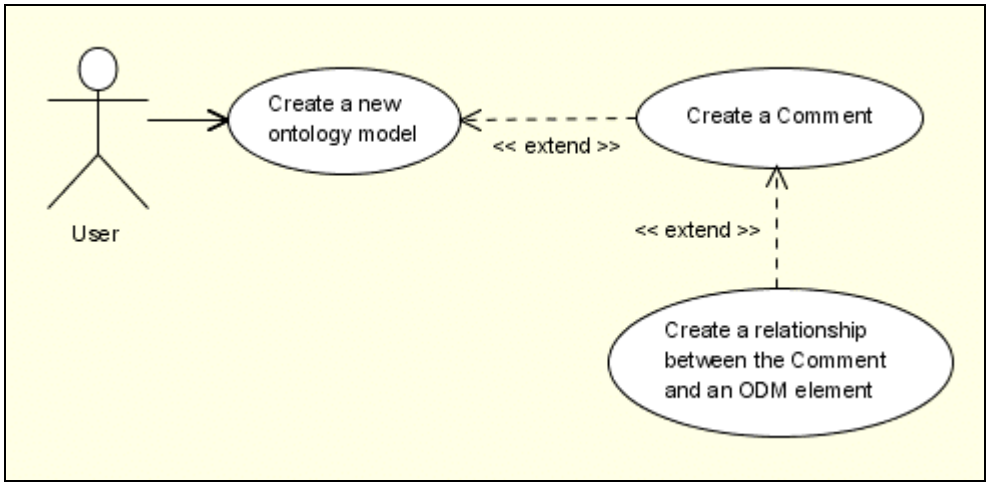

Figure 10: Create a Comment use case

Edit an ontology model

The editing of a new ontology model use case may be extended to simpler use cases such as edit ontology metadata, the ontology diagrams and the included ODM graphical objects, the ODM entities' properties, browse into the contents of the ontology, search about terms or element into the ontology etc. as shown in the figures below.

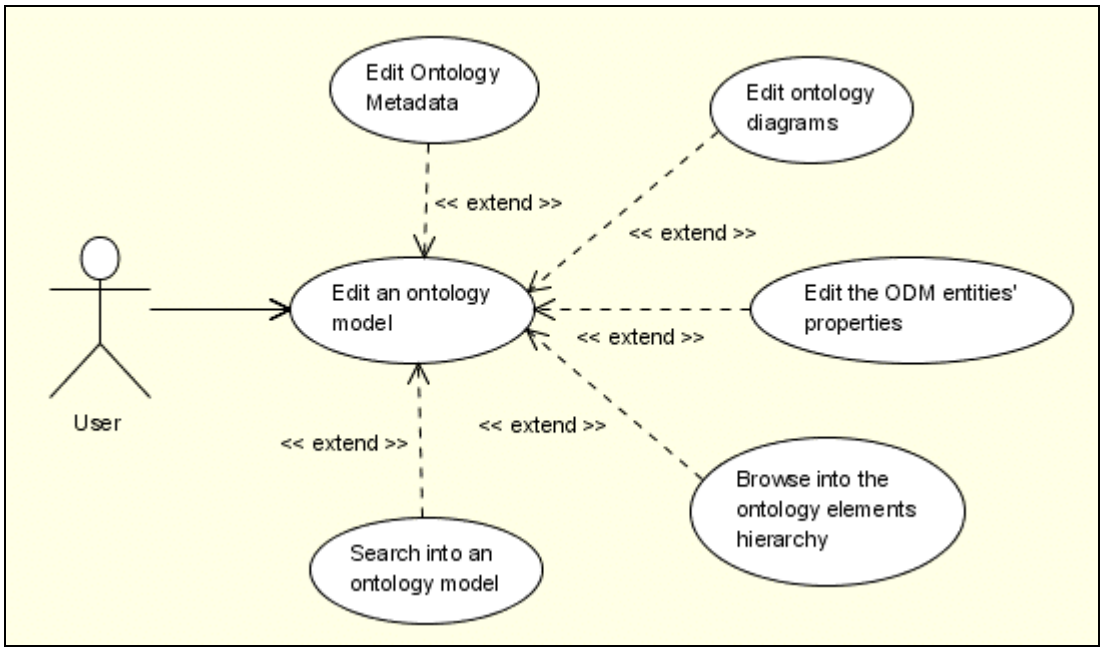

Figure 11: Edit an ontology model use case

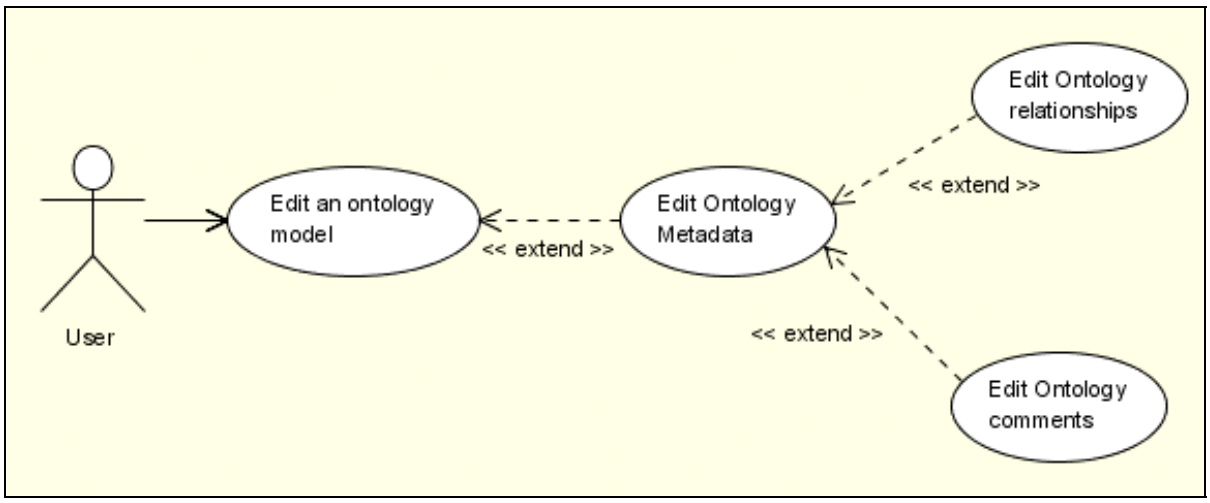

Figure 12: Edit ontology metadata use case

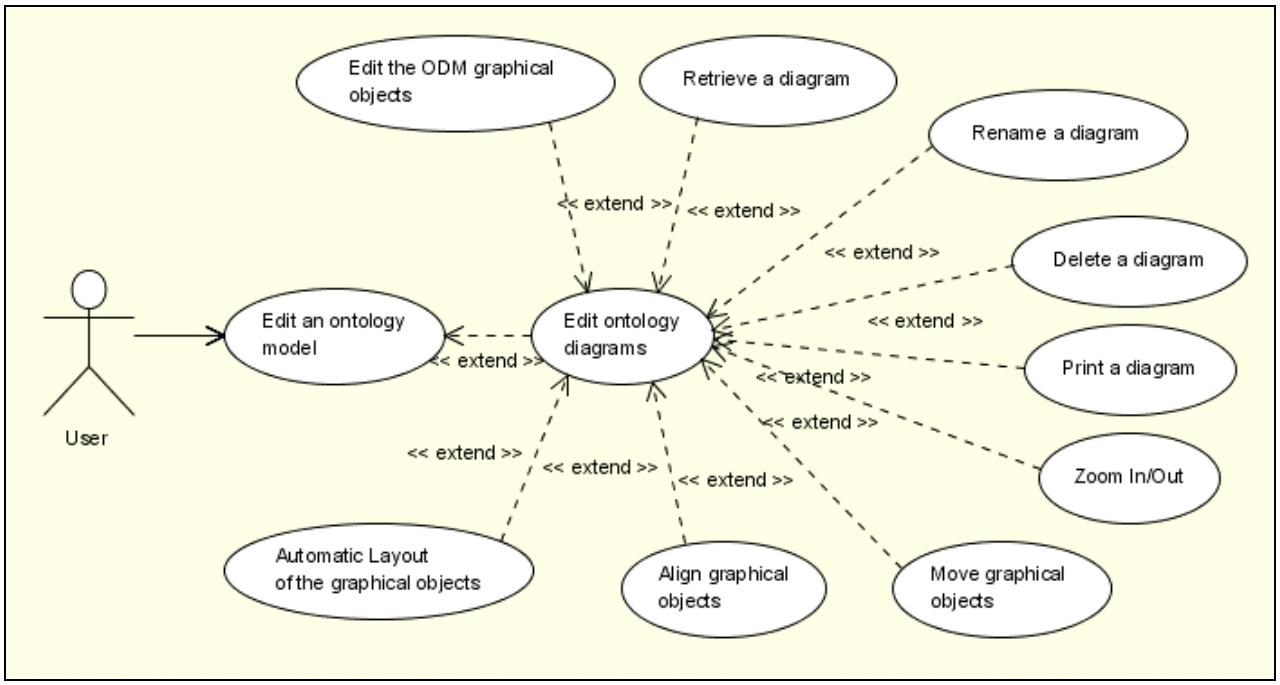

Figure 13: Edit ontology diagrams use case

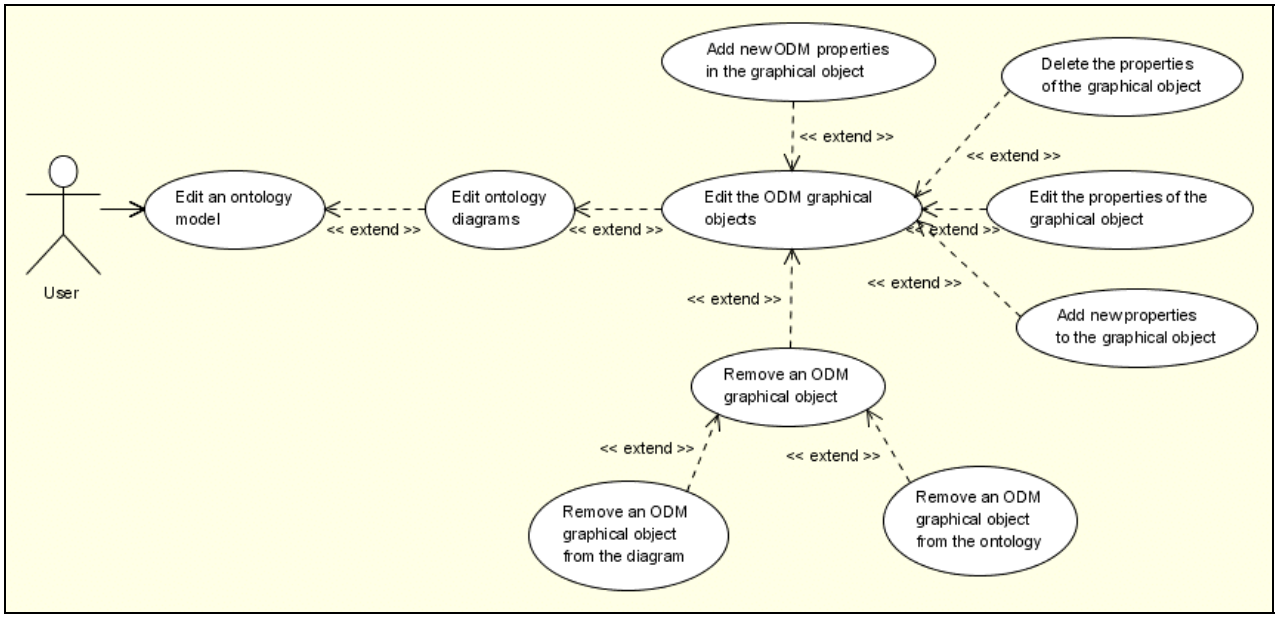

Figure 14: Edit the ODM graphical objects use case

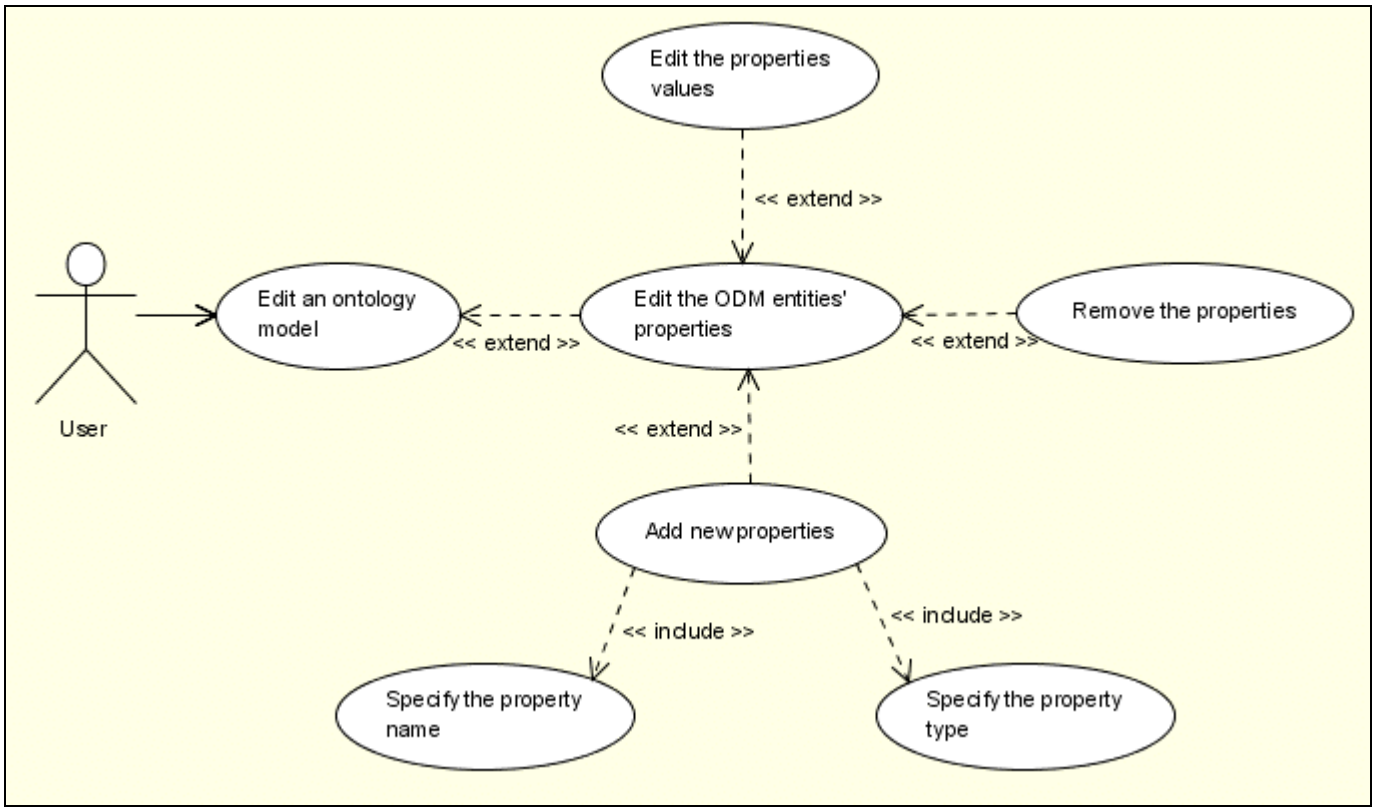

Figure 15: Edit the ODM entities' properties use case

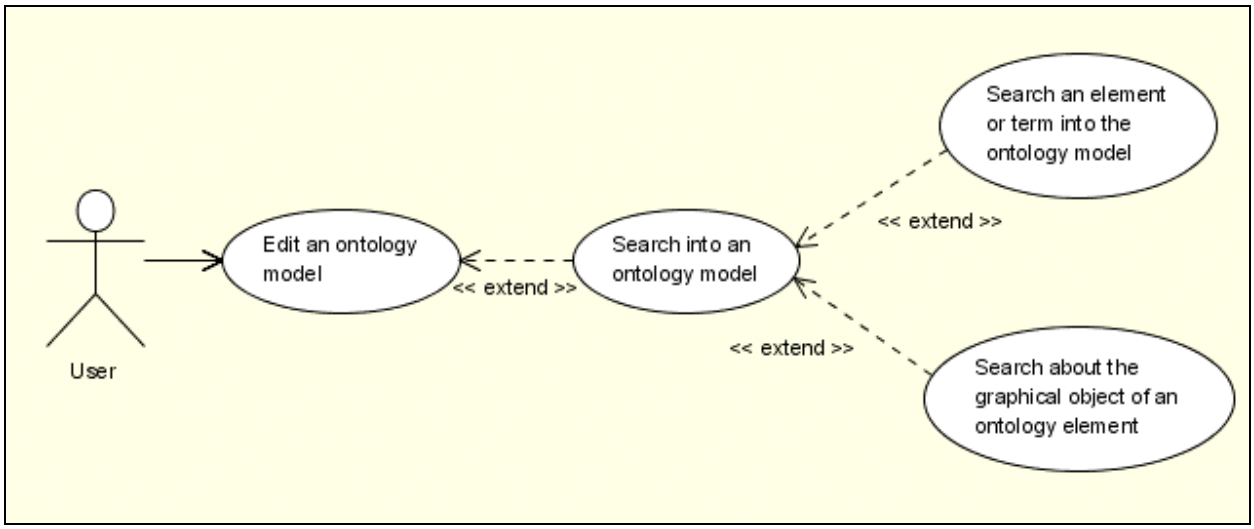

Figure 16: Search about an ontology element use case

Open an existing ontology model

The "open an existing ontology model" use case can be extended into two simpler use cases: open an ontology from the local file system or/and open an ontology from the DBE Knowledge Base, as shown in the figure below.

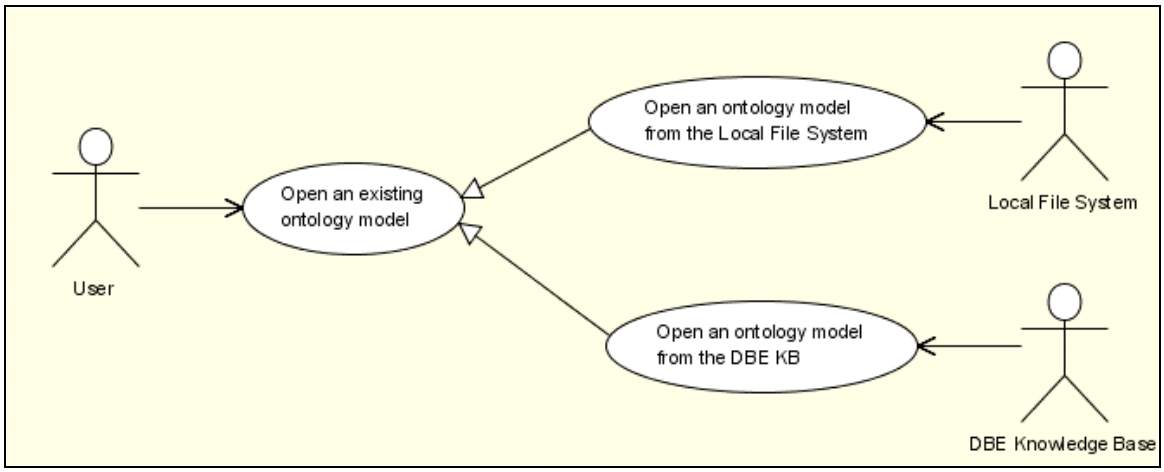

Figure 17: Open an existing ontology model use case

Save an ontology model

The "save an ontology model" use case can be extended into two simpler use cases: save the ontology in the local file system or/and save the ontology in the DBE Knowledge Base, as shown in the figure below.

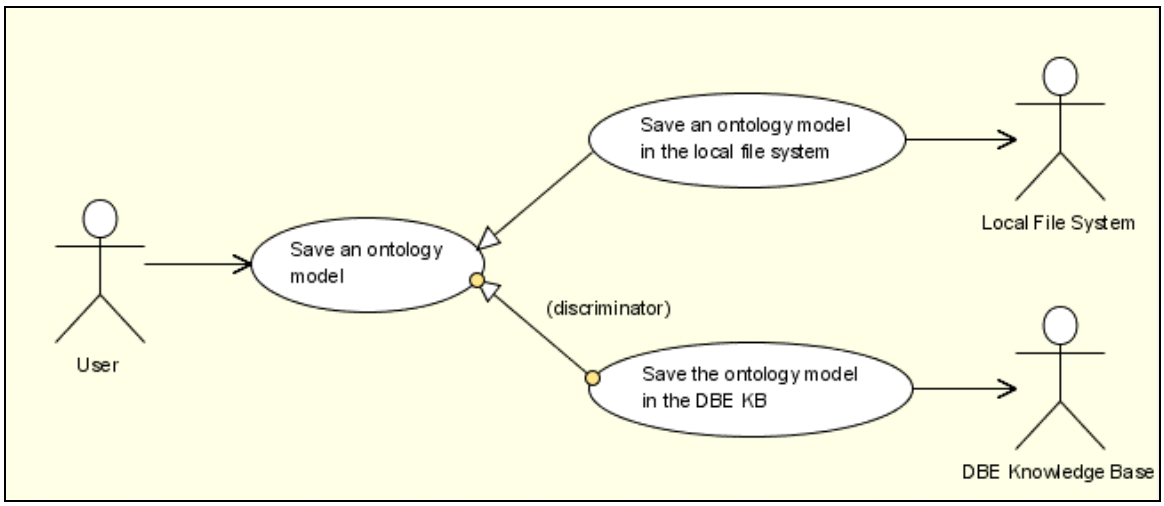

Figure 18: Save an ontology model use case

Export an ontology model

The "export an ontology model" use case can be extended into two simpler use cases: export the ontology in ODM-based XMI file or/and in an OWL-DL based XMI file, as shown in the figure below.

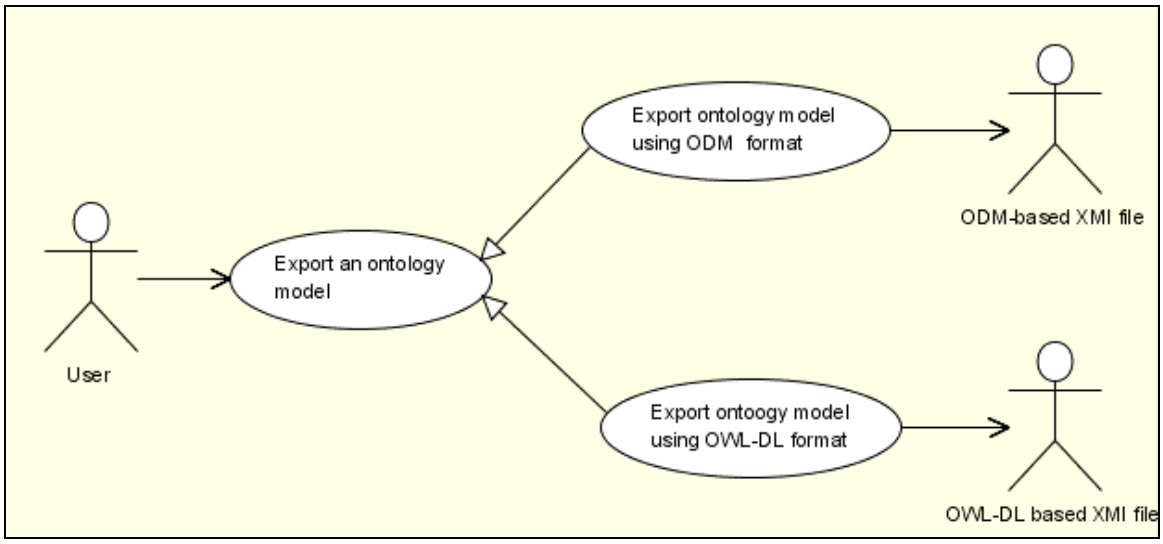

Figure 19: Export an ontology model use case

Remove an ontology model

The "export an ontology model" use case can be extended into two simpler use cases: remove the ontology from the local file system or/and from the DBE Knowledge Base, as shown in the figure below.

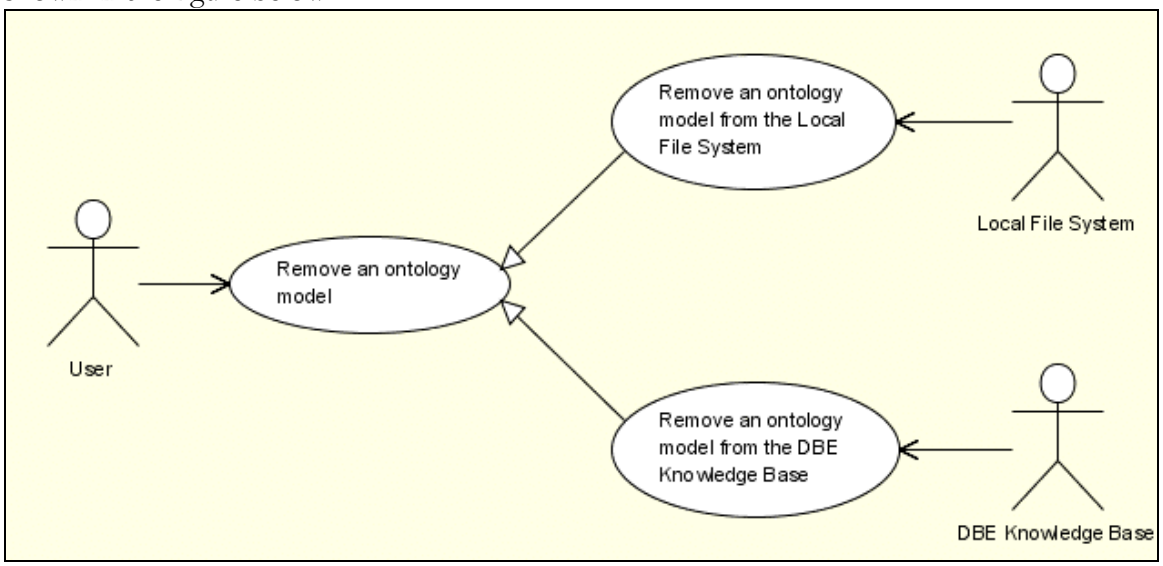

Figure 20: Remove an ontology model use case

Table 2 presents all the aforementioned use cases including information about their identity, the level, the primary actor, the goal and a short description of their actions. The template used is the one defined by Alistair Cockburn in his book titled "Writing Effective Use Cases" [18]. In short, the basic concepts of this template are the following:

- **Scope**: It refers to a system or a sub-component of the system where the use case occurs.
- **Goal**: It is the main objective of a use case.

- **Primary actor**: It refers to the user or the system that interacts with the Scope in order to proceed to the Goal of the use case.
- **Level**: It specifies the level (low or high, abstract or more specific) of the Goal in a use case.

The use cases can be classified in three categories depending on their Goal Level:

- **Primary Task or User-Task Level**: The Primary Task Level use case has a separate Goal and the Primary Actor can interact with the system in order to exclusively succeed it. It corresponds to elementary business processes in business process engineering.
- **Summary or Strategic Level**: The Summary Level use case comprises multiple primary-task level use cases in order to succeed a more complicated Goal. The Summary Level is more abstract than the Primary Task Level; it belongs to the highest level in the Use Cases Level hierarchy and provides a table of contents for the lower level use cases.

Sub-Function Level: The Goal of a Primary Task Level use case usually comprises many separate sub-goals. Each sub-goal can be represented using a Sub-Function Level use case. The Sub-Function Level is more specific than the Primary Task Level and belongs to the lowest level of the Use Cases Level hierarchy.

#	Level	Primary Actor	Goal	Brief
UC_1	Summary	User	Create an Ontology	The User wants to create an ontology based on the ODM model
UC_2	Primary Task	User	Create a New Ontology	The User wants to create a new ontology
UC_3	Primary Task	User	Edit Ontology Meta-Data	The User wants to edit the ontology meta-data as defined in the ODM
UC_4	Sub-function	User	Create a relationship between ontologies	The User wants to create a relationship between two or more ontologies
UC_5	Sub-function	User	Add Ontology Comments	The User wants to add comments about the ontology
UC_6	Primary Task	User	Manage Ontology Diagrams	The User wants to manage the diagrams of the ontology
UC_7	Sub-function	User	Create a Diagram	The User wants to create a new diagram in the ontology
UC_8	Sub-function	User	Rename a Diagram	The User wants to rename the name of a diagram in the ontology
UC-9	Sub-function	User	Delete a Diagram	The User wants to remove a diagram from the ontology
UC_10	Sub-function	User	Zoom-In/Zoom-Out	The User wants to zoom-in and zoom-out in the elements of a diagram
UC_11	Sub-function	User	Align Graphical Objects	The User wants to align some of the graphical objects in a diagram of the ontology
UC_12	Sub-function	User	Retrieve a diagram	The User wants to retrieve some of the ontology diagrams and display it in

				the working area.
UC_13	Sub-function	User	Print a Diagram	The User wants to send for printing a diagram of the ontology
UC_14	Sub-function	User	Automatic Layout of the graphical objects in a diagram	The User wants to apply an automatic layout on the graphical objects of a diagram
UC_15	Primary Task	User	Search in the Ontology	The User wants to search in the contents of an ontology
UC_16	Sub-function	User	Search an element in an Ontology	The User wants to search about an element that has been defined in the ontology
UC_17	Sub-function	User	Search the graphical object of an element in an Ontology	The User wants to search about the graphical object related to an element of the ontology
UC_18	Primary Task	User	Save the Ontology	The User wants to save the ontology
UC_19	Sub-function	User	Save the Ontology in the local file system	The User wants to save the ontology in the local file system
UC_20	Sub-function	User	Save the Ontology in the DBE Knowledge Base	The User wants to save the ontology in he DBE Knowledge Base
UC_21	Primary Task	User	Remove an Ontology	The User wants to remove an ontology
UC_22	Sub-function	User	Remove an Ontology from the local file system	The User wants to remove one or more ontologies from the local file system
UC_23	Sub-function	User	Remove an Ontology from the DBE Knowledge Base	The User wants to remove one or more ontologies from the DBE Knowledge Base
UC_24	Primary Task	User	Open an Ontology	The User wants to retrieve an existing ontology
UC_25	Sub-function	User	Open an Ontology from the local file system	The User wants to retrieve an existing ontology from the local file system
UC_26	Sub-function	User	Open an Ontology from the DBE Knowledge Base	The User wants to retrieve an existing ontology from the DBE Knowledge Base
UC_27	Primary Task	User	Export Ontology	The User wants to export a newly created ontology in another format
UC_28	Sub-function	User	Export the Ontology as ODM XMI	The User wants to export a newly created ontology as a file in a valid standard XMI representation
UC_29	Sub-function	User	Export the Ontology as OWL-DL	The User wants to export a newly created ontology as a file in a valid OWL-DL language representation

UC_30	Primary Task	User	Import Ontology	The User wants to import an existing ontology represented in another format
UC_31	Sub-function	User	Import an Ontology in ODM XMI	The User wants to import an ontology that exists as a file in a valid standard XMI representation
UC_32	Sub-function	User	Import an Ontology in OWL-DL	The User wants to import an ontology that exists as a file in a valid OWL-DL language representation
UC_33	Primary Task	User	Create a Class	The User wants to create a new ODM Class in the ontology
UC_34	Sub-function	User	Create a Named Class	The User wants to create a new ODM Named Class in the ontology
UC_35	Sub-function	User	Create a Union Class	The User wants to create a new ODM Union Class in the ontology
UC_36	Sub-function	User	Create an Intersection Class	The User wants to create a new ODM Intersection Class in the ontology
UC_37	Sub-function	User	Create a Complement Class	The User wants to create a new ODM Complement Class in the ontology
UC_38	Sub-function	User	Create an Enumeration Class	The User wants to create a new ODM Enumeration in the ontology
UC_39	Sub-function	User	Create a Property Restriction	The User wants to define a restriction on an ODM Class Property
UC_40	Sub-function	User	Create a Class Axiom	The User wants to create an ODM Class Axiom
UC_41	Primary Task	User	Create a Data Type Property	The User wants to add an ODM Data Type Property in an ODM Class
UC_42	Sub-function	User	Add a new Data Type Property	The User wants to create a new ODM Data Type Property in an ODM Class
UC_43	Sub-function	User	Add an existing Data Type Property	The User wants to add an existing ODM Data Type Property in an ODM Class
UC_44	Sub-function	User	Create a Data Type Property Axiom	The User wants to create an ODM Data Type Property Axiom
UC_45	Primary Task	User	Create an Object Property	The User wants to create an ODM Object Property
UC_46	Sub-function	User	Add an Object Property	The User wants to add an ODM Object Property in a Class
UC_47	Sub-function	User	Define Object Property Type	The User wants to define the type of an ODM Object Property
UC_48	Primary Task	User	Create an Enumeration	The User wants to create an ODM Enumeration

UC_49	Sub-function	User	Create an Enumeration Literal	The User wants to create an ODM Enumeration Literal
UC_50	Primary Task	User	Create a Graphical Comment	The User wants to add a Graphical Comment about an ODM Object in a diagram
UC_51	Sub-function	User	Create an relationship between a Comment and an ODM Object	The User wants to create a relationship between a Comment and an ODM object
UC_52	Primary Task	User	Create an Individual	The User wants to create an ODM Individual
UC_53	Sub-function	User	Create a Class Individual	The User wants to create an ODM Class Individual
UC_54	Sub-function	User	Create an Data Type Property Individual	The User wants to create an ODM Data Type Property Individual
UC_55	Sub-function	User	Create an Object Property Individual	The User wants to create an ODM Object Property Individual
UC_56	Primary Task	User	Remove an element from an Ontology	The User wants to remove an element from the ontology
UC_57	Sub-function	User	Validation and Consistency Checking	The system makes validation and consistency checking before any creation or removal action in the ontology
UC_58	Sub-function	User	Remove an element from a diagram	The User wants to remove an element from a diagram. The deletion can be triggered by the system too.
UC_59	Primary Task	User	Add an existing element in the diagram	The User wants to add an existing element of the ontology in a diagram
UC_60	Primary Task	User	Edit the properties of an element	The User wants to edit the properties of an element as they defined in the ODM

Table 2: Use Cases Table (using Cockburn's template)

3. The Architecture of the Ontology Analysis Tool

The Ontology Analysis Tool is a graphical UML-like editor that enables business analysts to create and manage business models using the ODM specification.
The adoption of Eclipse as the main platform for the Service Factory Environment in DBE posed the technical requirement of developing the Ontology Analysis Tool as an Eclipse plug-in that will be plugged as all other tools into the same platform (DBE Studio).
As a consequence, the Ontology Analysis Tool is being developed as an Eclipse plug-in based on Eclipse 3.1 [2], and exploiting of the GEF 3.1 [3] and Draw 2D graphical frameworks that have been built for this platform.

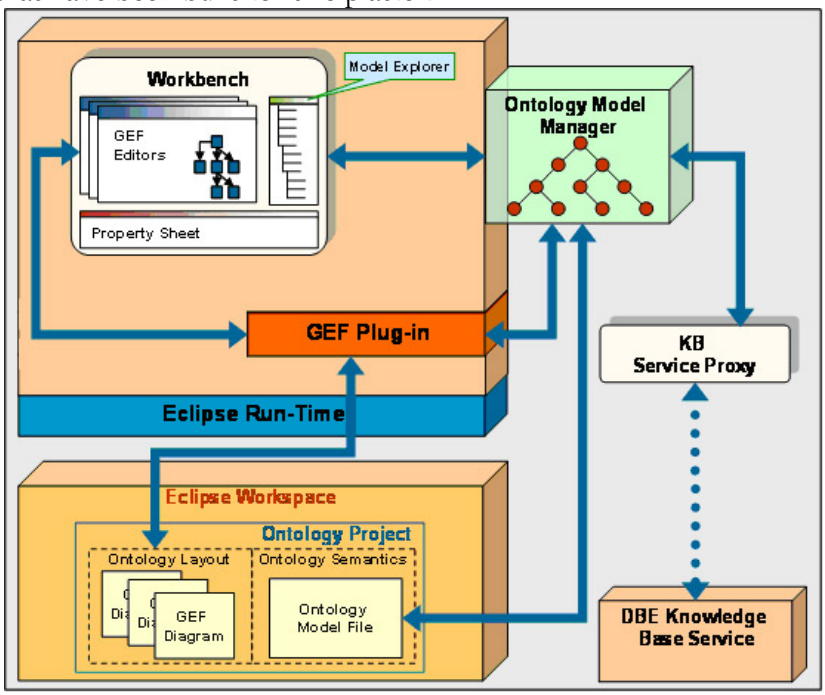

Figure 21: The architecture of the Ontology Analysis Tool

From an architectural point of view, the Ontology Analysis Tool exploits the DBE Knowledge Base for its persistency requirements. Service-Oriented architecture is followed for integrating the Ontology Analysis Tool with the DBE Knowledge Base using the DBE ServENT platform. The Ontology Analysis Tool connects with a DBE ServENT (the KB Service Proxy) using a specific JMI interface and looks up into the network to find an appropriate DBE Knowledge Base (KB) Service. The Ontology Analysis Tool enables the creation of ontologies and using the functionality of the KB-service is able to store, retrieve and update ontologies in the KB.
The Ontology Analysis Tool architecture comprises the following components:

- The Eclipse Platform
- The Eclipse Workbench
- The Eclipse Workspace and the Ontology Files Model
- The Ontology Model Manager
- The DBE ServENT (DBE ProXY Service)
- The DBE Knowledge Base Service

3.1. The Eclipse Platform

It is the core component of the architecture and provides the development and the run-time environment of the Ontology Analysis Tool. The Development Environment provides the general working environment (workbench), the workspace area, the capability to manage graphical objects using the GEF plug-in, a basic menu, help wizards etc. On the other hand, the Eclipse Run-Time Environment is responsible to detect the Ontology Analysis Tool Plug In, to interpret its file manifest and register the Ontology Analysis Tool as a new Plug-In in the Eclipse platform.

3.2. The Eclipse Workbench

This component provides the graphical user interface (GUI) of the Eclipse Platform. The graphical environment of the Ontology Analysis Tool is based on the Eclipse Workbench and consists of its own Eclipse Perspective. The Perspective contains a set of special views, editors, menus and toolbars located in specific positions of the graphical environment. The uppermost sub-components of the graphical environment are the following:

- **The Model Explorer:** This component is responsible for the presentation of the ontology semantics as defined by the user. It also provides a searching and browsing mechanism in the content of the ontology. The ontology elements are displayed in a tree-view hierarchy using the JFace toolkit. They are located as nodes in the first level as well as the element properties and relationships in the second level of the tree hierarchy.
- **The Graphical Diagrams Editors:** The ontologies semantics can be presented as graphical objects in graphical diagrams using a UML-like user interface. Every diagram is a separate GEF editor that contains a group of graphical objects; it is responsible for the management and editing of these objects and acts independently from the other editors. The Ontology Analysis Tool manages the separate diagrams as a whole and provides a synchronization mechanism between the GEF diagram editors and the Model Explorer.
- **The Properties Sheet**: This component is responsible for the presentation and editing of the properties' values of the ontology elements. The properties are explicitly defined in the ODM model for each ontology element. The synchronization mechanism keeps the Properties Sheet up-to-date in any changes happened in the Model Explorer or the Graphical Diagrams Editors.

The above sub-components have been implemented using the GEF plug-in, the Standard Widget Toolkit and the JFace Toolkit.

3.3. The Eclipse Workspace and the Ontology Files Model

The Eclipse Workspace is the local file area where the Eclipse platform stores the projects and the files created during the usage of the Eclipse platform. In case of the Ontology Analysis Tool, each new ontology corresponds to a new Eclipse project.

Each project creates its own folder inside the default workspace of the Eclipse Platform. The folder name is the same as the ontology name, which specified by the user. In the project's folder the following files are created:

- An initial ".project" file in XML format that contains information about the project and the platform.
- A "MainMemory.memory" file in binary format that contains information about the ontology semantics.
- A list of "%DiagramName%.diagram" files in binary format that contain information about the ontology layout and the graphical objects for each ontology diagram.

3.4. The Ontology Model Manager

It is the most important component of the system because it is responsible for the management of the whole ontology processing. Each action related to the processing of the active ontology and any interaction between the various components of the system is explicitly managed by the Ontology Model Manager. The component also provides all the appropriate data structures in order to manage the ontology semantics as they are displayed in the Model Explorer.

3.5. The DBE ServENT (DBE ProXY Service)

This service provides a ProXY mechanism that enables the Ontology Analysis Tool to look up into the network in order to find an appropriate DBE Knowledge Base (KB) Service and obtain access to the functionality it provides.

3.6. The DBE Knowledge Base Service

This service provides an API based on the JMI standard that enables Ontology Analysis Tool to access all the functionality of the DBE Knowledge Base as described below:

- Save an ontology in the DBE Knowledge Base
- Browse and retrieve the ontologies saved in the DBE Knowledge Base
- Export ontology in a XMI file using the ODM specification.

3.7. The Architecture Components Interaction

This section contains a comprehensive description of the interactions between the key components of the Ontology Analysis Tool.

The Ontology Model Manager is the core component of the architecture because it manages any interaction occurs between the various components of the system. It provides a synchronization mechanism between the diagrams (GEF Editors) of the ontology, the Ontology Model Explorer (the JFace tree-viewer that displays all the ontology semantics and elements in a tree hierarchy mode) and the Properties Sheet. Any change happens in one of the three components is captured by the Ontology Model Manager and is automatically propagated to the other two synchronized components.

For example, each time a user selects a graphical object in a diagram of the ontology, the GEF editor communicates with the Ontology Model Manager in order to update the Ontology Model Explorer and highlight the element of the ontology in the JFace tree-viewer that is represented by the selecting graphical object.

In case a user changes any property's value of a graphical object via direct editing, the GEF editor communicates with the Ontology Model Manager, which updates the Ontology Model Explorer in order to refresh the value of the selected property in the Properties Sheet. The same procedure happens when a new element (a class, an object property, a data type property, an individual etc.) is added in a diagram or an exiting element is selected for editing.

In case a user selects an ontology element in the JFace tree-view, the Ontology Model Explorer communicates with the Ontology Model Manager, which searches the diagrams list; it displays the first of the ontology diagrams that contain the element in a GEF editor and highlights the corresponding graphical object. The Ontology Model Explorer also opens a Properties Sheet with the properties' values of the selected element.

In a similar way, when a user changes the value of an element's property from the Properties Sheet, the Ontology Model Manager takes over to update the JFace tree-view of the

Ontology Model Explorer with the new value as well as the graphical objects in all diagrams (opened or not) of the ontology that represent the selected element.
During opening any ontology from the local file system, the Ontology Model Manager retrieves the content of the ontology from the ontology model files and forwards it to the JFace tree-viewer of the Ontology Model Explorer for editing.
Finally, when the Ontology Analysis Tool tries to save or open any ontology from the DBE Knowledge Base, the Ontology Model Manager sends a request to the DBE ServENT ProXY Service, which establishes a communication with the DBE Knowledge Base Service. This service takes over to save the ontology in the Knowledge Base or retrieve the ontology from the Knowledge Base and send it back to the Ontology Model Manager correspondingly.

4. The User Interface of the Ontology Analysis Tool

The Ontology Analysis Tool has been developed as an Eclipse plug-in into the DBE Studio environment. The users can activate the Ontology Analysis Tool by selecting the DBE Ontology Analysis perspective, from the Eclipse main menu, **Window->Open Perspective->Other-> DBE Ontology Analysis**.

By default, the Ontology Analysis Tool Perspective creates, edit and save ontologies in the local file system or attempts to connect to the address http://localhost:2728 assuming that there is a running instance of the DBE ServENT, in order to connect to the appropriate DBE Knowledge Base Service and retrieve the list of ontologies that have been created for a specific SME. The area of the local file system used by the Ontology Analysis Tool is identical to the Eclipse workspace.

Alternatively, users can specify explicitly the URI of the DBE ServENT service as well as the SME id, a unique identifier for an SME, by selecting Window->Preferences->DBE Ontology Analysis from the Eclipse menu, as shown in the figure below.

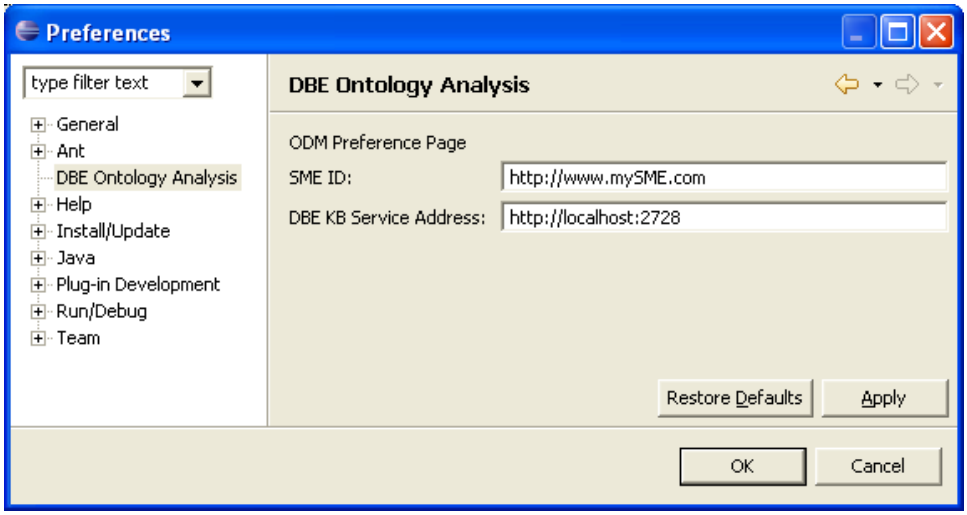

Figure 22: The Ontology Analysis Tool Preference page

4.1. The Ontology Analysis Tool Perspective

The Ontology Analysis Tool Perspective is activated by selecting Window->Open Perspective->Other->DBE Ontology Analysis perspective from the Eclipse menu. The perspective comprises the following components as shown in Figure 23:

- **The Model Tree View Explorer**: The model tree view explorer is located in the Outline View, in the left side of the user interface and represents the active ontology model in a tree hierarchy manner. Users can navigate inside the tree hierarchy and select any of the ODM elements of the active ontology. By clicking in an ontology element, the properties' values of the element are displayed in the Property Sheet of the Ontology Analysis Tool, and the graphical representation of the ODM element is highlighted in the ontology diagram that includes it. The Outline View also contains a toolbar and a menu bar with the basic functionality of the Ontology Analysis Tool. It enables users to create new ontologies and diagrams, open, save and delete ontologies in the local file system or in the DBE knowledge base and export ontologies in text files using ODM or OWL-DL specification.

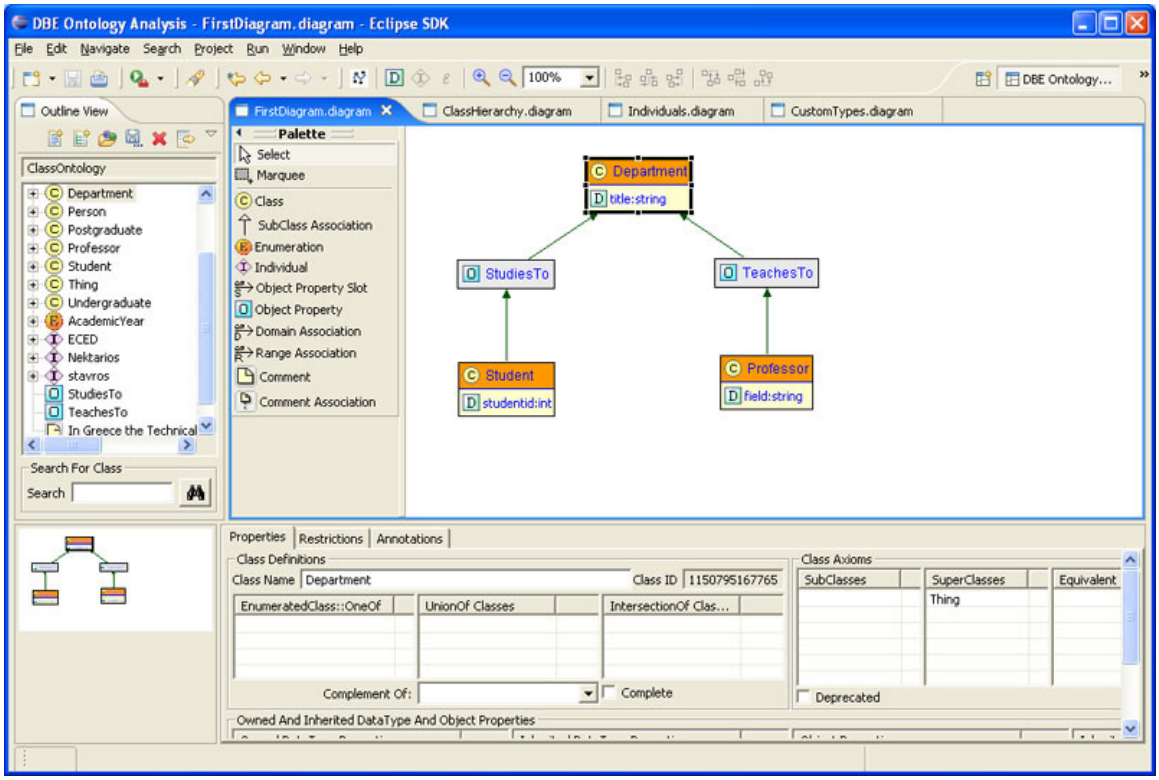

Figure 23: The Ontology Analysis Tool Perspective

- **The diagrams editor (GEF Editor)**: The diagrams editor is the main working area located in the centre of the User Interface and it is responsible for the manipulation of the diagrams of the active ontology. It provides a graphical UI environment and a tools palette with all the ODM model elements. Users can select an element from the editor's palette in order to add it in the active ontology's diagrams or "drag-and–drop" an existing ontology element from the tree-view into the graphical editor. The synchronization mechanism of the Ontology Analysis Tool enables it to propagate any changes occurred in the graphical editor to the contents of the Model Tree View Explorer and the Properties Sheet.
- **The Ontology Analysis toolbar**: The Ontology Analysis Tool adds a new toolbar in the Eclipse environment in order to provide some actions that applied to the content of the graphical editor environment, such as zoom in-out, alignment, automatic layout, activation, selection actions etc.
- **A Properties Sheet**: The Properties Sheet is located in the bottom side of the user interface and presents the properties' values of the ontology elements. There is a synchronization mechanism between the graphical editor and the Properties Sheet. So any changes in the graphical editor update the contents of the Properties sheet and vice versa.
- **A Search Utility**: The Search Utility enables users to search about terms and entities that have been defined in the active ontology. The results of the searching are filtered and displayed into the Model Explorer tree view.
- **The Outline Diagrams View**: It is a region in the left-bottom corner of the user interface that represents the diagrams of the active ontology in a small scale. In this way,

the users can easily navigate into large-side diagrams by scrolling up and down in the minimized diagram using a "thumbnail".

- **A Status Bar**: In the status bar of the Ontology Analysis Tool some important information and error messages are displayed during the usage of the Ontology Analysis Tool.

4.2. The Outline View and the Model Tree View Explorer

The Outline View is located in left side of the user interface and contains a toolbar and a menu bar that supports the following functionalities:

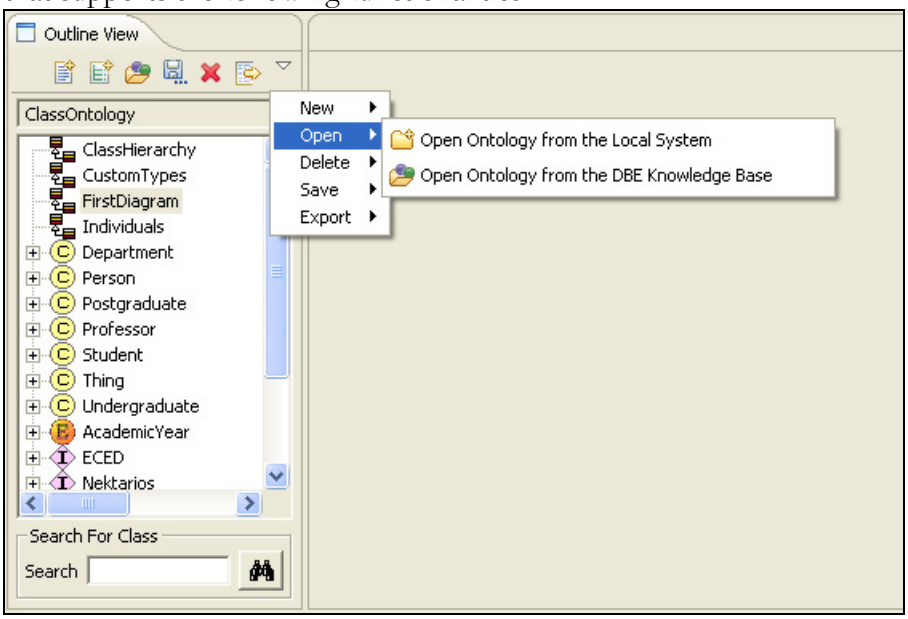

Figure 24: The Ontology Analysis Tool menu

- **Create a new ontology**. Users can create a new ontology by selecting **New -> New Ontology** from the Outline View menu and specifying the ontology name as shown in Figure 25. The Ontology Analysis Tool deploys a new Eclipse project for the ontology into the Eclipse workspace and it creates a directory with the same name as the ontology name. The ontology model can be saved locally as a binary file with name "MainMemory.memory" into the directory.

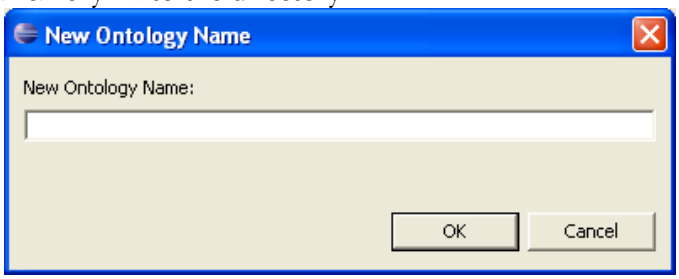

Figure 25: Create a new ontology

- **Create a new diagram in the active ontology**. Users can add a new diagram in the active ontology by selecting **New -> New Diagram** from the Outline View menu and specifying the diagram name as shown in the figure below. The Ontology Analysis Tool

can save locally the ontology diagrams as binary files with name "%DiagramName%.diagram" into the ontology directory.

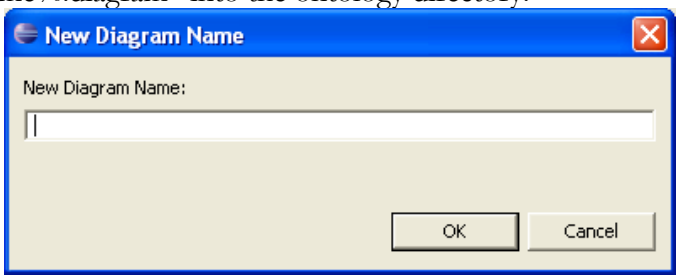

Figure 26: Create a new diagram

- **Open ontology from the local file system and/or the DBE Knowledge base**. The Ontology Analysis Tool manages ODM-based ontology models that have been saved in the local file system as well as in the DBE knowledge base. Users can open an ontology from the local file system by selecting **Open -> Open Ontology from the local system** from the Outline View menu and specifying the ontology name as shown in the figure below.

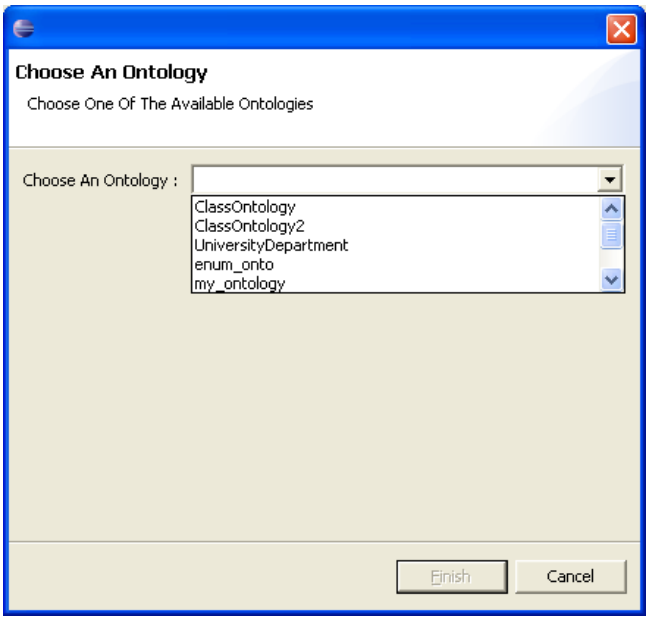

Figure 27: Open an ontology from the local file system

Alternatively users can select to open an ontology from the DBE Knowledge Base by selecting **Open -> Open Ontology from the DBE Knowledge Base** and specifying the ontology name as shown in the figure below.

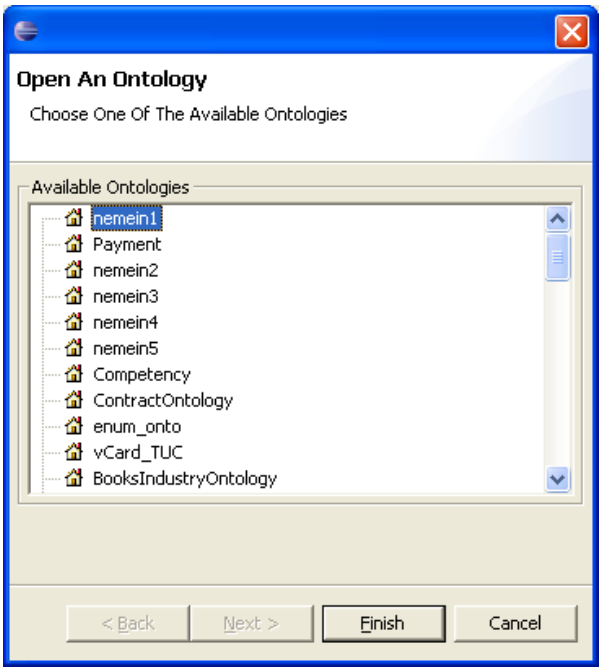

Figure 28: Open an ontology from the DBE Knowledge Base

- **Save a diagram or all diagrams of an ontology locally.** Users can save the active ontology model and its diagrams locally, according to Ontology Files Model in a binary file format**,** by selecting **Save -> Save Diagram in the local system** or **Save -> Save all diagrams in the local system** from the Outline View menu correspondently.
- **Save ontology in the DBE Knowledge base**. Users can also save the active ontology and all its diagrams in the DBE knowledge base as specified by the Preferences, by selecting **Save-> Save Ontology in the DBE Knowledge Base** from the Outline View menu. Diagrams are saved in the KB in a binary format and the ontology model in an ODM-based XMI 1.2 format.
- **Export the ontology in ODM-based XMI 1.2 format**. Users can export the active ontology model in a local XMI file using the ODM specification, by selecting **Export->Export the ontology in ODM format** from the Outline View menu and specifying the filename and the path of the XMI file in the local file system as shown in the figure below.

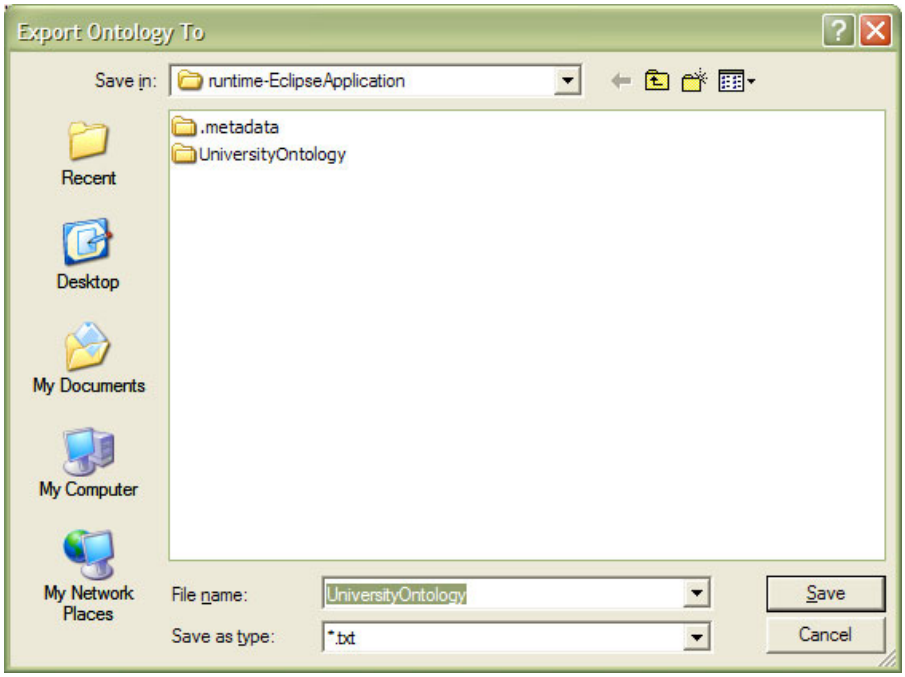

Figure 29: Export the ontology in ODM-based XMI format

- **Export the ontology in OWL-DL format**[1]. Users can also export the active ontology model into a local file using the OWL-DL specification, by selecting **Export->Export the ontology in OWL-DL format** from the Outline View menu and specifying the filename and the path of the XMI file in the local file system as shown in the figure below.

[1] The reverse functionality for this operation (i.e. importing an ontology that exists as a file in an OWL-DL language format) is supported from the Ontology Viewer that is a stand-alone plugin for browsing ontologies and therefore the respective user interface screenshots are presented in deliverable D15.5 "BML Editor Final Release". The mechanisms that have been developed in order to support this functionality are presented in chapter 5 of this report.

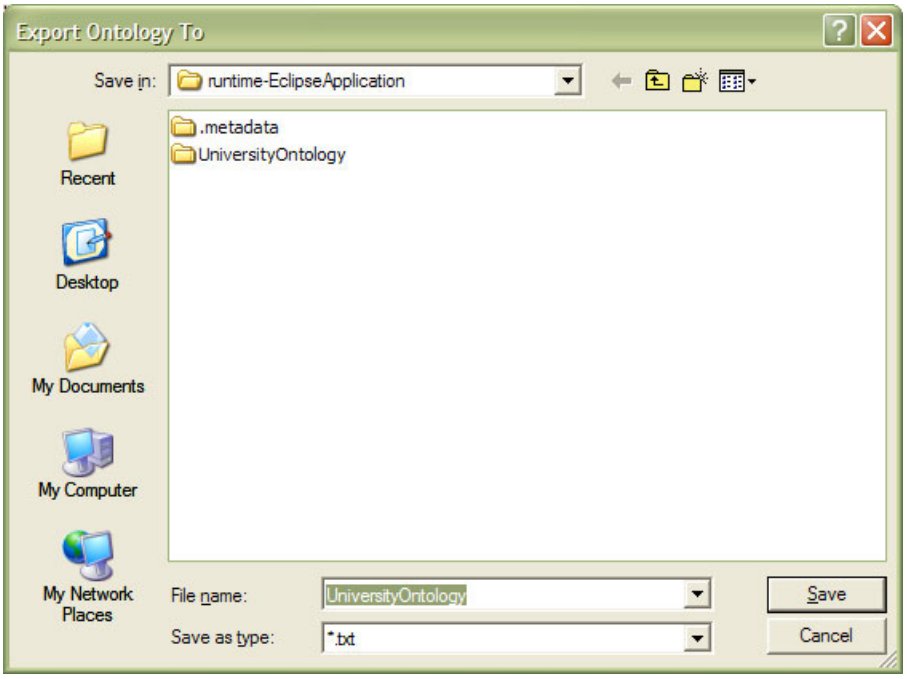

Figure 30: Export the ontology in OWL-DL format

- **Delete ontology from the local file system.** Users can delete the ontology models that have been saved locally by selecting **Delete-> Delete Ontology from the local file system** and specifying the ontologies from a list of all the available ontologies as shown in the figure below.

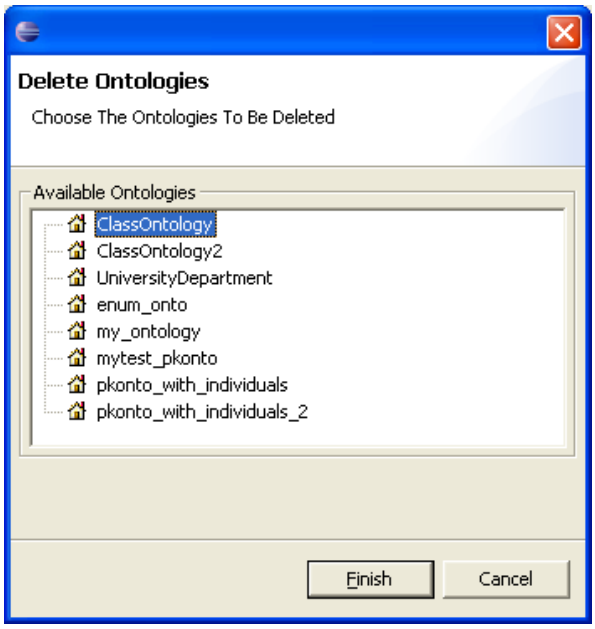

Figure 31: Delete ontology from the local file system

- **Delete ontology from the DBE Knowledge base.** Users can delete the ontology models that have been saved in the DBE Knowledge Base by selecting **Delete-> Delete**

Ontology from the DBE Knowledge Base and specifying the ontologies from a list of all the available ontologies of the Knowledge Base as shown in the figure below.

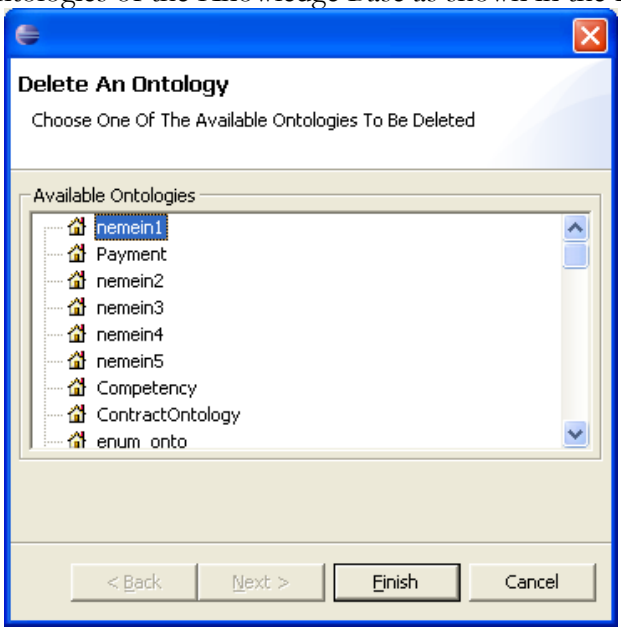

Figure 32: Delete ontology from the DBE Knowledge base

The Model Tree View Explorer

The model tree view explorer is located in the Outline View and represents the active ontology model in a tree hierarchy manner. Users can navigate inside the tree hierarchy and select any of the ODM elements of the ontology. By clicking in an ontology element, the properties values of the element are displayed in the Property Sheet of the Ontology Analysis Tool, and the graphical representation of the ODM element is highlighted in the ontology diagram that includes it.

In order to add an existing ODM element in a diagram, users can select the element from the Model TreeView Explorer and "drag-and-drop" it into the diagrams editor. In this way an ODM element can participate in more than one ontology diagrams. For example, for an ODM class, users can define its class hierarchy (sub-classes and super-classes) in one diagram and its ObjectProperties in another diagram.

Note: The drag-and-drop utility does not enable users to add an existing ODM element into an ontology diagram twice.

4.3. The Diagrams Editor (GEF Editor)

The diagrams editor is the main working area that is located in the centre of the Ontology Analysis Tool UI, is responsible for manipulating the diagrams of the active ontology. It provides a graphical environment and a tools palette that contains a list with all the ODM model elements. Using the tools palette users can deploy the ontology models that meet the business requirements of the SMEs in the context of the DBE project.

For more convenience, an ODM Ontology model can be represented in more than one diagrams as shown in the figure below.

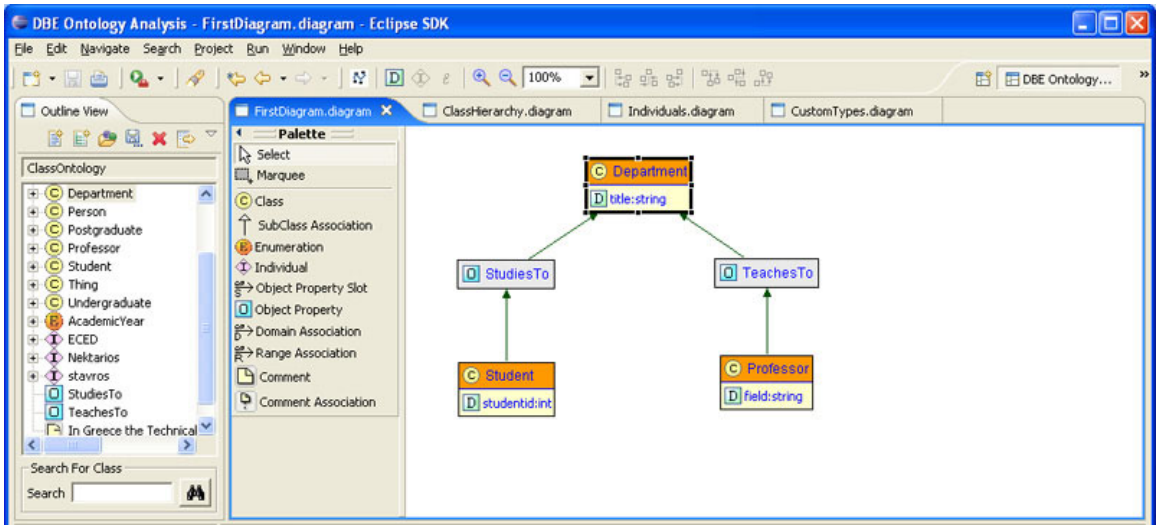

Figure 33: The Diagrams Editor

The tools palette supports the following operations:

- **Select an ODM element**: Using the **Select** tool users can select an ODM element by clicking in the graphical representation of it inside the diagrams editor. Alternatively, using the **Marquee** tool users can drag a rectangle and select all the ODM elements inside the rectangle simultaneously.
- **Create an ODM class and a hierarchy of ODM classes**: Users can create a new ODM class in an ontology diagram using the **Class** tool and specify sub-class associations between them using the **SubClass Association** tool as shown in the figure below.

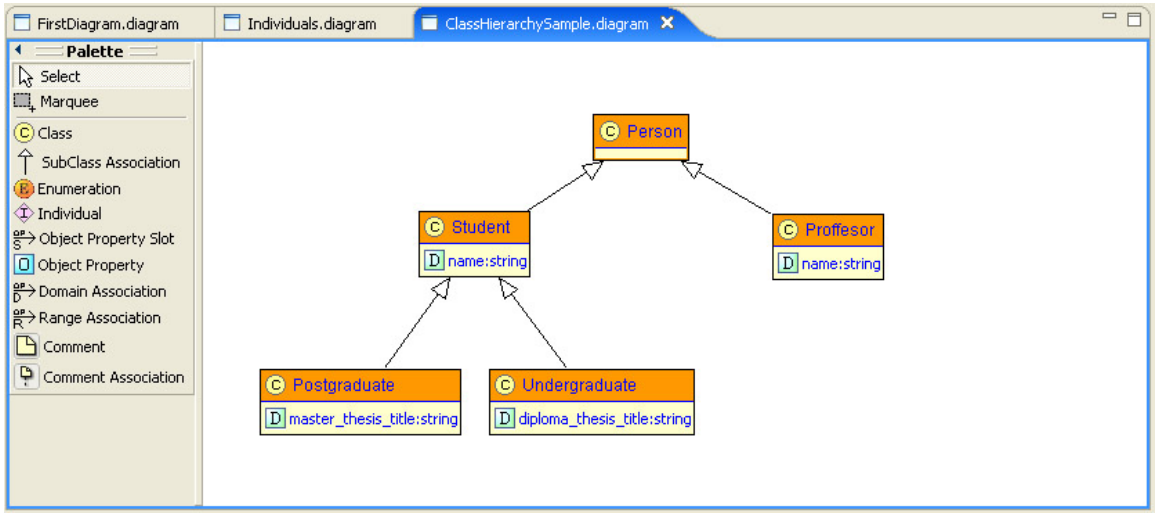

Figure 34: An ODM class hierarchy example

- **Create an ODM ObjectProperty and associations between ODM classes and ODM ObjectProperty**: Users can create an Object Property in the ontology diagram using the **Object Property** tool. Also, they can specify the Domain and Range relationships between ODM Class elements and ObjectProperties elements using the **Domain Association** and the **Range Association** tools as shown in the figure below.

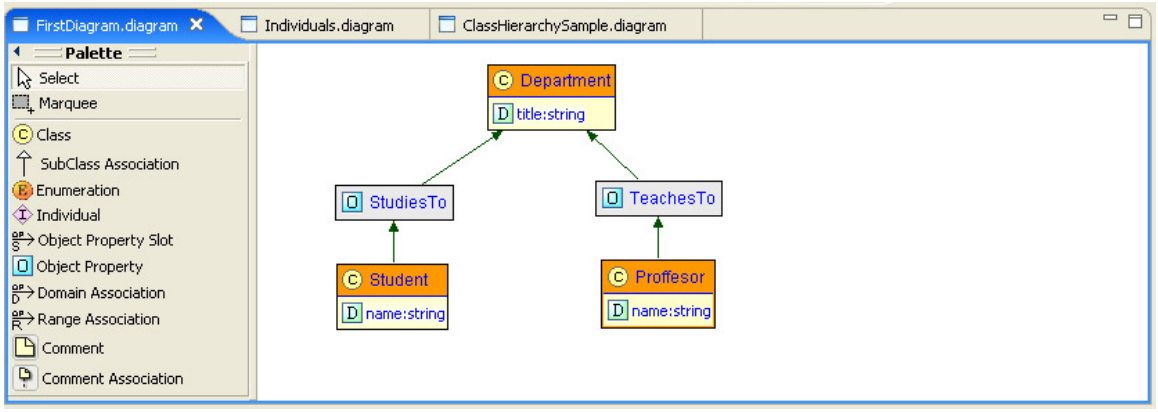

Figure 35: An ODM ObjectProperty Example

- **Create an ODM Enumeration**: Using the **Enumeration tool** the users define an ODM Enumeration element in the ontology diagram as shown in the figure below.

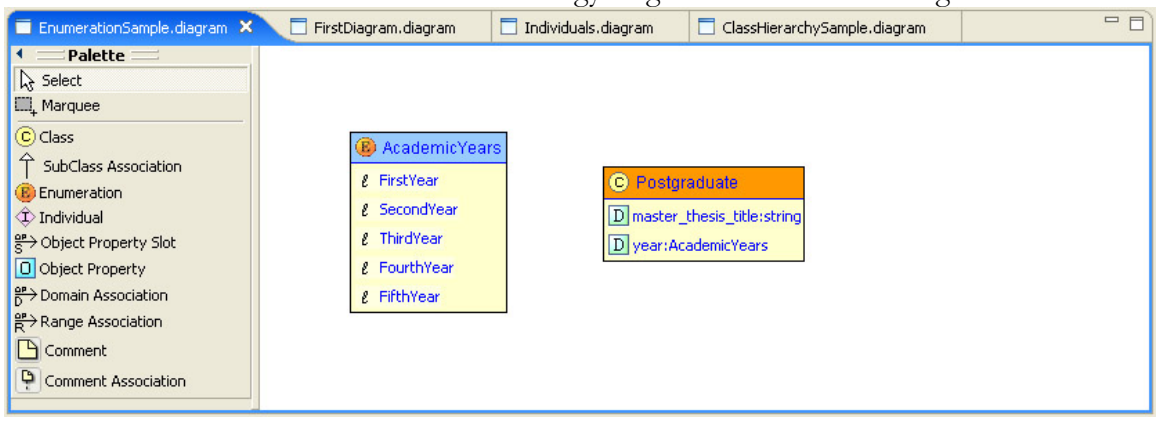

Figure 36: An ODM Enumeration Example

- **Create ODM Individuals and Property Slots between them:** Users can insert an ODM Individual element in the active ontology using the **Individual tool** and specifying the name and the individual type as shown in the figure below.

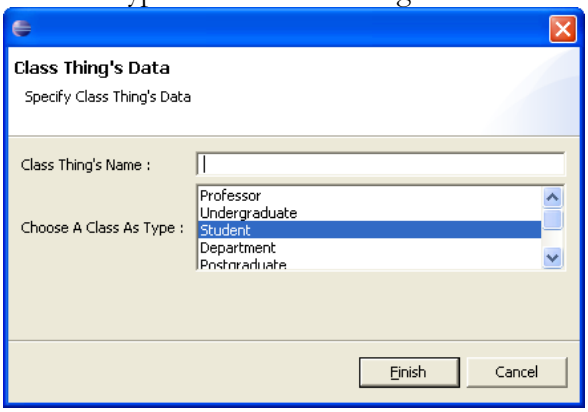

Figure 37: Add an individual

Also, using the **Object Property Slot**, users create instances of an ODM Object Property (ObjectPropertyThing) in order to specify the relationships between the ODM Individuals.

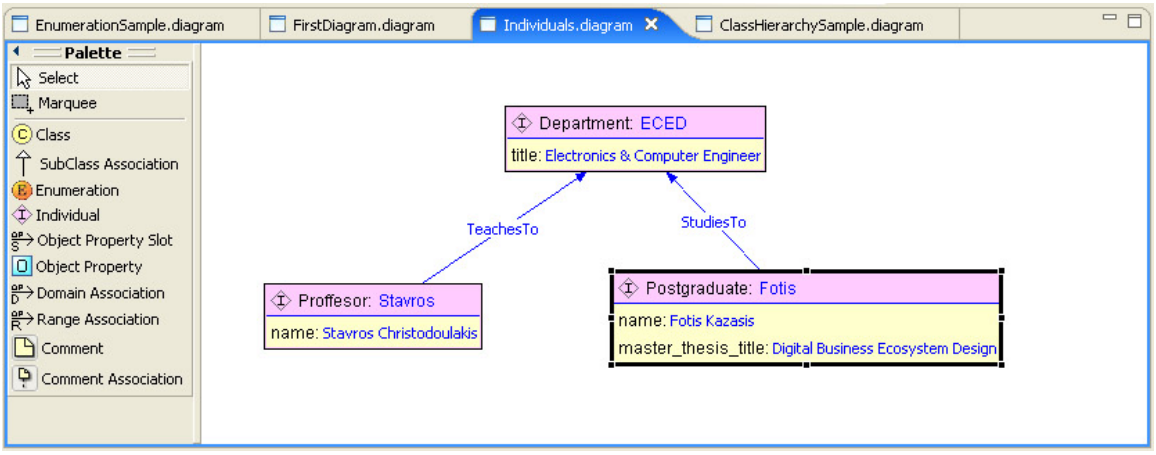

Figure 38: An ODM Individuals Example

- **Create ODM Comments and Associate them with an ODM element**: Users can insert an ODM Comment element using the **Comment tool**, and associate it with another ODM element using the **Comment Association tool** as shown in the figure below.

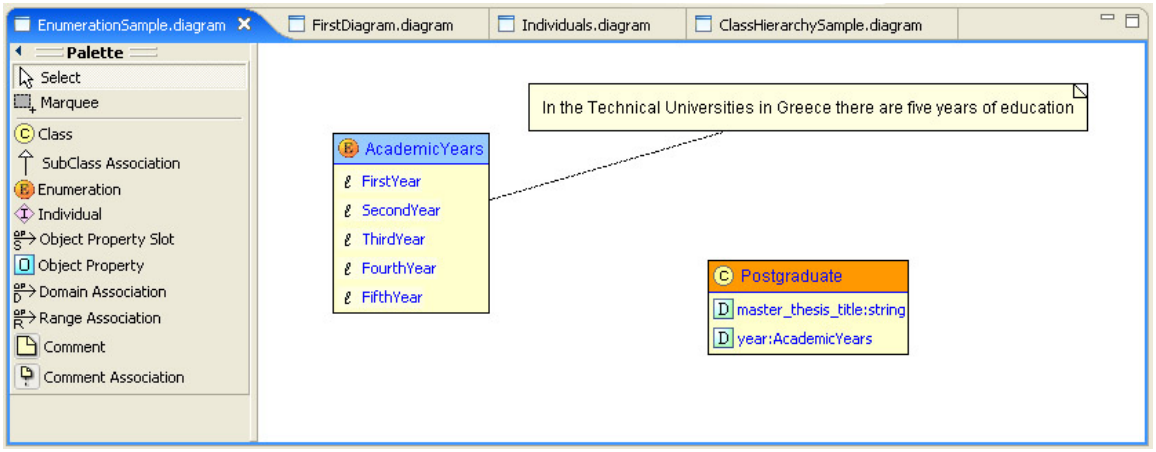

Figure 39: An ODM Comment Example

The Ontology Analysis Tool provides a **context menu** that is displayed when users click with the right button of the mouse on a graphical element of a diagram. The menu items of the context menu are customized and fully dependent on the graphical object where users click on. In the figure shown below, the context menu of the ODM Class element is presented.

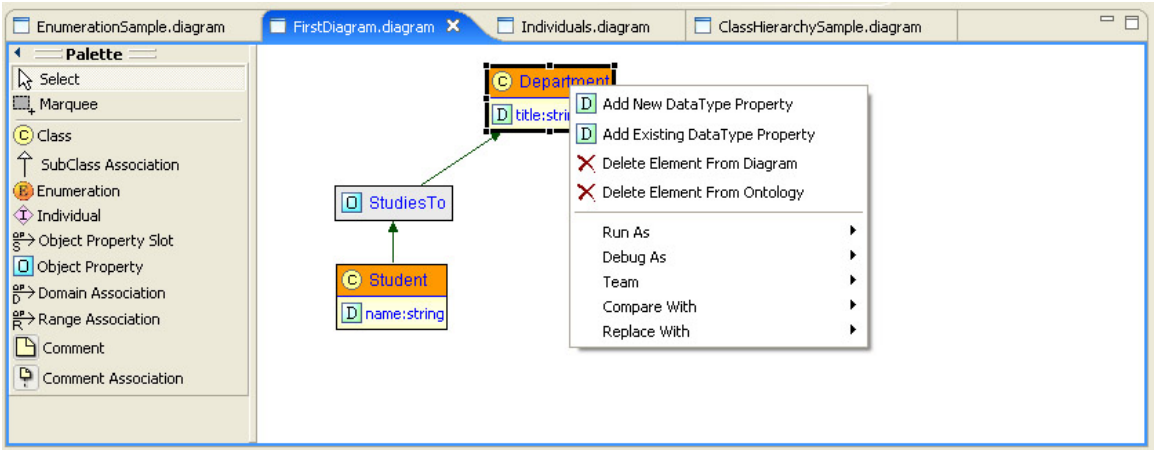

Figure 40: A Content Menu Example

The context menu provides the following functionality:

- Add a new ODM DataTypeProperty in an ODM class
- Add an existing ODM DataTypeProperty in an ODM class
- Add a new ODM DataType Property Slot in an ODM Individual
- Add an ODM Literal in an ODM Enumeration
- Delete an ODM element (Class, DataTypeProperty, ObjectProperty, Enumeration, Literal) from the diagram
- Delete an ODM element (Class, DataTypeProperty, ObjectProperty, Enumeration, Literal) from the Ontology
- Delete an association between an ODM class and an ODM ObjectProperty from diagram
- Delete an association between an ODM class and an ODM ObjectProperty from the Ontology
- Delete an ODM DataType Property Slot from the Ontology, etc.

Direct Edit and Validation

The **direct edit** utility, as provided by the GEF diagrams editors, enables users to edit the properties of the ODM elements not only from the Property Sheet but also from the corresponding graphical objects that represent the ODM elements in the diagrams editor.

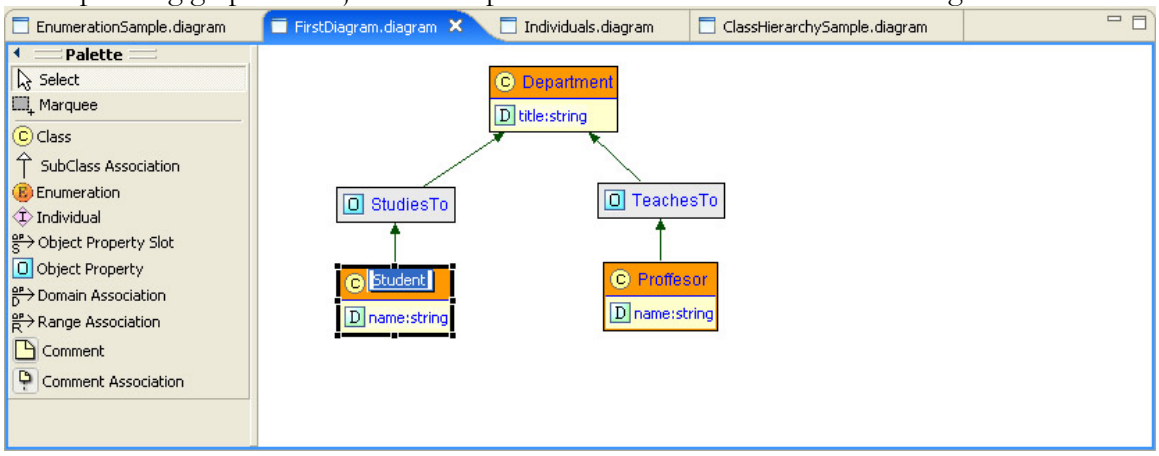

Figure 41: The Direct Edit Utility

For example, with a left button click on the name of the graphical object that represents the ODM class "Student", users can edit and change the name of the class.
In the same way, the direct edit utility can be applied to the following ODM elements: Class, Enumeration, DataTypeProperty, ObjectProperty, Individuals, Comment, and Plain Literal.
During applying the direct edit utility, a validation mechanism is active, in order to check and validate the values inserted by the users. For example, the validation mechanism prohibits the null values, the space " " character, the existence of two different ODM elements with the same name etc. and displays appropriate warning messages in the Status Bar.

4.4. The Ontology Analysis Tool Toolbar

The Ontology Analysis Tool adds a new toolbar in the Eclipse user interface environment that provides some functionality related to the graphical diagrams editor as shown in the figure below:

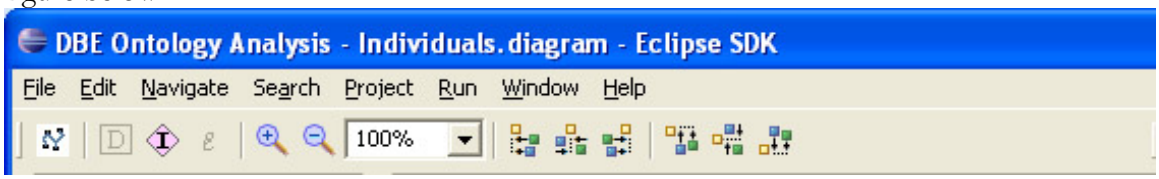

Figure 42: The graphical editor's toolbar

- **Automatic Layout**: The users of the graphical editor have the ability to place the ontology element anywhere in the diagram. With this operation, the graphical objects of the active diagram can be placed automatically with the most optimal way. Users do not have the possibility of intervention in the provision (arrangement) the graphic elements of diagram. After applying the automatic layout, users cannot manually arrange the graphical elements until they disable the automatic layout.
- **Add a new DataType Property:** Users can add a new DataTypeProperty element in a selected ODM Class element that belongs to the active diagram.
- **Add a new DataType Property Slot:** Users can add a new DataTypeProperty slot in a selected ODM Individual element that belongs to the active diagram.
- **Add a Literal:** Users can add a new Plain Literal property in a selected ODM Enumeration element that belongs to the active diagram.
- **Zoom In - Out:** The Ontology Analysis Tool enables users to determine the focus and the scale of the ontology diagrams in the graphical editor by zoom-in and out and the diminution of the diagrams. Users can also determine a custom scale value between the 25% and 2000% of the width or the length of the diagram.
- **Graphical Elements Alignment:** A set of six buttons enables users to proceed to horizontal and vertical alignment on a group of selected graphical objects of a diagram.

4.5. The Properties Sheet

The Properties Sheet is located in the bottom side of the Ontology Analysis Tool user interface and provides a customized environment that presents the properties and values of the ODM elements. It is very important for those properties that cannot be represented via a diagram. Users can select any ODM element in the Model Tree View or in a diagram and view and edit its properties and their values. There is also a synchronization mechanism between the Properties Sheet, the Diagrams Editor and the Model Tree View Explorer. So, any changes in the properties sheet update the contents of the model tree view explorer and

the diagrams and vice versa. In the rest of the section some screenshots with special ODM element properties will be presented.

Ontology Properties

There is not any special graphical element that presents a whole ontology in the Ontology Analysis Tool user interface. The users can specify some special properties about the active ontology by clicking with the left mouse button in the empty area of a diagram. In this way, the properties of the active ontology are presented in the Properties Sheet as shown in the figure below.

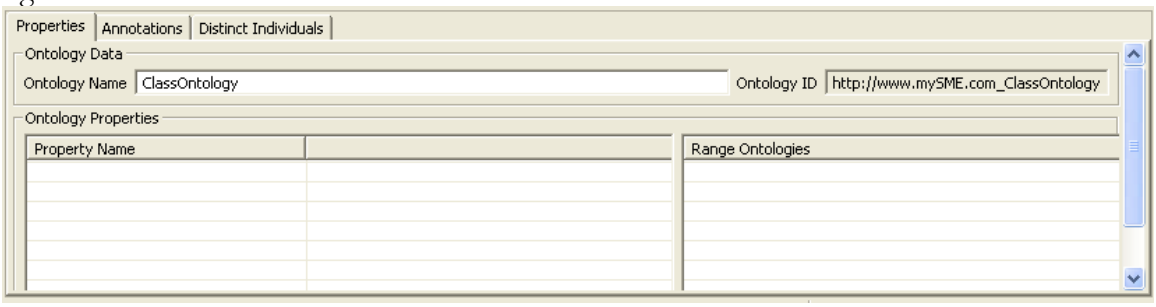

Figure 43: The Ontology Properties

Using the Properties Sheet users can view some basic properties of the active ontology such as Ontology Name and ID, add new properties, and specify range ontologies. From the Annotations tab, users can add annotations and comments about the active ontology by clicking in the corresponding areas with the right mouse button. Finally, using the Distinct Individuals tab users can specify lists of Individuals of the same Class element that can be related with the AllDifferent property.

Class Properties

The ODM Class properties are presented in the Properties Sheet if the users select an ODM class from the model tree view explorer or a diagram as shown in the three figures below.

In these figures, the properties of an ODM class are presented: Class Name, Class ID, the DataType and ObjectProperties, Inherited DataType and ObjectProperties , SubClasses or SuperClasses in the Ontology Class hierarchy, Enumerated, UnionOf, IntersectionOf or ComplementOf classes.

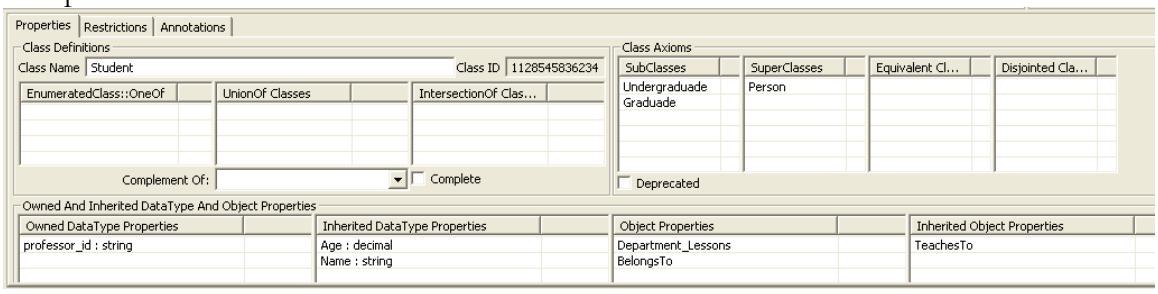

Figure 44: The ODM Class Properties

According to the ODM, Property Restrictions can be applied in a DataType or ObjectProperty of an ODM class.

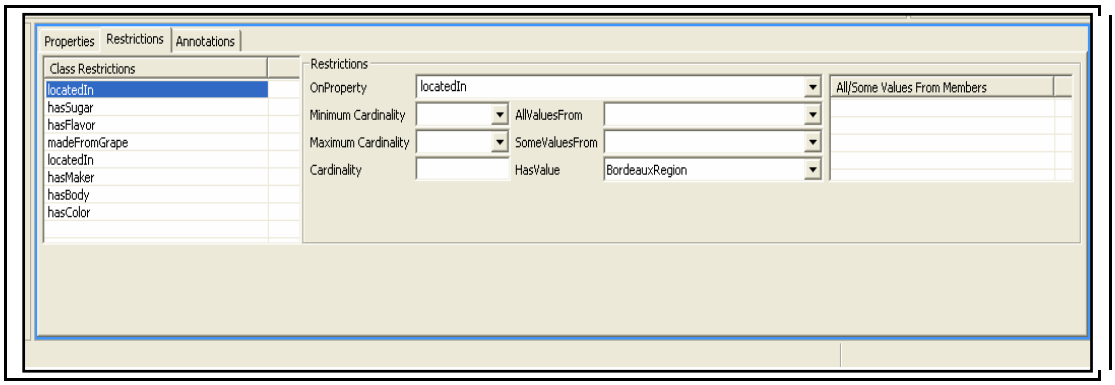

Figure 45: ODM Property Restrictions

Finally, many Annotation Properties can be related with an ODM class such as Label, VersionInfo, Comment, isDefinedBy etc.

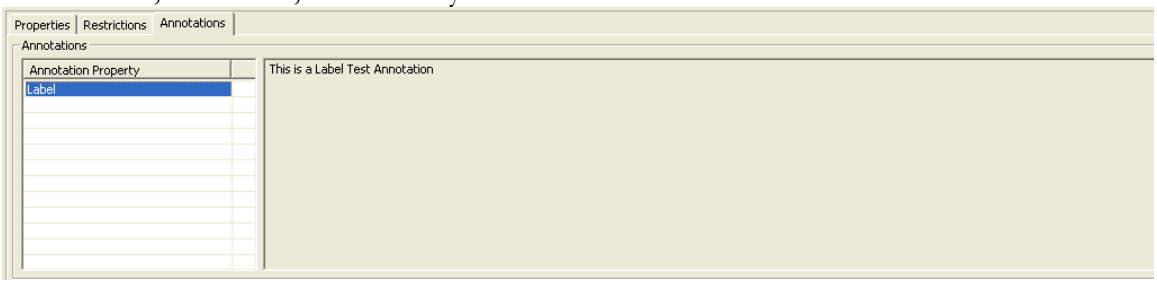

Figure 46: Class Annotation Properties

Enumeration Properties

The ODM Enumeration properties are presented in the Properties Sheet if the users select an ODM Enumeration element from the model tree view explorer or a diagram as shown in the figure below: the Enumeration id, the name and the literals of the Enumeration.

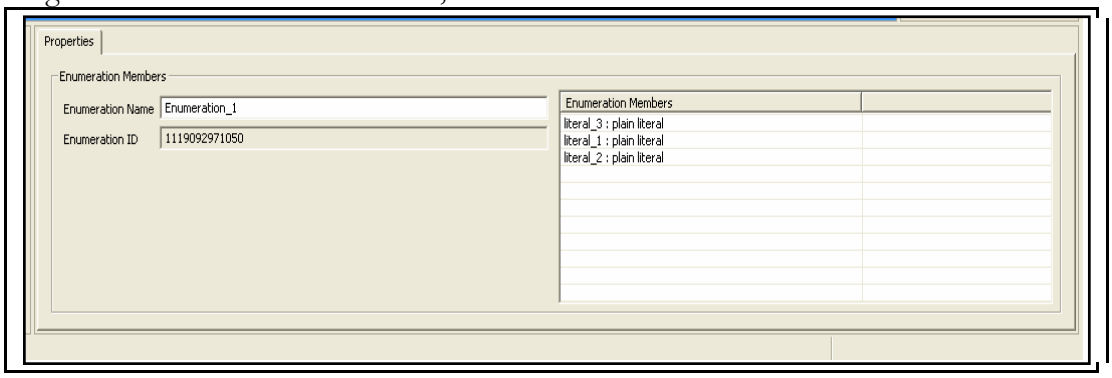

Figure 47: ODM Enumeration Properties

Users can select an Enumeration and Add Literals using the Context Menu that is displayed when they click on an Enumeration graphical object with the right mouse key. The Literal Properties are shown in the following figure.

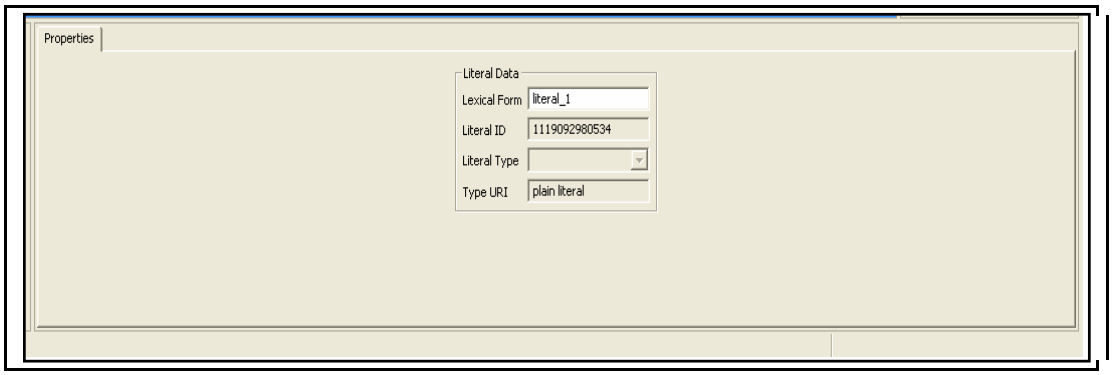

Figure 48: ODM Literal Properties

ObjectProperty Properties

The ODM ObjectProperty creates relationships between ODM classes or Enumerations in ontology. The ObjectProperty properties are presented in the Properties Sheet if the users select an ODM ObjectProperty from the model tree view explorer or a diagram. In the following figure the properties of an ODM ObjectProperty are presented: Property Name, Id, Domain and Range Classes, Super Properties in an ObjectProperty hierarchy, Equivalent Properties, InverseOf, Property Axioms etc.

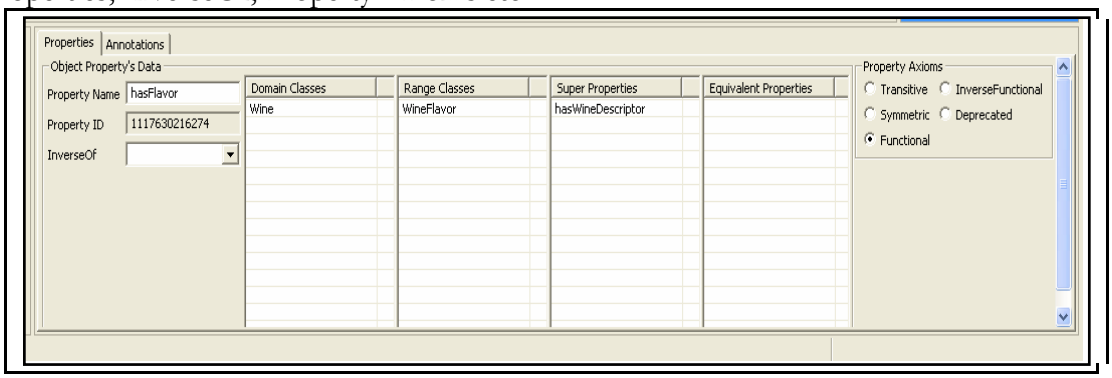

Figure 49: ODM ObjectProperty Properties

DataTypeProperty Properties

Users can select an ODM class and add a DataTypeProperty using the Context Menu that is displayed when they click on an ODM class graphical object with the right mouse key. Then, the DataTypeProperty Properties will be displayed in the Properties Sheet as shown in the following figure: Property Name, Id, Data Range, Domain Classes, Super Properties, Equivalent Properties, InverseOf, Property Axioms, Annotations etc.

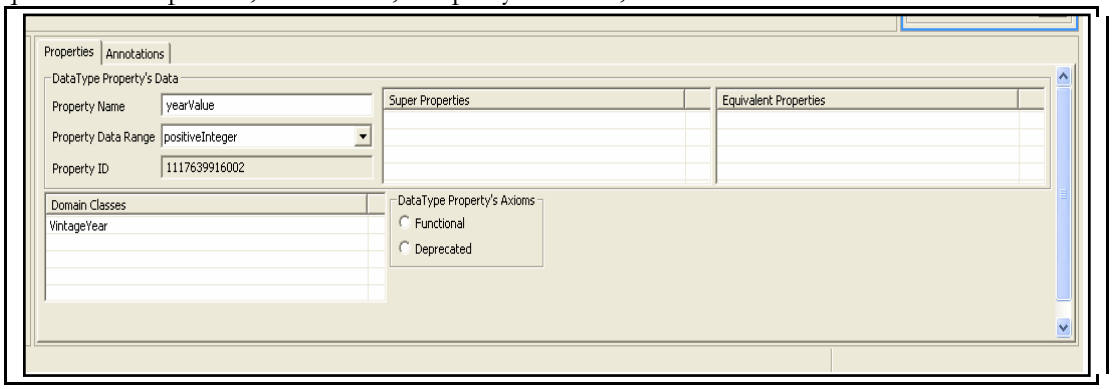

Figure 50: ODM DataTypeProperty Properties

Individual Properties

The ODM Individual properties are presented in the Properties Sheet when the users select an ODM Individual from the model tree view explorer or the graphical editor.

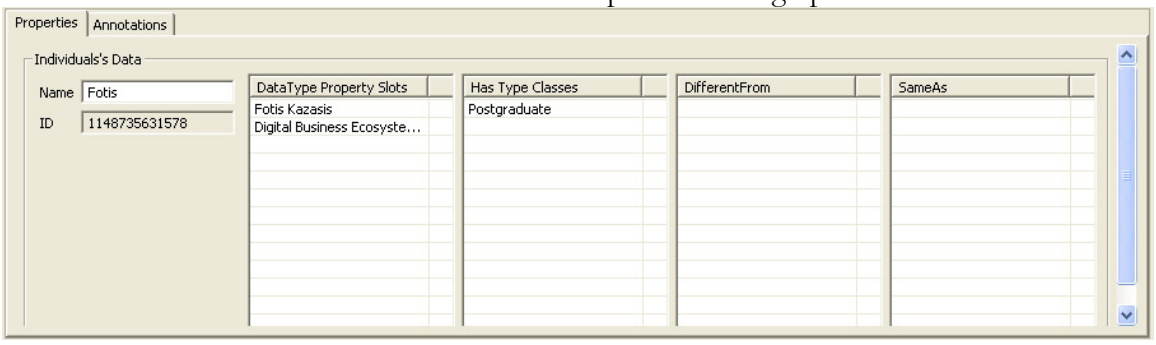

Figure 51: ODM Individual Properties

The figure above represents the properties of an ODM Individual: the Individual Name and ID, the DataType Property Slots, the HasType Classes that the selected Individual element related to, the list of other Individuals that are DifferentFrom or SameAs with the selected Individual element.

Comment Properties

The ODM Comment Properties are displayed in the Properties Sheet when the users select a Comment graphical object from the graphical editor as shown in the following figure.

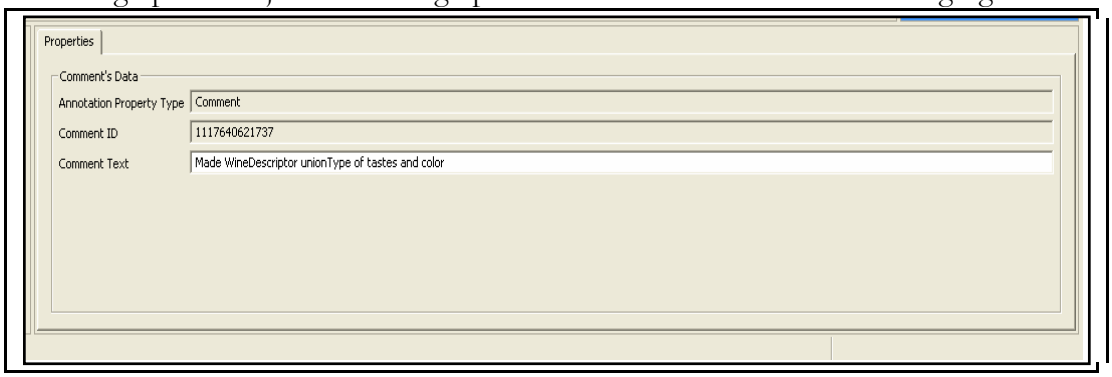

Figure 52: ODM Comment Properties

4.6. The Search Utility

The Search Utility enables users to search about terms and elements that have been defined in the active ontology and it is located in the bottom side of the Outline View as shown in the following figure.

The search utility includes terms from the following ODM elements: Classes, Enumerations, ObjectProperties, Individuals and Comments. The results of the searching process are filtered and displayed into the Model Explorer tree view.

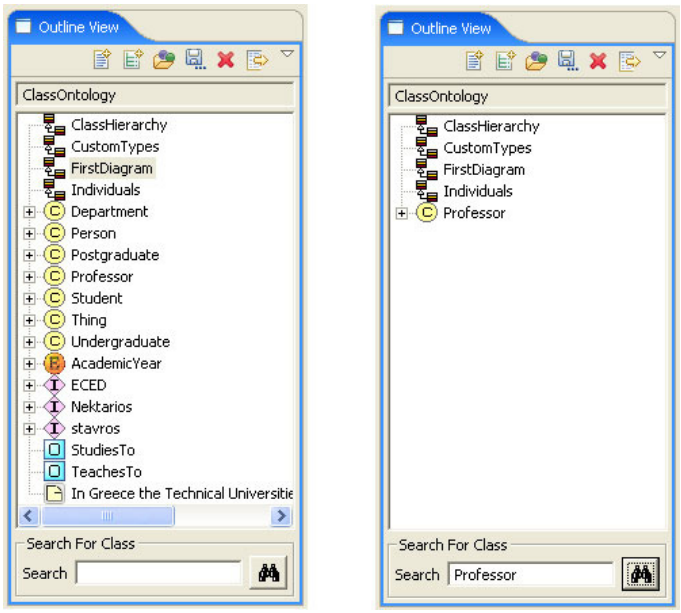

Figure 53: The Search Utility

4.7. The Outline Diagrams View

It is a region in the left-bottom corner of the user interface where the diagrams of the active ontology can be represented in scale.

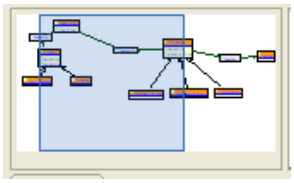

Figure 54: The Outline Diagrams View

In this way, users can easily navigate into large-side diagrams by scrolling up and down using a "thumbnail".

5. ODM and OWL mapping mechanisms

In this chapter we describe the mapping from ODM to OWL concepts and vice versa. These mappings have been applied in order to allow the interoperability of the DBE with the Semantic Web as far as the ontology framework is concerned. In particular, appropriate mechanisms have been developed in order to support the re-usage of ontologies defined in the OWL-DL language within the DBE semantic framework as well as the exporting of the domain-specific ontologies created within the DBE in a well accepted format like OWL. In order to handle in an efficient manner the desired interoperability, ODM has been designed as a MOF model (i.e. a metamodel) that is compatible with the OWL Language.

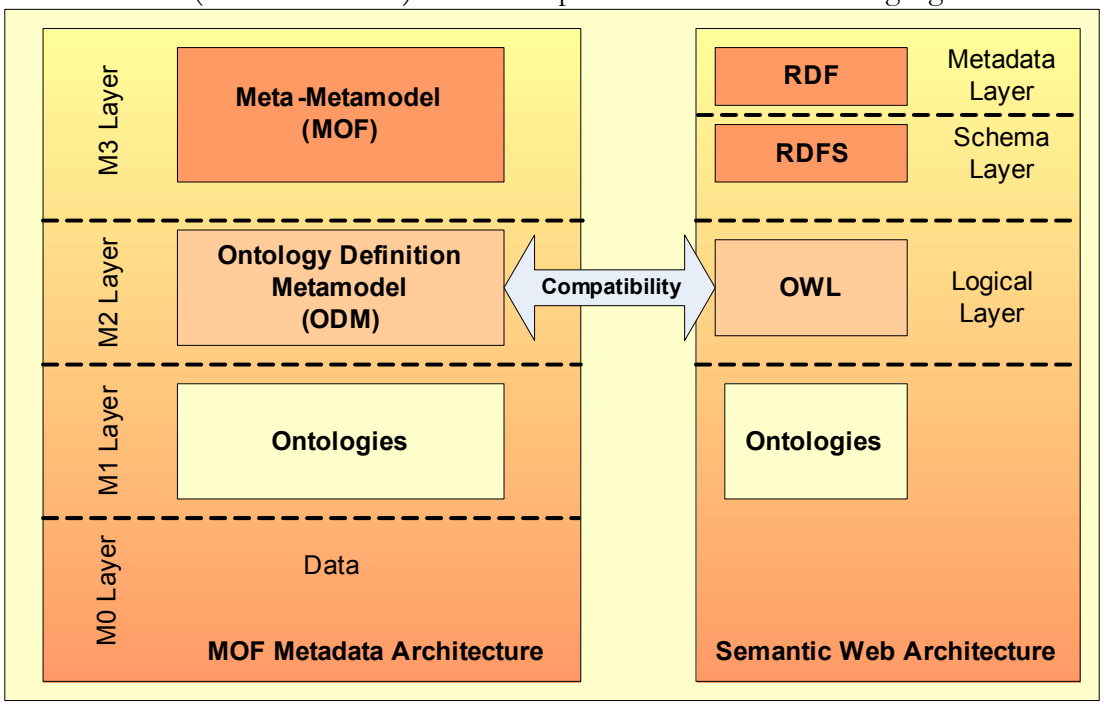

Figure 55: Metamodeling approach to ODM

Figure 55 shows the ODM in the context of the MOF metadata architecture and where it stands with respect to the Semantic Web's metamodeling architecture. The Ontology Definition Metamodel is presented as an M2 layer Model in the MOF metadata architecture and it has been appropriately defined in order to be compatible with OWL. Instances of this metamodel (M1 Layer Models) are the ontologies that the DBE experts and/or users define. Importing and exporting ontologies from ODM to OWL and vice versa is therefore supported in the M1 Layer utilizing the elements of the M2 Layer Model.

ODM is not a one-to-one representation of the OWL data model in MOF. The effort was to be as close as possible; however, due to the formalization required in MOF, and with respect to the OWL/RDF model some extra constructs were required for ensuring MOF compliance. These constructs are indicated in the ODM to OWL-DL mapping (see Table 3) as not available (N/A). On the other hand, OWL has pre-defined instances of some of its classes (e.g. the predefined ontology properties), something that is not allowed in MOF since it is not possible to define at M2 level both classes and their instances. The respective cases are indicated in the OWL-DL to ODM mapping (see Table 4) again as N/A.

The following tables show the mappings a) from ODM to OWL and b) from OWL to ODM:

ODM Class or Association	OWL Concept
Ontology	owl:Ontology
OntologyProperty	owl: OntologyProperty
AnnotationProperty	owl: AnnotationProperty
AnnotationObject	N/A
Class	owl:Class
complementOf_As	owl:complementOf
intersectionOf_As	owl:intersectionOf
unionOf_As	owl:unionOf
oneOf_As	owl:oneOf
disjointWith_As	owl:disjointWith
equivalentClass_As	owl:equivalentClass
subClassOf_As	rdfs:subClassOf
DeprecatedClass	owl:DeprecatedClass
Restriction	owl:Restriction
allValuesFrom_As	owl:allValuesFrom
someValuesFrom_As	owl:someValuesFrom
hasValue_As	owl:hasValue
onProperty_As	owl:onProperty
maxCardinality_As	owl:maxCardinality
minCardinality_As	owl:minCardinality
cardinality_As	owl:cardinality
Value	N/A
ValueRange	N/A
Thing	owl:Thing
AllDifferent	owl:allDifferent
distinctMembers_As	owl:distinctMembers
sameAs_As	owl:same_As
hasType_As	owl:type
differentFrom_As	owl:differentFrom
Property	N/A
DatatypeProperty	owl:DatatypeProperty
ObjectProperty	owl:ObjectProperty
hasDomain_As	rdfs:domain
hasDataRange_As	rdfs:range
hasRange_As	rdfs:range
inverseOf_As	owl:inverseOf
subPropertyOf_As	owl:subPropertyOf
equivalentProperty	owl:equivalentProperty
SymmetricProperty	owl: SymmetricProperty

TransitiveProperty	owl: TransitiveProperty
InverseFunctionalProperty	owl: InverseFunctionalProperty
FunctionalObjectProperty	owl:FunctionalProperty
FunctionalDatatypeProperty	owl:FunctionalProperty
DeprecatedDatatypeProperty	owl:DeprecatedProperty
DeprecatedObjectProperty	owl:DeprecatedProperty
PrimitiveType	N/A
DeprecatedDatatype	owl:DeprecatedClass
Datatype	rdfs:Datatype
DataRange	rdfs:DataRange
Enumeration	rdf:List
Literal	owl:Literal
PlainLiteral	N/A
TypedLiteral	N/A
URIreference	N/A
DefinitionURI_As	N/A
Datatype_As	N/A
NonNegativeInteger	N/A

Table 3: ODM to OWL-DL mapping

OWL	**ODM Class or Association**
owl:allDifferent	allDifferent
owl:allValuesFrom	allValuesFrom_As
owl:AnnotationProperty	AnnotationProperty
owl:*backwardCompatibleWith*	N/A
owl:cardinality	cardinality_As
owl:Class	Class
owl:complementOf	complementOf_As
owl:DatatypeProperty	DatatypeProperty
owl:DeprecatedClass	DeprecatedClass
owl:DeprecatedProperty	DeprecatedDatatypeProperty DeprecatedObjectProperty
owl:DataRange	DataRange
owl:differentFrom	differentFrom_As
owl:disjointWith	disjointWith_As
owl:distinctMembers	distinctMembers_As
owl:equivalentClass	equivalentClass_As
owl:equivalentProperty	equivalentProperty
owl:FunctionalProperty	FunctionalObjectProperty FunctionalDatatypeProperty
owl:hasValue	hasValues_As
owl:*imports*	N/A

owl:*incompatibleWith*	N/A
owl:intersectionOf	intersectionOf_As
owl:inverseFunctionalProperty	inverseFunctionalProperty
owl:inverseOf	inverseOf_As
owl:maxCardinality	maxCardinality_As
owl:minCardinality	minCardinality_As
owl:Nothing	N/A
owl:ObjectProperty	ObjectProperty
owl:oneOf	oneOf_As
owl:onProperty	onProperty_As
owl:Ontology	Ontology
owl:OntologyProperty	OntologyProperty
owl:*priorVersion*	N/A
owl:Restriction	Restriction
owl:sameAs	sameAs_As
owl:someValuesFrom	someValuesFrom_As
owl:SymmetricProperty	SymmetricProperty
owl:Thing	Thing
owl:TransitiveProperty	TransitiveProperty
owl:unionOf	unionOf_As
owl:*versionInfo*	N/A
rdf:type	HasType_As
rdfs:Datatype	Datatype
rdfs:domain	hasDomain_As
rdfs:Literal	Literal
rdfs:range	hasDataRange_As hasRange_As
rdfs:subClassOf	subClassOf_As
rdfs:subPropertyOf	subPropertyOf_As

Table 4: OWL-DL to ODM Mapping

Next we will describe more details regarding the import module; the mechanism that has been developed in order to support the importing of OWL-DL ontologies into the DBE environment and their representation and manipulation as ODM ontologies. A similar approach has also been followed with respect to the export module, i.e. the mechanism for exporting ODM ontologies as ontologies expressed in the OWL-DL language.
The next figure depicts the architecture of the importer module.

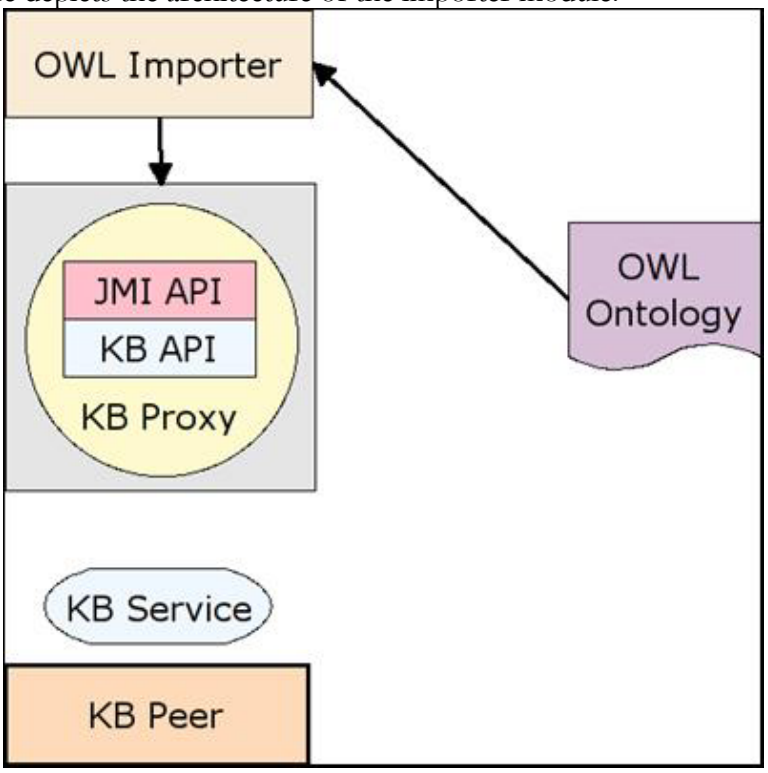

Figure 56: The architecture of the importer module

The OWL Importer reads a file containing a valid OWL-DL Ontology. The Importer connects to the KB Proxy using the JMI API and through it to the KB Service. In general, the tool stands between the OWL ontologies and the Knowledge Base stored ODM ontologies. The OWL Importer source code is divided into seven (7) packages (see also Figure 57):

Connectors: The java classes that handle the connection with the KB Proxy.

Factories: The classes that handle the creation of new class, property and restriction objects.

Modifiers: The classes that handle the completion of the objects (classes, properties, restrictions) and take care of all the possibilities (super/sub classing, enumerations etc.)

Parsers: The classes that parse the OWL ontology file

Util: Utility classes used for various jobs within the Importer module.
Writers: On-screen and file writers for the XMI file.

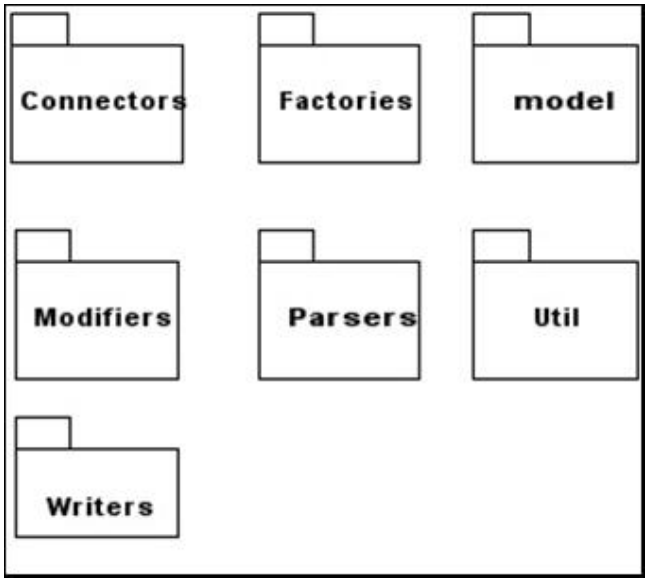

Figure 57: The organization of the Importer module into packages

The next two figures present in the form of UML activity diagrams some of the mechanisms that have been adopted in order to support the OWL-DL to ODM mapping.

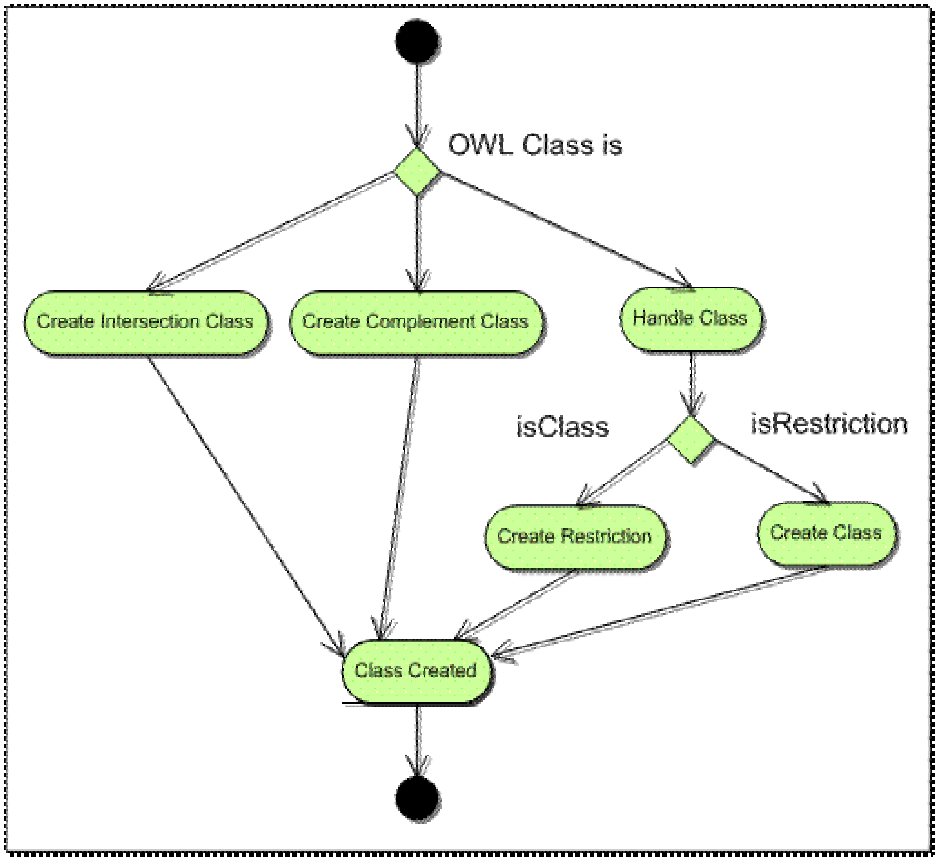

Figure 58: Parsing an OWL Class

When we parse an OWL class, we determine whether it is an Intersection, Complement or other type of class. If it doesn't fit in the first two categories, then it must be a "simple"

class, or a Restriction. In all cases, the appropriate objects are created and the process is finished.

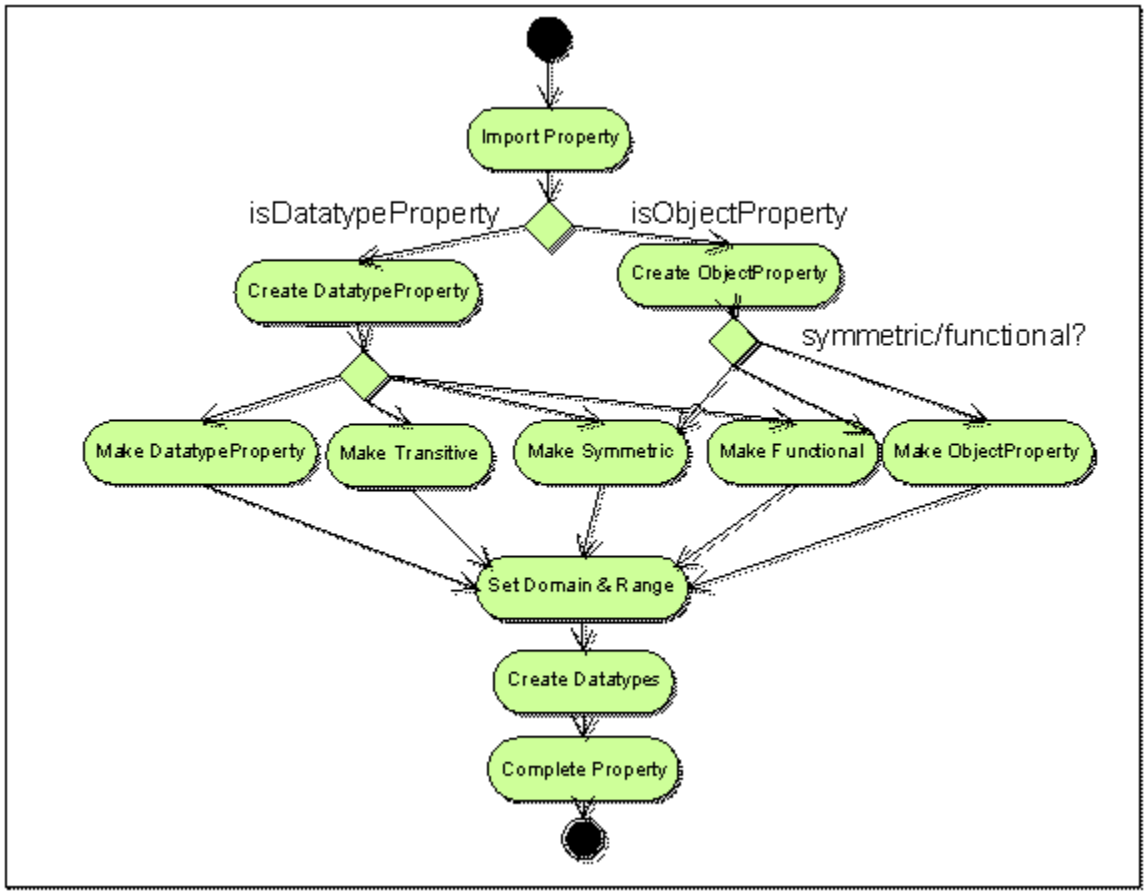

Figure 59: Parsing an OWL Property

When we parse a Property, we first check whether it is a Datatype Property or an Object Property. Based on the property kind we then create the appropriate objects:

- Datatype Property: We check to see whether the Datatype Property is Transitive, Symmetric, Functional or just "simple" Datatype Property and modify the object accordingly to match that.
- Object Property: We check whether the property is Symmetric, Functional or "simple" Object Property and then modify the object accordingly.

For both cases, we set the correct domain; ranges and datatypes (for datatype properties) and then we have the final Property.

6. Installation Instructions

In order to use the Ontology Analysis Tool, users have to install the Eclipse 3.1 platform as well as GEF 3.1. Ontology Analysis Tool is an eclipse plug-in that is dependent on some additional plug-ins. All the required feature set can be downloaded from SourceForge (https://sourceforge.net/projects/dbestudio). For detailed description of the installation process you can visit http://dbestudio.sourceforge.net/.
The Ontology Analysis Tool can also be installed independently of the DBE Studio. This can be done downloading from TUC/MUSIC site at http://www.music.tuc.gr/DBE/soft/ontology_editor/ the following plugins:

- org.dbe.studio.core.perspectives_0.2.2.zip
- org.dbe.studio.core.preferences_0.2.1.zip
- org.dbe.studio.tools.kbtoolkit_0.2.1.zip
- org.dbe.studio.editors.odm_0.2.1.zip

After that, unzip the above files in the <ECLIPSE_HOME>/plugins folder and restart Eclipse.

7. Glossary

Term	Description
API	Application Programming Interface: Is a technology that facilitates exchanging messages or data between two or more different software applications
BML	Business Modelling Language
CIM	Computational Independent Model: The most abstract layer in the MDA architecture. Models of this layer describe a modelled system from the business point of view (i.e. requirements, abstract functions, etc.) without any assumption on the implementation (software or not) details.
DBE KRF	DBE Knowledge Representation Framework: The Framework of the interconnected DBE Knowledge Representation Models and/or Languages that capture the knowledge of the ecosystem.
KB	Knowledge Base: Is the part of the DBE system where the DBE knowledge is stored and managed. Such knowledge refers to ontologies, business and service descriptions, etc.
MDA	Model Driven Architecture: An approach (proposed by OMG) to IT system specification that separates the specification of system functionality for the specification of the implementation of that functionality on a specific technology.
MOF	Meta Object Facility: A generalized facility and for specifying abstract information about very concrete object systems.
MOF Repository	A Repository for storing, managing and retrieving meta-data (models) and meta-meta-data (metamodels) that have been described with MOF.
ODM	Ontology Definition Metamodel: A MOF model (metamodel) developed in DBE for ontology representation.
OMG	Object Management Group: International standardization body
OWL	Ontology Web Language: The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics. OWL has three increasingly expressive sublanguages: OWL Lite, OWL DL, and OWL Full.
PIM	Platform Independent Model of a modelled system
PSM	Platform Specific Model of a modelled system
RDF	Resource Description Framework. RDF is a general framework for how to describe any Internet resource such as a Web site and its content. It provides interoperability between applications that exchange machine-understandable information on the Web. RDF descriptions are often referred to as metadata, or "data

	about data". These can include the authors of the resource, date of creation or updating, the organization of the pages on a site. i.e. the sitemap, information that describes content in terms of audience or content rating, key words for search engine data collection, subject categories, etc.
SCM	Service Composition Metamodel: The DBE metamodel used to model the internal workflow and communication logic of composite services. At the moment BPEL4WS is used for this purpose
SDL	Service Description Language: A MOF model (metamodel) that provides technical description of the programmatic interface of a service
Semantic Registry (SR)	It is the component of the DBE Knowledge Base that hosts the service descriptions published in the DBE environment (in the form of Service Manifest Documents) and available for discovery and consumption.
Service Manifest (SM)	The Service Manifest is a two-fold formal description of a specific DBE Service, and contains both the models and data (comprising both business and technological information) of a single specific real word service owned by a specific SME.
SME	Small and Medium Enterprise: Independent enterprise with less than 250 dependent.
SSL	Semantic Service Language: A MOF-based language for semantically describing SME services in DBE.
UML	Unified Modeling Language: A method for specifying, visualizing, and documenting the artefacts of an object-oriented system under development; as well as for business modelling.
User Profiling Mechanism	A DBE mechanism used to trace user's actions (and transactions) in order to inspect his preferences on desirable services, and partners.
W3C	World Wide Web Consortium: International Standardization body that has defined and is maintaining many IT related standards like HTML, XML, XML-Schema, OWL, etc.
XMI	XML Metadata Interchange: An SMIF (see SMIF description) standard specification based on XML.
XML	eXtensible Mark-up Language. XML is a flexible way to create common information formats and share both the format and the data on the World Wide Web, intranets, and elsewhere.
XML –Schema	An XML schema is a description of the type of an XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntax constraints imposed by XML itself. An XML schema provides a view of the document type at a relatively high level of abstraction.

8. References[2]

[1] OWL Web Ontology Language Overview, W3C Recommendation 10 February 2004, Deborah L. McGuinness (Knowledge Systems Laboratory, Stanford University), Frank van Harmelen (Vrije Universiteit, Amsterdam).
[2] Eclipse Platform Technical Overview, Object Technology International Inc., February 2003 (updated for 2.1; originally published July 2001).
[3] GEF Overview, http://www.eclipse.org/gef/.
[4] Randy Hudson, Software developer, IBM: Create an Eclipse-based application using the Graphical Editing Framework, 29 July 2003.
[5] Michael Denny: Ontology Tools Survey, Revisited, July 14, 2004, http://www.xml.com/ pub/a/2004/07/14/onto.html
[6] Protégé Official Home Page, http://protege.stanford.edu/overview/index.html
[7] Protégé Owl Plug-in Official Home Page, http://protege.stanford.edu/plugins/owl/
[8] Protégé ezOWL Plug-in Official Home Page, http://iweb.etri.re.kr/ezowl/ #Introduction
[9] Construct Tool Official Home Page, http://www.networkinference.com/ products/construct_it.html
[10] IsaViz Tool Official Home Page, http://www.w3.org/2001/11/IsaViz/
[11] OI-modeler Tool Official Home Page, http://kaon.semanticweb.org/users
[12] MR3 Tool Official Home Page, http://panda.cs.inf.shizuoka.ac.jp /mmm/mr3/index.html
[13] OntoTrack Tool Official Home Page, http://www.informatik.uni-ulm.de /ki/ontotrack/
[14] RDFAuthor Tool Official Home Page, http://rdfweb.org/people/ damian/RDFAuthor/
[15] TUC, DBE Deliverable, D14.1 – DBE Knowledge Representation Models, May 2005
[16] INTEL, DBE Deliverable, D26.3 DBE Studio Integration, April 2006
[17] OMG XML Metadata Interchange (XMI) Specification v1.2 http://www.omg.org/cgi-bin/apps/doc?formal/02-01-01.pdf, 2002
[18] Cockburn Alistair, Writing Effective Use Cases (The Crystal Collection for s/w professionals), Addison-Wesley, 2001
[19] ISUFI, DBE Deliverable, D15.1: Business Modelling Language 1.0, May 2005
[20] SOLUTA.net, DBE Deliverable, D16.1: Service Description Models and Language Defintion, October 2005
[21] SOLUTA.net, DBE Deliverable, D21.2: Architecture Scope Document, June 2005
[22] INTEL, DBE Deliverable, D20.9: DBE Studio User Interface Evaluation, draft version, July 2006
[23] The Object Management Group, ODM RFP, http://www.omg.org/cgi-bin/apps/doc?ad/03-09-06.pdf

[2] DBE deliverables are available at the DBE official site http://www.digital-ecosystem.org