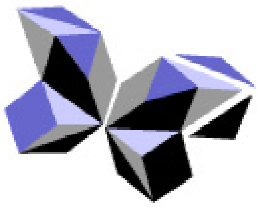
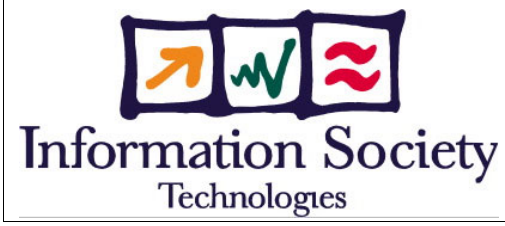


Digital Business Ecosystem

Contract n° 507953

# Workpackage 15: DBE Business Modeling Language

# Deliverable D15.5: BML Editor Final Release



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

**Contract Number**: 507953

**Project Acronym**: DBE

**Title**: Digital Business Ecosystem

**Deliverable N°**: D15.5

**Due date**: 06/2006

**Delivery Date**: 07/2006

**Short Description**:

This document accompanies the software deliverable for the Final Release of the BML Editor for the DBE project. This editor will be used by Business Analysts on behalf of the SMEs for defining business and service models according to he corresponding metamodels. The BML Editor as a visual modeling tool provides a UML-like Graphical User Interface (GUI), similar to that of well know UML editors, for supporting the modeling tasks and stores the created models in DBE **K**nowledge **B**ase (KB).

Author: Technical University of Crete (TUC)

**Partners contributed**: Technical University of Crete (TUC)

**Made available to**: Public

**Versioning**

| Version | Date | Author, Organization |
|---|---|---|
| *1.0* | *3/07/2006* | *GEORGE ANESTIS, TECHNICAL UNIVERSITY OF CRETE* |
| *2.0* | *12/07/2006* | Revisions based on reviewers' suggestions/comments/changes |
| | | |

**Quality check**

**1st Internal Reviewer**: Leon Mak, UniS

**2nd Internal Reviewer**: Dominik Dahlem, TCD

## Table of Contents:

## List of Figures

# 1. Introduction

This document accompanies the software deliverable for the final version of the BML Editor (the Business Analysis Tool) for the DBE project [1]. This editor will be used by business analysts on behalf of SMEs for defining business and service models according to the corresponding metamodels. The BML Editor as a visual modeling tool provides a UML-like Graphical User Interface (GUI), similar to that of well known UML editors, for supporting the modeling tasks and stores the created models in the DBE Knowledge Base (KB). The current version of the editor supports both, the semantic description of the services offered by an SME and the business model of the particular SME. The former provides to the user the ability to create service models according to the Semantic Service Language metamodel [2] and the latter to create business models based on the Business Model Language (BML) metamodel [3]. Both metamodels are described using OMG's MOF 1.4 [4].

During the modeling process the user can take advantage from domain specific ontologies that have been described using the Ontology Definition Metamodel (ODM) [2] and stored into the DBE Knowledge Base. These ontologies can be explored using the Ontology Viewer that provides a tree-view representation of an ontology that allows the browsing and navigation of it. In addition, the OWL Importer Module –part of the Ontology Viewer- provides the transformation of an OWL-DL [5] ontology into an ODM ontology and stores it in the DBE KB. In this way the modeller can use ontology concepts as types in the models that designs taking full advantage of the expressiveness provided by ODM and OWL-DL ontologies. Ontology Viewer will be also described in this document.

This document does not intend to describe the SSL and BML metamodels but only the usage of the BML Editor for the creation of these models. Consequently, it is highly recommended the reader of this document to read first the documents that describe both the SSL and the BML metamodels.

# 2. User Requirements Analysis

The user requirements of the BML Editor have been described in detail in [6]. In this section the functionality and the user requirements of the OWL Importer module will be described using the "use cases" templates as they have defined by Alistair Cockburn in his book titled "Writing Effective Use Cases" [7]. Use Cases is a software engineering methodology that provides a standard and comprehensive representation of the functionality of a system.

The use case methodology includes a set of distinct use case scenarios. Each of them captures a contract between the stakeholders of a system about its behaviour. The use case describes the behaviour of the system under various conditions as the system responds to a request from one of the stakeholders (called the primary actor). The primary actor initiates an interaction with the system to accomplish some goal. Different sequences of behaviour or scenarios can unfold, depending on the particular requests and the conditions surrounding the requests. The use cases gather different scenarios together.

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 process in business process engineering.
- **Summary or Strategic Level**: This Summary Level use case involves 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 it provides a table of contents for the lower level use cases.
- **Sub-Function Level**: The Goal of a Primary Task Level use case usually consists of 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 in the Use Cases Level hierarchy.

A diagram (see Figure 1) that displays all the use cases and their relationships according to the level hierarchy they belong follows. A use case is presented as a rectangle and a relationship as an arrow:

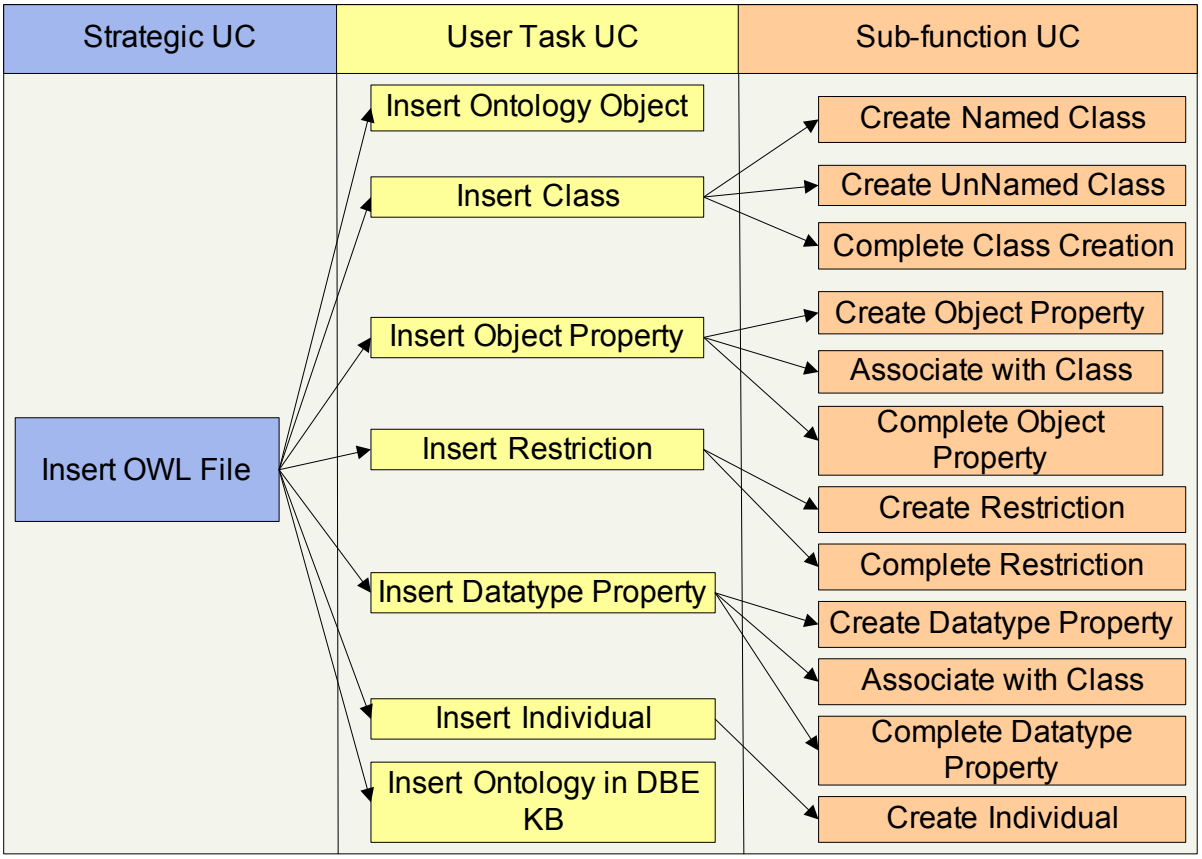



**Figure 1: Use cases for the OWL Importer module**

A diagram (see Figure 2) that displays all the use cases and their relationships using the UML use cases notation follows:

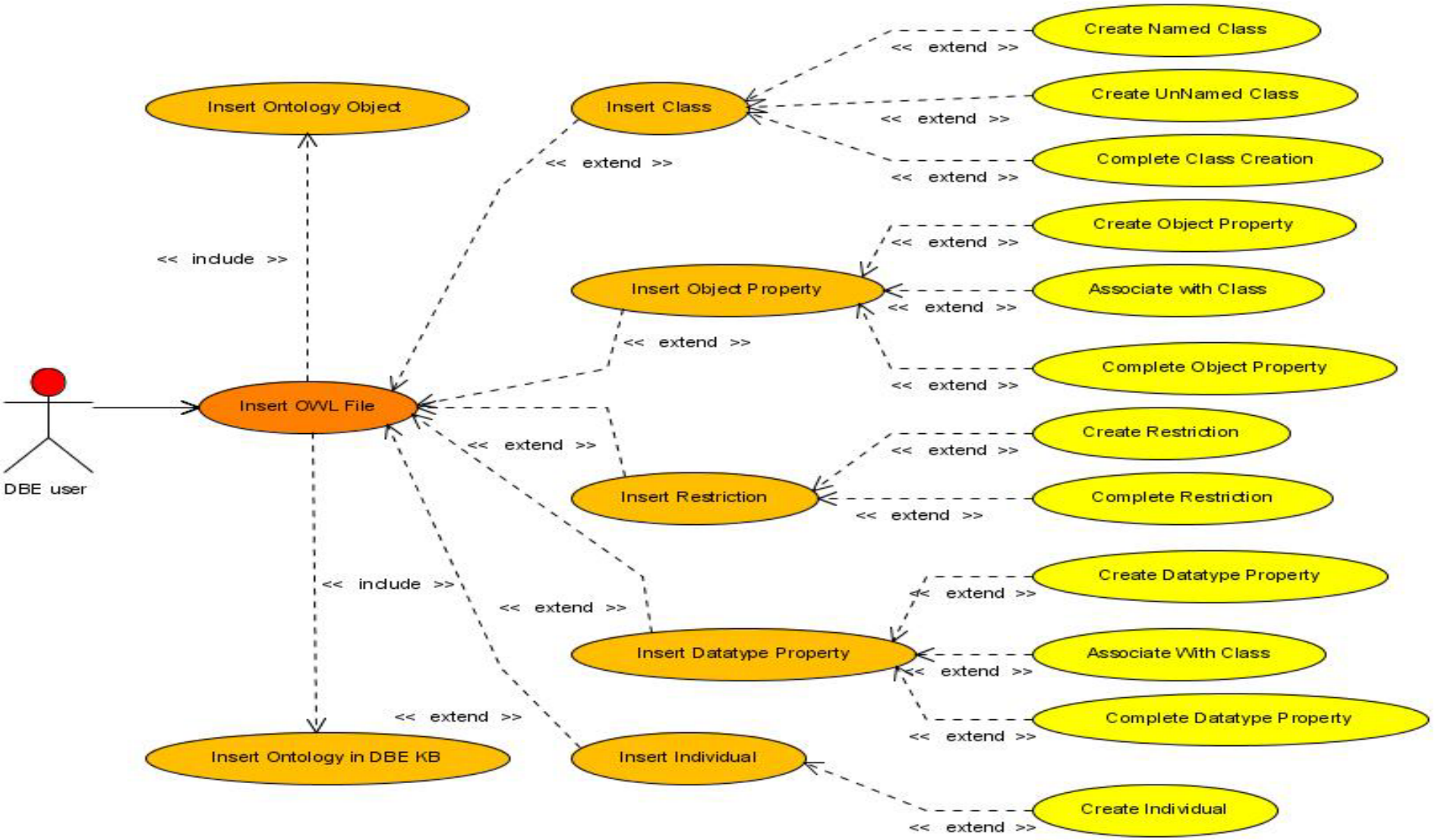



**Figure 2: Use cases for the OWL Importer module in UML notation**

# 3. The BML Editor Architecture

The BML Editor is a graphical, UML-Like Editor allowing for modeling business and service models. The adoption of Eclipse [8] as the main platform for the Service Factory Environment of DBE posed the technical requirement of developing the BML Editor as an eclipse plug-in that will be plugged as all other tools into the DBE Studio [9]. As a consequence, the BML Editor is being developed as an Eclipse plug-in based on Eclipse 3.1.2, and exploiting of the GEF 3.1.1 [10] and Draw 2D graphical frameworks that have been built for this platform. In the frame of the MDA [11] approach that has been adopted by the DBE project, the BML Editor allows the creation of M1 models as direct instances of the M2 metamodels that have been defined in DBE under the BML umbrella with the use of MOF 1.4 (Meta-Metamodel).

The Ontology Viewer is also an autonomous Eclipse plug-in based on Eclipse 3.1.2 that is used by the BML Editor and other tools of the DBE Studio.

The BML editor is used to define business and service models. From an architectural point of view, the BML Editor exploits the DBE Knowledge Base for its persistency requirements. Service-Oriented architecture is followed for integrating the BML Editor with the DBE Knowledge Base based on the DBE ServENT platform (Swallow project [12]). The BML Editor connects with a DBE ServENT and looks up into the network to find an appropriate Knowledge Base (KB) service. The BML Editor using the BML specific JMI [13] interfaces enables the creation of M1 Models and using the functionality of the KB-service is able to store, retrieve and update BML models in the KB in XMI format [14].

Ontology Viewer connects also with a DBE ServENT and looks up to network to find and appropriate KB service. The KB service connected it can be different from the KB service of the BML Editor.

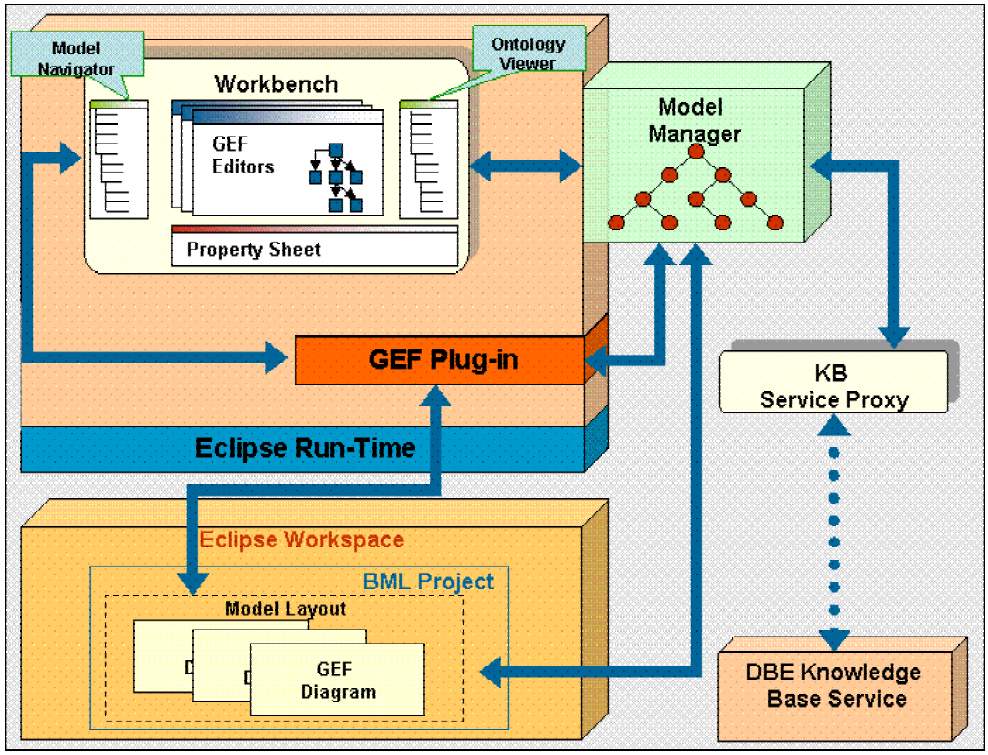



**Figure 3: The BML Editor architecture**

The BML Editor and Ontology Viewer architecture consists of the following components:

## 3.1 The Eclipse Platform

It is the core component of the architecture and provides the development and the run-time environment of the BML Editor and the Ontology Viewer. The Eclipse Platform provides the general working environment (workbench), a workspace area, the capability to manage graphical objects using the GEF plug-in, a basic menu, help wizards etc. The Eclipse Run-Time Environment is responsible to detect the BML Editor and the Ontology Viewer plug-ins, to interpret their file manifest (plugin.xml) and register them as new plug-ins in the Eclipse platform.

## 3.2 The Eclipse Workbench

It is the component that provides the graphical user interface (GUI) of the Eclipse Platform. The graphical environment of the BML Editor is based on the Eclipse Workbench and consists of its own Eclipse perspective, the DBE Business Analysis perspective. The perspective contains a set of special views, editors, menus, toolbars located in specific positions of the graphical environment. The graphical environment consists of three main sub-components:

- **The Model Navigator:** This component is responsible for the presentation of the business models that have been created by an SME. The models are displayed in a tree-view hierarchy and they are located as nodes in the first level of the tree hierarchy.
- **The Graphical Diagrams Editor:** The service and business description models are presented as UML-like graphical diagrams. The diagrams contain a group of graphical objects that are handled by a GEF editor, which is responsible for the management, and editing of these objects. The BML Editor is responsible to manage the separate diagrams as a whole, so as a model can be presented in many different diagrams.
- **Model Manager:** It is the most important component of the system because it is responsible for the management of the model processing. Every action related to the processing of the active model and any interaction between the various components of the system is explicitly managed by the Model Manager. The component also provides the appropriate data structures that represent the model semantics and are displayed in the GUI.
- **The Properties Sheet:** This component is responsible for the presentation and editing of the properties values of the service and business description entities. The synchronization mechanism keep the Properties Sheet up-to-date in any changes happened in the Model Explorer or the Graphical Diagrams Editor.

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

## 3.3 The Ontology Viewer

The Ontology Viewer is also an autonomous Eclipse plug-in based on Eclipse 3.1.2 that is used by the BML Editor and other tools of the DBE Studio for exploring ODM and OWL-DL ontologies. It provides a tree-view representation of an ontology that allows for the browsing and navigation of it. In this way the modeller can use ontology concepts as types

in the models that designs taking full advantage of the expressiveness provided by ODM and OWL-DL ontologies.

### 3.3.1 The OWL Importer Module

Ontology Viewer, exploiting the functionality provided by the OWL Importer Module, supports the importing of OWL-DL ontologies in the DBE KB. There is a mapping between ODM and OWL-DL as described in detail in [2]. The OWL Importer Module uses the Jena Framework [15] and JMI interfaces providing by DBE KB in order to transform an OWL-DL ontology into an ODM ontology and to store it in the DBE KB.

The conceptual architecture of the OWL Importer Module is depicted in the figure that follows:

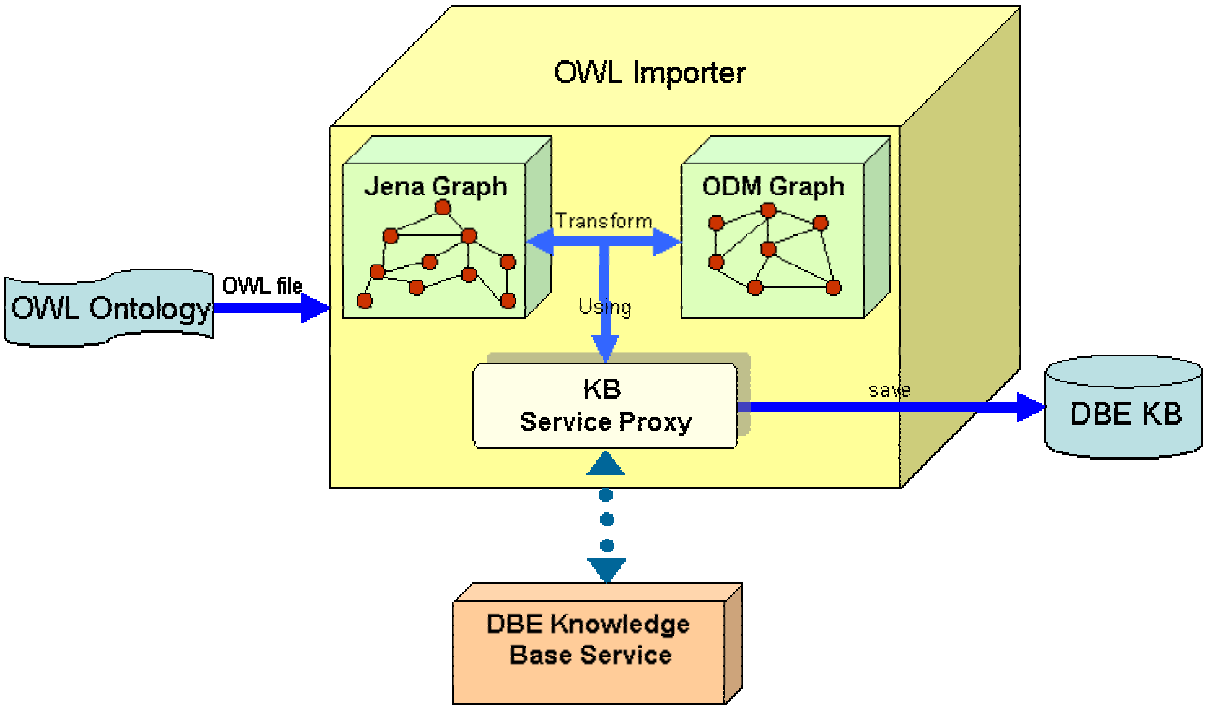



**Figure 4: The OWL Importer Module Conceptual Architecture**

## 3.4 The DBE ServENT (DBE Proxy Service)

This service provides a Proxy mechanism that enables the BML Editor and the Ontology Viewer to communicate with the DBE Knowledge Base Service. Using the Proxy service, the BML Editor has access to the following functionality:

- Browsing the models saved in the DBE Knowledge Base
- Saving a new model in the DBE Knowledge Base
- Retrieving a model from the DBE Knowledge Base

while the Ontology Viewer allows for:

- Browsing the ontologies that exist in the DBE Knowledge Base
- Searching for a particular concepts in an ontology
- Importing an OWL-DL ontology into the DBE Knowledge Base.

### 3.4.1 The DBE Knowledge Base Service

This service is part of the DBE **K**nowledge **R**epresentation **F**ramework (KRF) and provides an API based on the JMI standard [13] that enables the BML Editor to access the

functionality of the DBE Knowledge Base in order to retrieve an existing model or save a new model in the Knowledge Base and the Ontology Viewer to retrieve and browse ontologies.

# 4. Installation Instructions

In order to use BML Editor you must have installed Java 1.4.x or Java 1.5.x, Eclipse 3.1.2 as well as GEF 3.1.1. BML Editor is an eclipse plugin 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/.

BML Editor 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/bml-editor the following plugins:

- org.dbe.studio.editors.bml_0.2.1.zip
- org.dbe.studio.tools.kb-toolkit_0.2.1.zip
- org.dbe.studio.tools.ontologyviewer_0.2.1.zip
- org.dbe.studio.tools.queryformulator_0.2.1.zip
- org.dbe.studio.core.perspectives_0.2.2.zip
- org.dbe.studio.core.preferences_0.2.1.zip

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

# 5. User Guide

## 5.1 Starting BML Editor

A DBE Business Analysis Perspective has been created for BML Editor. So, the BML Editor can be activated selecting, from Eclipse main menu, Window->Open Perspective->Other-> DBE Business Analysis. The BML Editor main window is presented in the following figure:

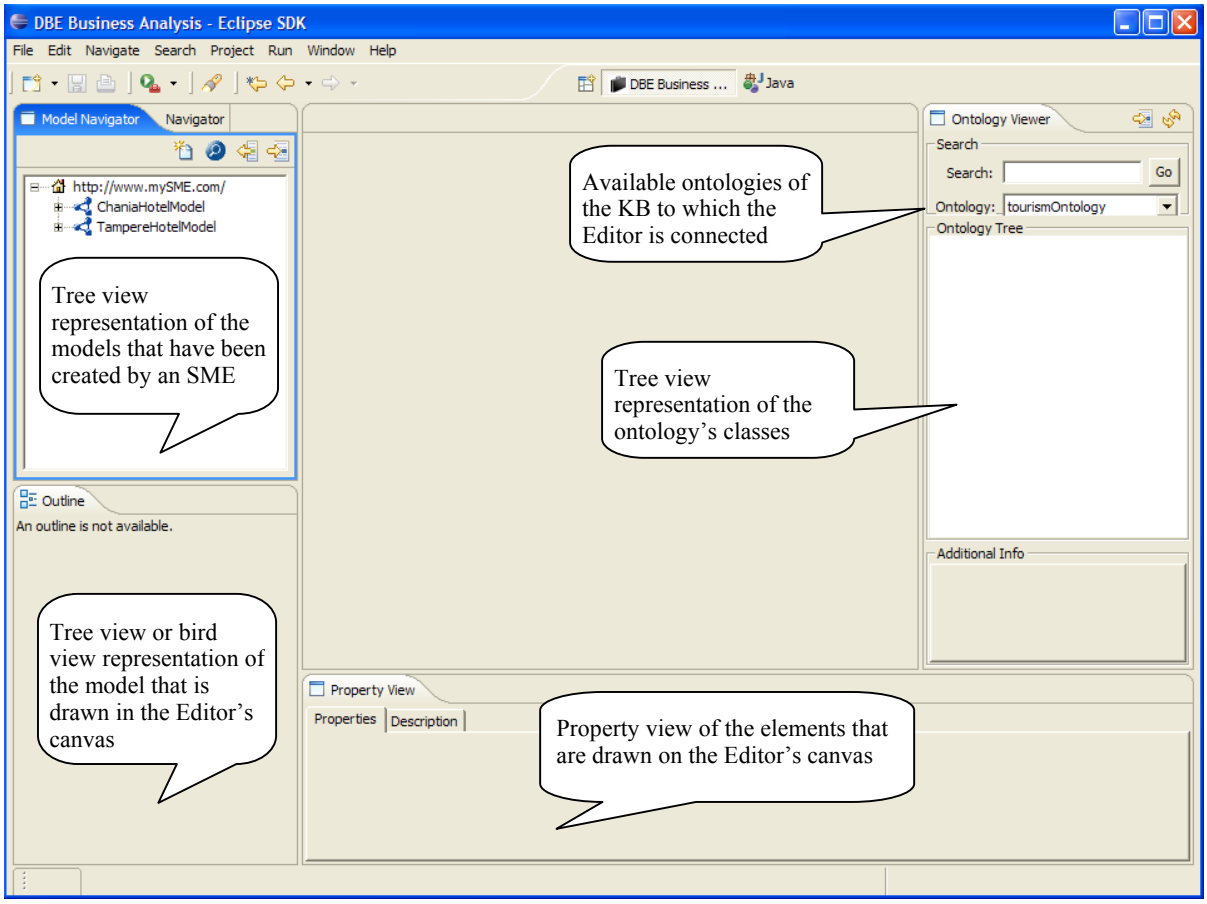



**Figure 5: BML Editor main window**

When the editor is activated it attempts to connect to a particular KB service in order to retrieve the list of models that have been created by a specific SME. An SME is identified by its SME ID, a unique identifier for an SME like a URI. By default the BML Editor attempts to connect to the address http://localhost:2728 assuming that there is a local, running, instance of the KB while the SME identifier is set to http://www.mySME.com/. The user can change these values as well as the values of the Ontology Viewer selecting from Eclipse main menu Window->Preferences->DBE Studio->BML Editor (or Ontology Viewer) like it is shown in the following figure:

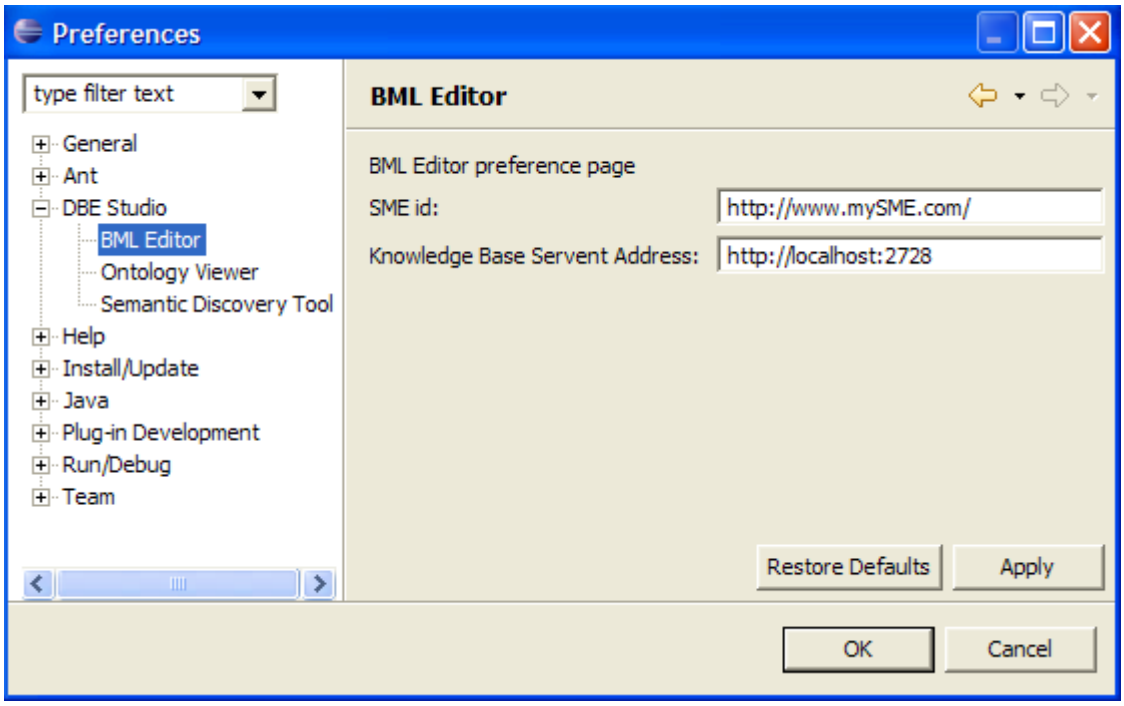



**Figure 6: The BML Editor Preferences dialog**

## 5.2 Creating a new model

In order to create a new model the user has to click on the icon in Model Navigator View. The editor window is shown in the next figure:

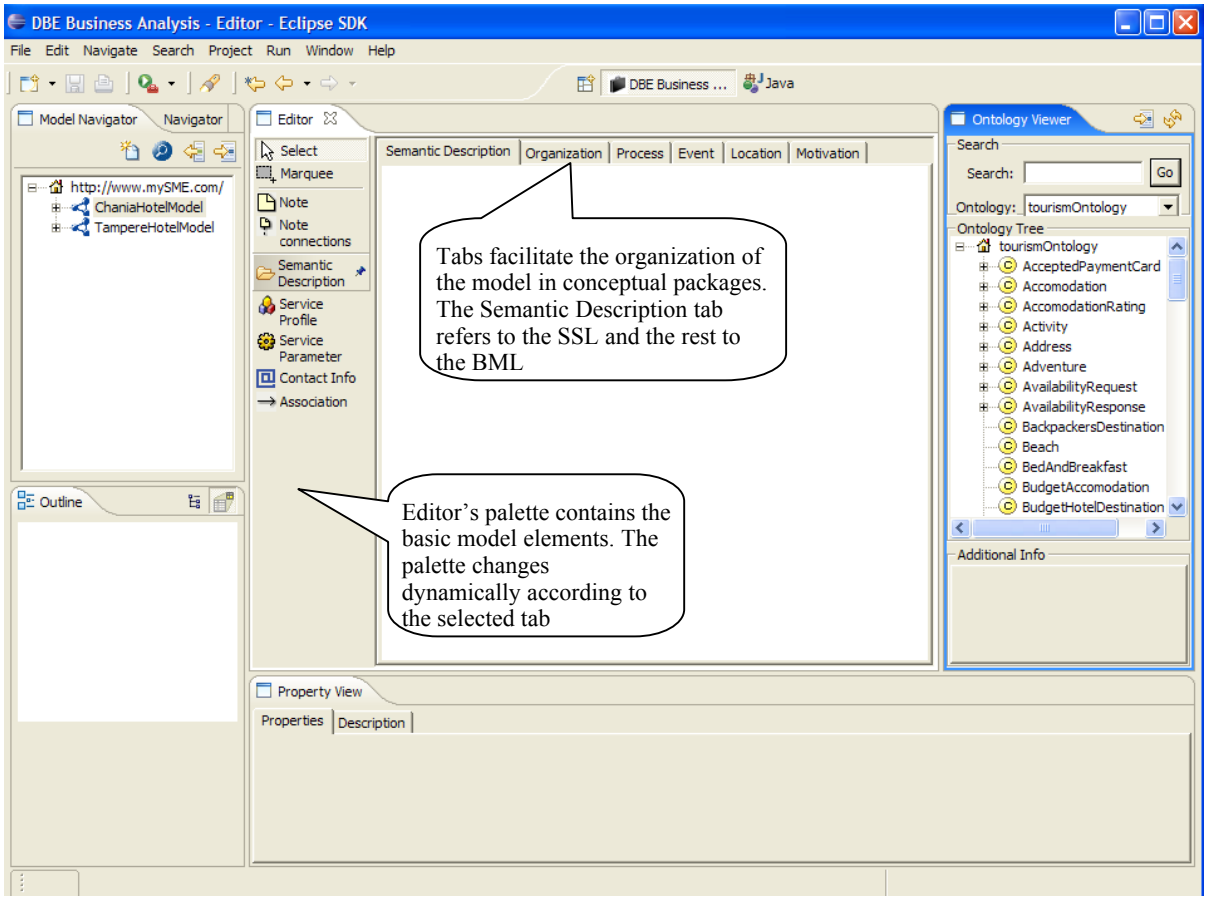



**Figure 7: The BML Editor main window, creating a new model**

## 5.3 Creating Semantic Descriptions of Services (SSL)

### 5.3.1 Creating a Service Profile

The first thing a user has to do in order to create the semantic description for a service is to create a Service Profile. This can be done selecting from the Editor's palette the corresponding tool and clicking on the Editor's canvas. A Service Profile is visually represented by a rectangle separated in tree areas. The first area (travelling top-down) contains the name of the Service Profile, the second one the attributes that a Service Profile may contains and the third the functionalities that can be defined as part of the particular profile.

Right-clicking in the area bellow the Service Profile name (the attributes area) a menu will appear. Selecting the *"Add attribute"* item one new attribute will be created. Following the same way the user can add as many attributes as he wants. Similar is the procedure for adding functionalities as shown in the next figures:

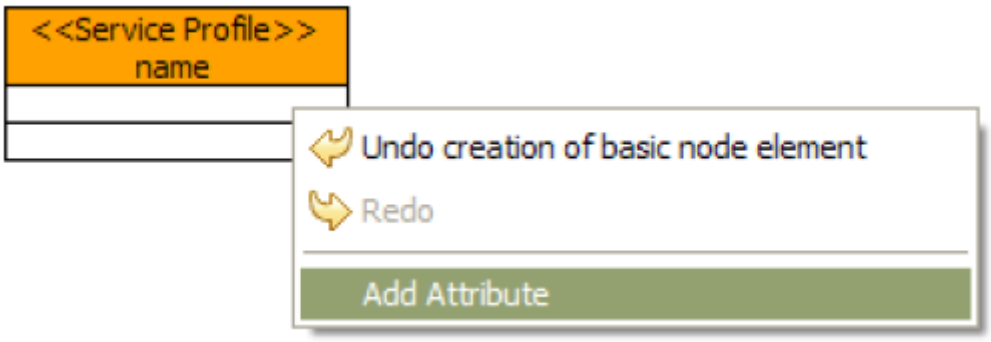



**Figure 8: Adding an attribute to a Service Profile**

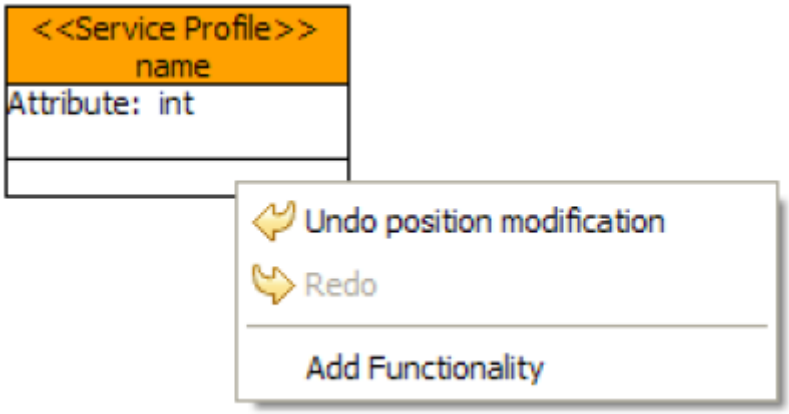



**Figure 9: Adding a functionality to a Service Profile**

The editing of the profile's name, attribute's name and type as well as functionalities can be done in the Property View selecting the ServiceProfile, the attribute and the functionality correspondingly as shown in the following figure:

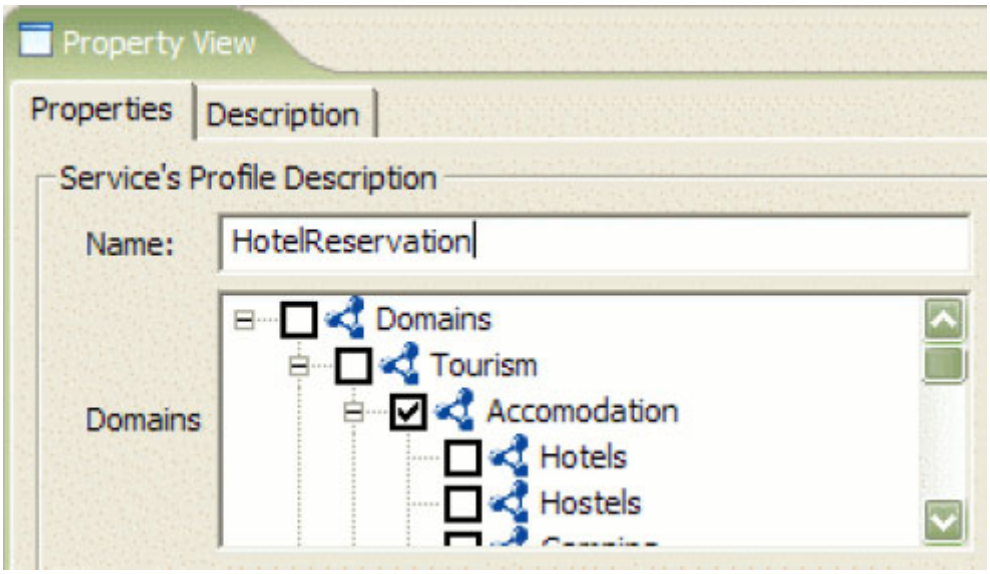



**Figure 10: A service Profile has a name and can be associated with many domains**

Service functionality has name and input and output parameters. Each parameter also has name and type. The user can add input ( ) and output ( ) parameters or delete input ( ) and output ( ) ones by selecting the functionality he wants and

clicking on the corresponding icons of the Property View. Double-clicking on the corresponding cells in the "Name" and "Type" columns can edit the name and the type of an input/output parameter as show in the next figure:

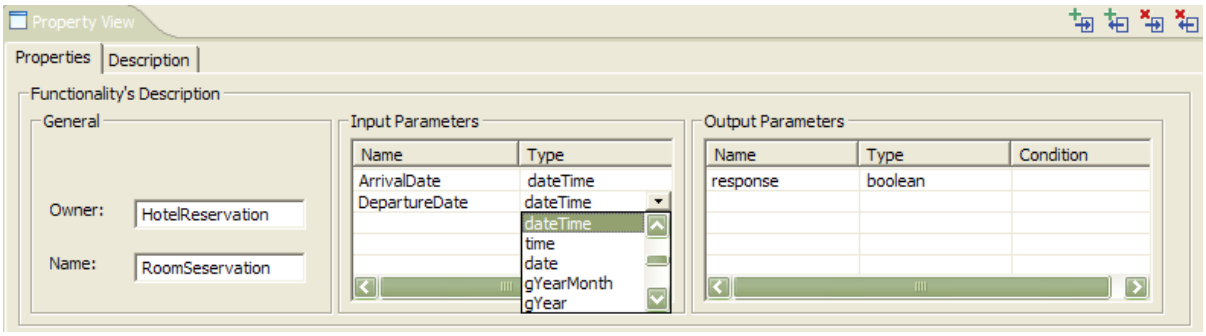



**Figure 11: Service Functionality has a name and input and output parameters. Each parameter has name and type**

### 5.3.2 Using Ontology Concepts as types

ServiceProfile's attributes as well as the parameters of a service functionality have name and type. The type can be primitive (int, float, string, etc.) or can be a class from an Ontology. In the previews examples we used only primitive types. In this section it will be described the usage of an Ontology's class as a type.

The usage of an Ontology's class as type can be achieved through three steps. The first step is to select from the type combo box the item "ConceptID" as shown in the next figures:

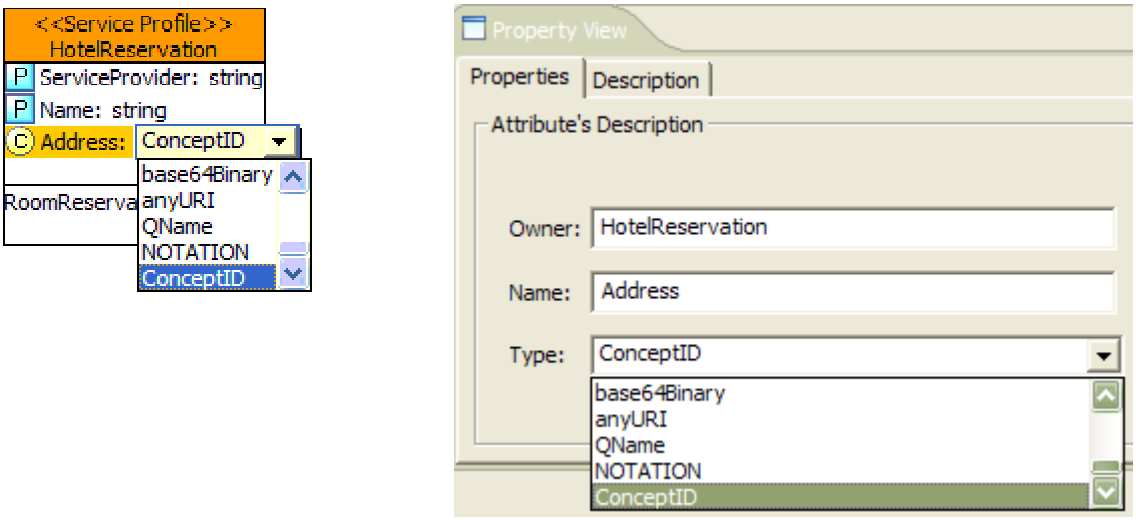



**Figure 12: First step in using an Ontology' s class as attribute type**

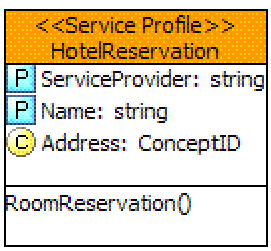



**Figure 13: Second step in using an Ontology's class as attribute type**

The next step is to use the Ontology Viewer in order to browse an Ontology. The Ontology Viewer connects to a KB node and lists the available ontologies ("Ontology" combo box). The user can select the ontology he wants and after that is able either to browse the whole ontology or to search for a particular class. If he writes nothing in the "Search" field and clicks the "Go" button the tree view of the selected ontology will be shown while if he writes something in the search field and presses "Go" he will get back the class (actually a whole part) of the Ontology that matches his request. The above are shown in the next figures:

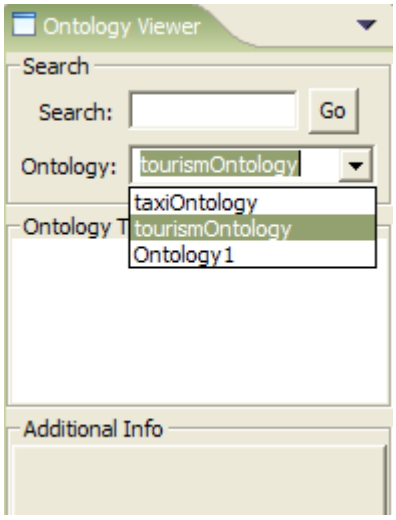



**Figure 14: Select the ontology you want**

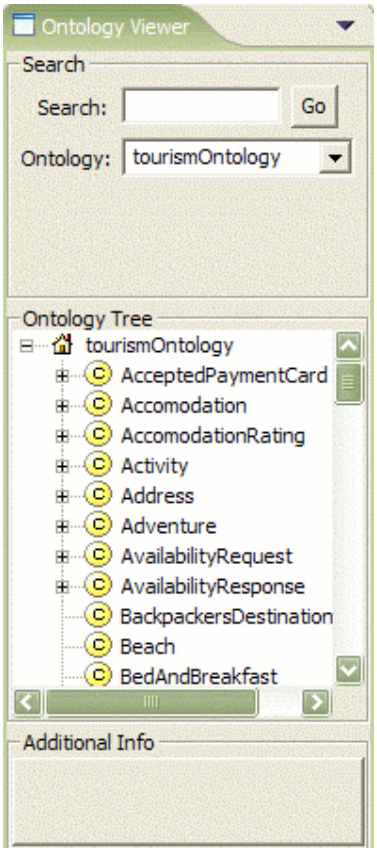



**Figure 15: Browse the ontology**

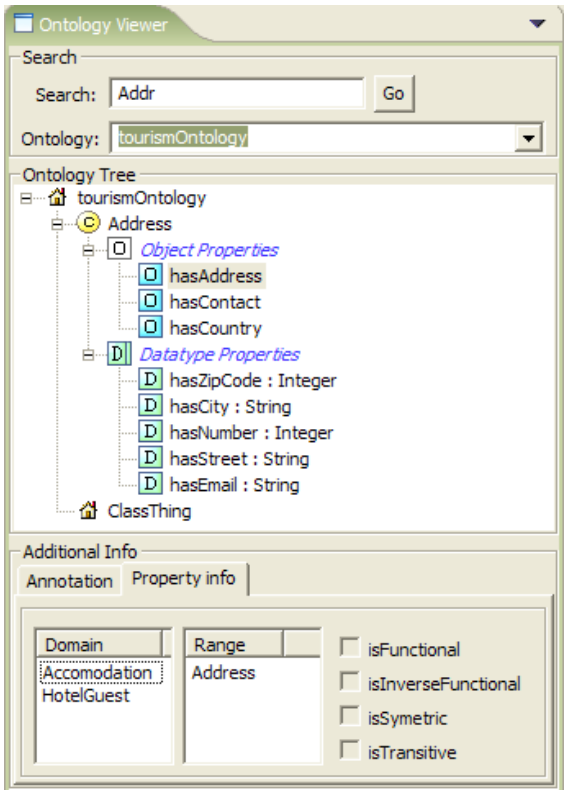



**Figure 16: Search for an ontology class. Partial matching is supported**

The final step is to double click on the class you wish. After that the class name replaces the type "ConceptID" of the attribute and becomes the attribute's type as shown the next figure:

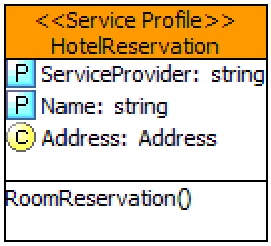



**Figure 17: Using an ontology class as type**

Users can exploit either ODM ontologies that have been created using the Ontology Analysis Tool [16] or OWL-DL ontologies. In the latter case the user can import an OWL-DL ontology taking advantage of the Ontology Viewer's OWL Importer module by clicking on the  icon of the Ontology Viewer and selecting an OWL-DL file from her/his local file system as shown in the figure that follows:

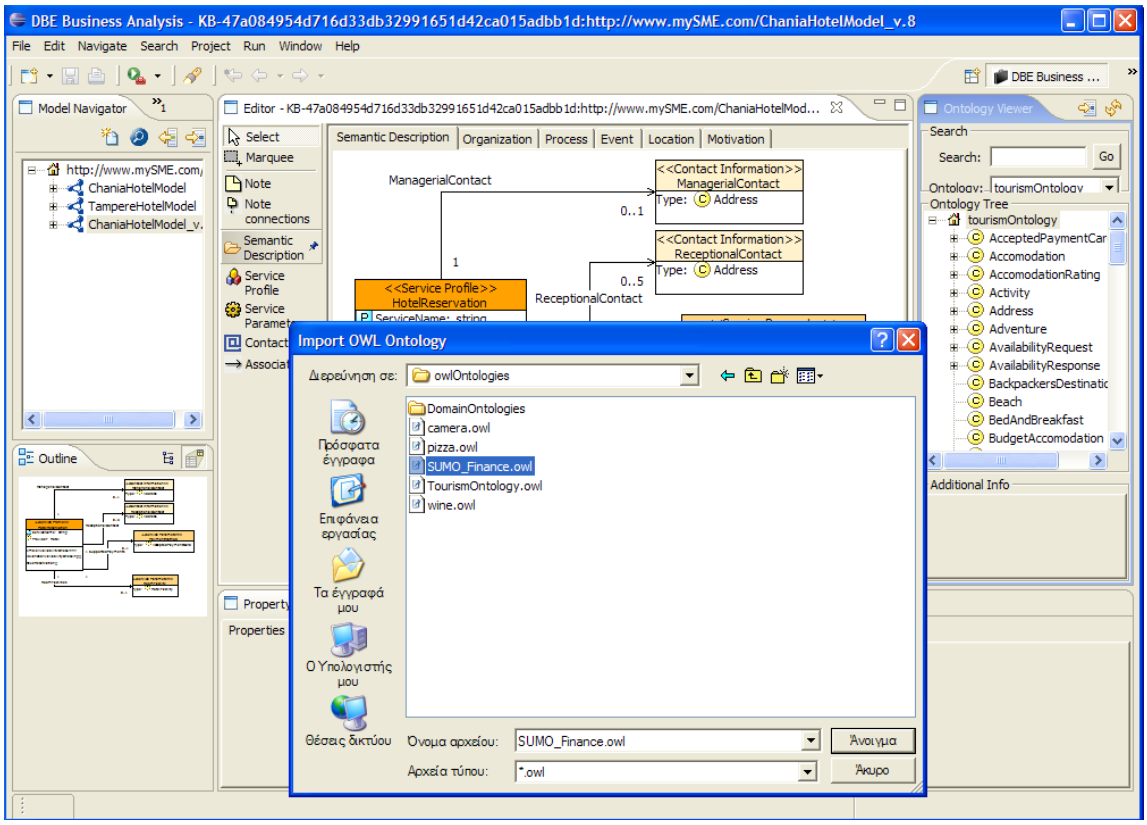



**Figure 18: Importing an OWL-DL ontology**

As already mentioned, the BML Editor connects to a KB node in order to save/retrieve BML models. The Ontology Viewer is also connected to a KB node in order to get access to the available ontologies. However, it is worth to remind that these two nodes can be different, in other words, the BML Editor can be connected to a KB node for saving its models and the Ontology Viewer can be connected to another in order to retrieve the available ontologies. The user can change the KB node address to which Ontology Viewer is connected through the Window->Preferences->DBE Studio-> Ontology Viewer dialog.

### 5.3.3 Creating Service Parameter and Contact Information elements

The user is able to create Service Parameter and Contact Information elements selecting from the Editor's palette the corresponding tools and clicking on the editor's canvas. Service Parameter and Contact Information element have a name and a type, which can be either primitive or an ontology class.

### 5.3.4 Creating Associations

The user can create associations between a Service Profile and Service Parameter and Contact Information elements selecting from the editor's palette the Association tool and clicking on the source and target elements. An association has a name and cardinality constraints that can be edited either in the Property View or by clicking twice on them. The BML Editor takes care for the restrictions imposed by the SSL metamodel allowing only the creation of associations between the suitable elements. An example is shown in the figure that follows:

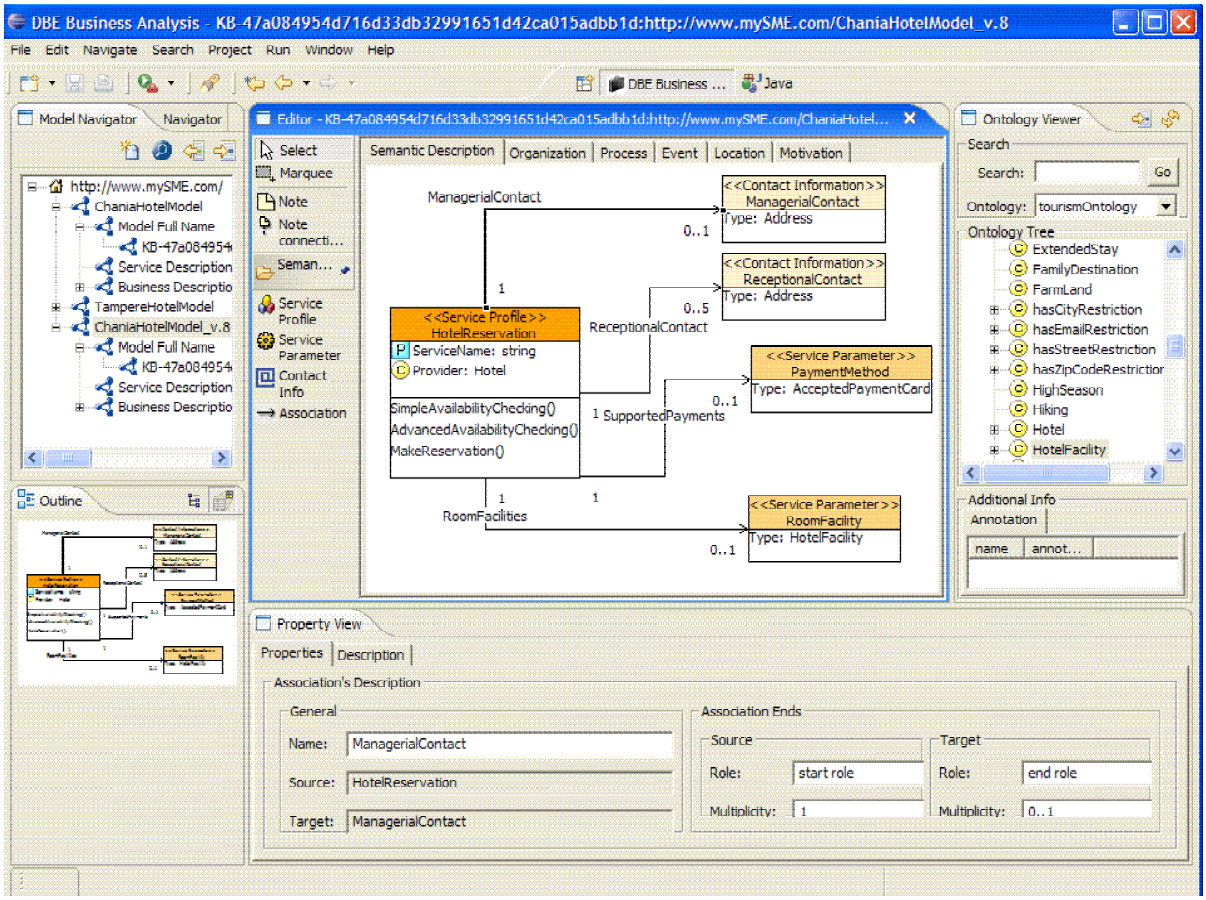



**Figure 19: Creating and editing associations**

# 5.4 Creating Business Model Descriptions (BML)

## 5.4.1 Creating Business Elements

A business model is conceptually divided into five parts: Organization, Process, Event, Location, Motivation and for each part there is the corresponding tab in the editor's main window. According to the active tab the editor's palette changes dynamically providing the available model elements. Selecting an element from the palette and clicking in the canvas the element's visual representation is rendered. Each business model element has a name and it may contain many attributes. An attribute has name and type, which can be either primitive or an Ontology's class. The steps for creating and editing attributes are the same as for a ServiceProfile.

## 5.4.2 Creating Business Associations

Associations among business elements are created by selecting the BML Association tool from the palette and clicking on the source and target elements. The BML metamodel provides particular associations with predefined names and cardinality constraints. The BML Editor provide for the restrictions imposed by the BML metamodel allowing only the creation of associations between the appropriate elements. The following tables describes the allowed associations:

***Organization***

| *Source Element* | *Target Element* | *Association* |
|---|---|---|
| BusinessEntity | Asset | owns asset |
| BusinessEntity | BusinessItem | provides |
| BusinessEntity | Product | provides |
| BusinessEntity | Service | provides |
| BusinessEntity | NetworkRole | performs |
| BusinessEntity | BusinessEntity | owns subunit |
| Network | NetworkRole | has role |

***Process***

| *Source Element* | *Target Element* | *Association* |
|---|---|---|
| Asset, BusinessItem, Product, Service (from Organization) | BusinessProcess, BusinessActivity, CollaborationActivity, Transaction | is used in, produces |
| Event (from Event) | BusinessProcess, BusinessActivity, CollaborationActivity, Transaction | begins, ends, generates |
| BusinessProcess, BusinessActivity, CollaborationActivity, Transaction | Commitment | fulfils |
| BusinessProcess, BusinessActivity, CollaborationActivity, Transaction | Role | involves actor |
| BusinessProcess, BusinessActivity, CollaborationActivity, Transaction | BusinessProcess, BusinessActivity, CollaborationActivity, Transaction | Precedes |
| Constraint | BusinessProcess, BusinessActivity, CollaborationActivity, Transaction | before, after, during |
| BusinessProcess | BusinessProcess | sub process |
| BusinessProcess | BusinessActivity, CollaborationActivity | owns task |

| *Source Element* | *Target Element* | *Association* |
|---|---|---|
| CollaborationActivity | Transaction | contains transaction |
| Asset, BusinessItem, Product, Service (from Organization) | Commitment | Reserves |
| Contract | Commitment | Establishes |
| Commitment | Role | Involves |
| BusinessEntity | Role | Performs |
| Contract, Commitment | Event (from Event) | Implies |

***Event***

| *Source Element* | *Target Element* | *Association* |
|---|---|---|
| Event | Event | precedes |

***Location***

| *Source Element* | *Target Element* | *Association* |
|---|---|---|
| Path | Path | includes |
| Path | LocationType | is toward |
| LocationType | Path | is from |
| LocationType | BusinessEntity (from Organization) | is in |

**Motivation**

| *Source Element* | *Target Element* | *Association* |
|---|---|---|
| Means, End, Influence, Assessment | BusinessEntity (from Organization) | has |
| Means, End, Influence, Assessment | any Business Element | impacts on |

It should be mentioned that the BML Editor protects the user from transitioning to invalid states. For example it does not allow the creation of circles (or deadlocks) as shown in the next figures:

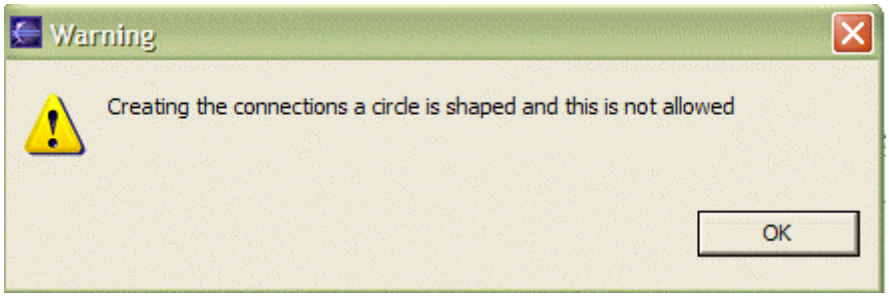



**Figure 20: Warning message informing the user for the creation of a circle**

### 5.4.3 Reusing business elements

The BML metamodel allows for the usage of elements defined in one part of the model in other parts. For instance, an Asset element, defined in the Organization part of the model, can be used (referenced) in the Process part of the Model. The BML Editor supports this need through the Outline View. The particular view provides to different looks of the model that is opened in the editor's main window, a bird view ( ) and a tree view ( ) as it is shown in the next figures:

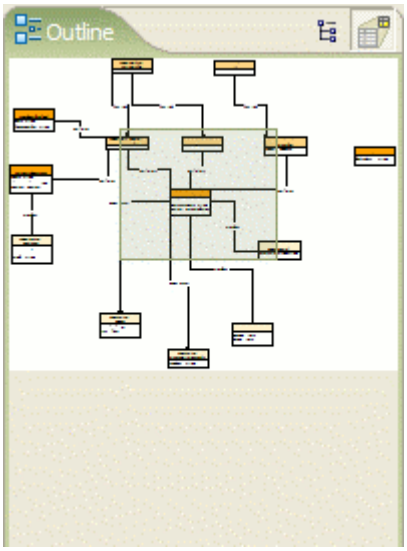



**Figure 21: Bird view model representation that facilitates the navigation in a big model**

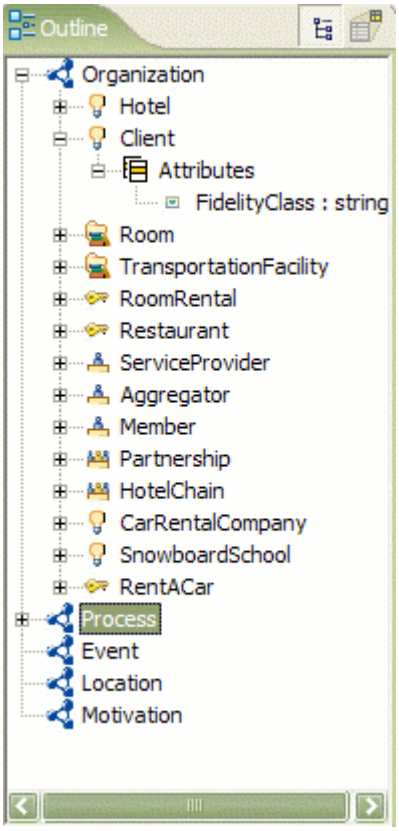



**Figure 22: Tree view model representation**

The user can drag and drop any element he wants from the tree view into the BML Editor main window (ensuring that does not violate the BML metamodel, actually BML Editor takes care of that). For example, a Business Entity element is defined in the Organization part (tab) of the model but it can also be used (referenced) in the Process part. So, the user can define such an element in the Organization tab and after that

can drag it from the tree in the Outline View into the Process tab in order to use it in this part of the model.

It should be noted that BML Editor takes care that the BML metamodel will not violated during this process. For example, if you drag a Network element (defined in the Organization part) into the Process part the editor will not allow this action because the BML metamodel does not support it. In addition if the user attempts to delete an element that is referenced in other parts of the model the BML Editor will not allow it and will provide a warning message.

## 5.5 Saving and Exporting Models

After the creation of a model the user has two options, either to save it in the DBE KB clicking on the save icon or to export the model in the local file system clicking on the  icon of the Model Navigator view.

### 5.5.1 Saving a model

The BML Editor exploits the versioning mechanism provided by the DBE KB and every time the user saves a model a new version of it is created. The identification format that is followed by the versioning mechanism is: *KB_NODE_ID:SME_ID:ModelName_vX* where X is a number and represents the consecutive number of the version. The following figure depicts the two versions of a model:

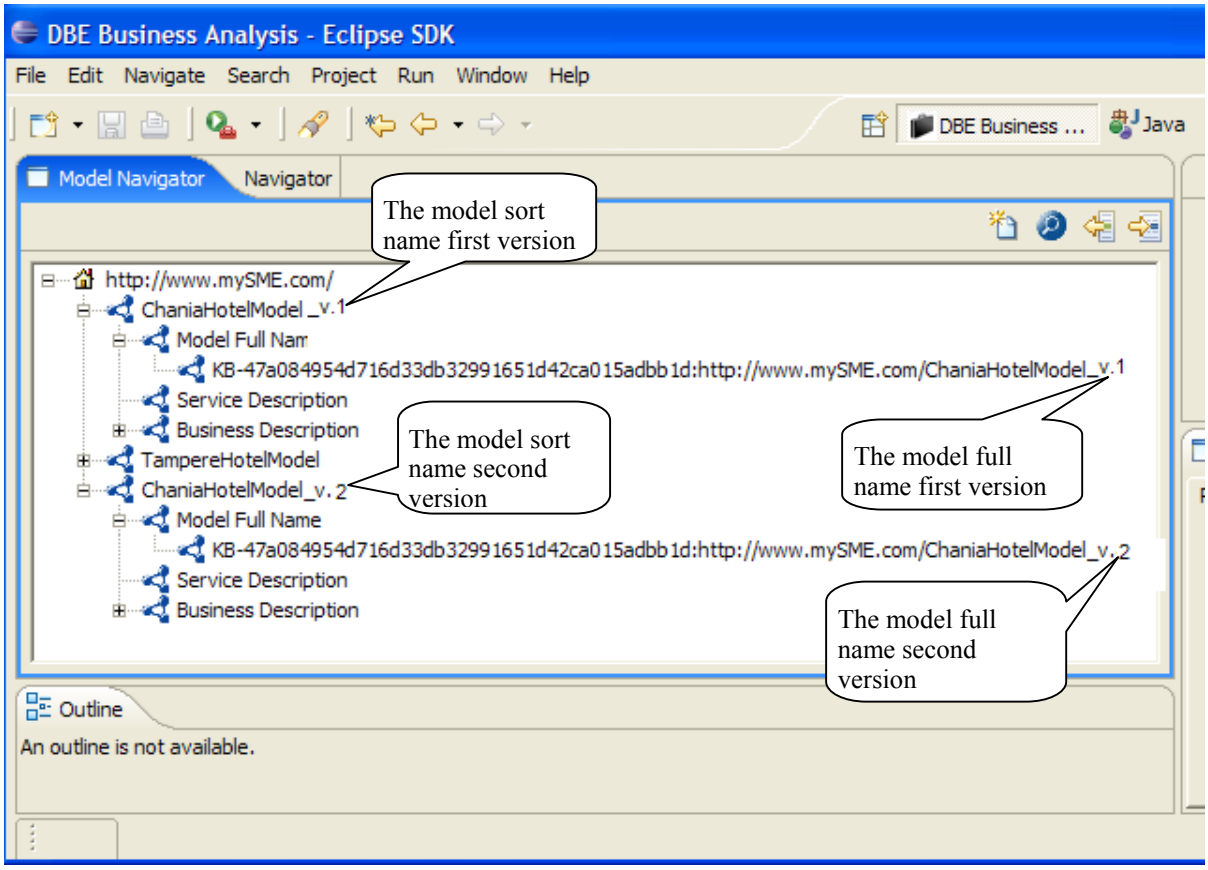



**Figure 23: Saving a model a new version of it is created**

### 5.5.2 Exporting a model

The user can export a model to the local file system clicking on the icon of the Model Navigator view. A file chooser dialog is appeared asking for the name of the file into which the model will be exported. When exporting a model to the local file system two files are actually created. The one of them has *.bml.xmi* extension and contains the XMI representation of the model according to the corresponding metamodel. The other file has a *.bml* extension and contains also the diagrammatic information of the model.

The user can also import a model from the local file system clicking on the of the Model Navigator view. A file chooser dialog is appeared asking for the file to be imported. The user must choose the file with the *.bml* extension.

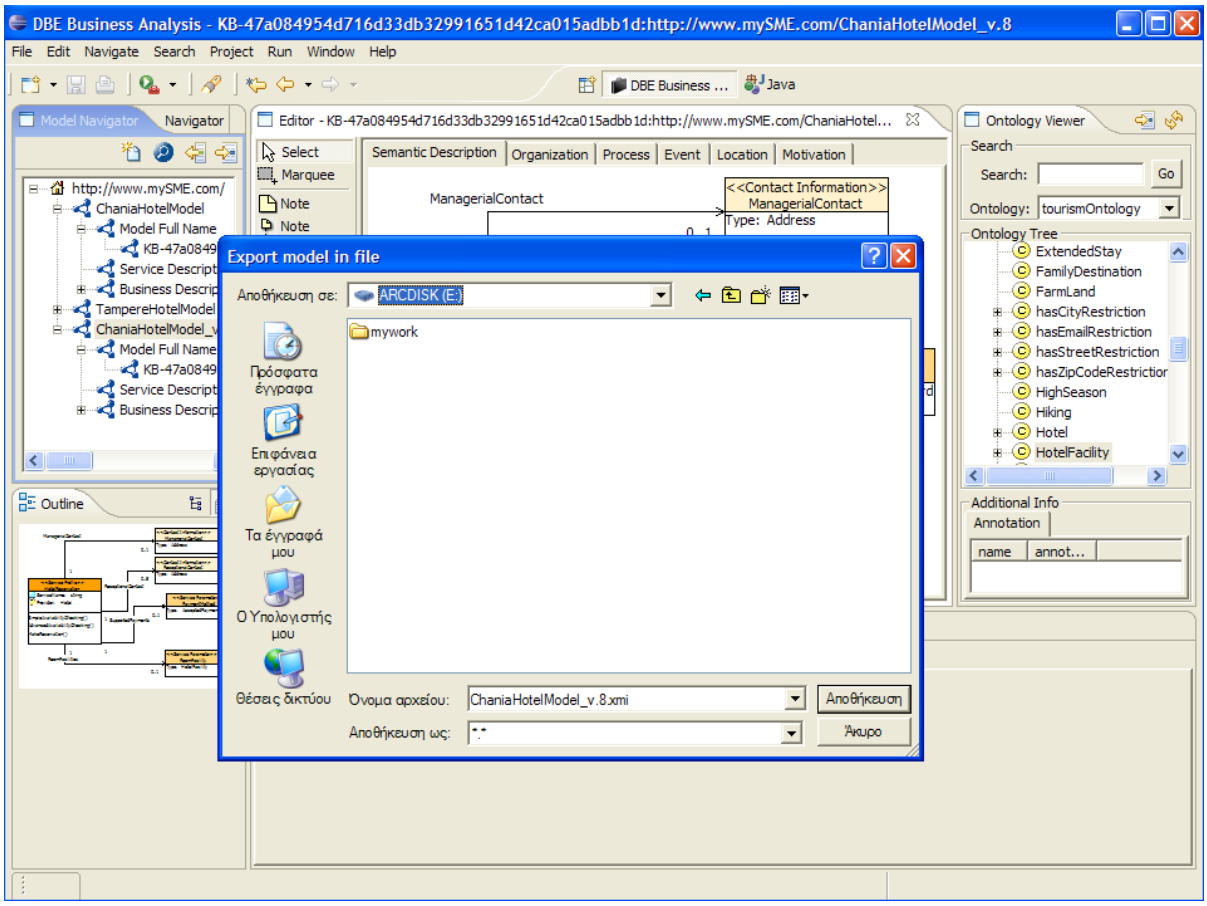



**Figure 24: Exporting a model to the local file system**

# 6. Searching for models

The user is able to search for models through BML Editor. The search process is provided by another plugin the Query Formulator-Semantic Discovery Tool [17] that can be called by BML Editor clicking on the  icon in the editor's tool bar. Using this plugin the user is able to formulate queries for existing M1 models by posing constraints on the attributes of primitives that the M2 metamodels (SSL and BML) provide. The user can define soft and hard constraints. The desired models may satisfy the soft constraints, but they have to satisfy the hard ones (see Semantic Discovery Tool Guide for details). Finally the user is able to open, in the editor, a model that was found.

# 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 |
| DBE KRF | DBE Knowledge Representation 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. |
| 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: 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. It has three increasingly expressive sublanguages: OWL Lite, OWL DL, and OWL Full. |
| 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. |
| 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: A standard specification based on XML. |

# 8. Bibliography[1]


[1] DBE: Digital Business Ecosystem, http://www.digital-ecosystem.org

[2] TUC, DBE Deliverable, D14.1 – DBE Knowledge Representation Models, May 2005

[3] ISUFI, DBE Deliverable, D15.1: Business Modelling Language 1.0, May 2005

[4] Object Management Group – OMG, "MetaObject (MOF) Facility Specification", http://www.omg.org/mof, 2002

[5] World Wide Web Consortium - W3C, "Web Ontology Language (OWL)", http://www.w3.org/2004/OWL/

[6] TUC, DBE Deliverable, D15.2: BML Editor Second Release

[7] Cockburn Alistair, Writing Effective Use Cases (The Crystal Collection for s/w professionals), Addison-Wesley, 2001

[8] Eclipse Platform Technical Overview, Object Technology International Inc., February 2003 (updated for 2.1; originally published July 2001).

[9] INTEL, DBE Deliverable, D26.3 DBE Studio Integration, April 2006

[10] GEF Overview, http://www.eclipse.org/gef/

[11] MDA Guide Version 1.0.1: <http://www.omg.org/docs/omg/03-06-01.pdf>, 2003

[12] The DBE Swallow project: http://swallow.sourceforge.net

[13] Java Metadata Interface, http://java.sun.com/products/jmi/index.jsp

[14] OMG XML Metadata Interchange (XMI) Specification v1.2 http://www.omg.org/cgi-bin/apps/doc?formal/02-01-01.pdf, 2002

[15] http://jena.sourceforge.net/

[16] TUC, DBE Deliverable, D15.4: Ontology Creator/Importer/Viewer, July 2006

[17] TUC, DBE Deliverable, D24.6: The Query Formulator and Semantic Discovery Tool Final Release, June 2006


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