

# MOMOCS

## Model driven Modernisation of Complex Systems

### DELIVERABLE # D51B DATA MODERNIZATION TOOL IMPLEMENTATION (XSM KNOWLEDGE BASE REPOSITORY)

---

Dissemination Level:	Public
Work package:	WP5
Lead Participant:	Atos Origin SAE
Contractual Delivery Date:	M17/M24
Document status:	Final

Preparation Date: 16 September, 2008

Document Version: 1.6

a

## Revisions

Document Version	Date	Author (Partner)	Description of Changes
0.1	22 Jan. 08	Yosu Gorroñogoitia (Atos)	ToC
0.2	24 Jan. 09	Luis Quijada (Atos)	Section 4 – ToC modification
0.4	12 Feb. 08	Yosu Gorroñogoitia (Atos)	Several sections fulfilled.
0.5	18 Feb. 08	Luis Quijada, Yosu Gorroñogoitia (Atos)	Several sections fulfilled.
1.0	19 Feb 08	Luis Quijada, Yosu Gorroñogoitia (Atos)	Final editing.
1.1	3 Mar 08	Luis Quijada (Atos)	Feature installation
1.2	4 Mar 08	Luis Quijada (Atos)	Advanced search section and FAQ
1.3	24 Jul 08	Yosu Gorroñogoitia (Atos)	Semantic search section, add attachment section, semantic search algorithm annex
1.4	28 Jul 08	Yosu Gorroñogoitia (Atos)	Model diagram opening as image files, models comparison
1.5	16 Sep. 08	Yosu Gorroñogoitia (Atos)	KBR Ontology design pattern  Summary of changes from v1.2.
1.6	22 Sep. 08	Yosu Gorroñogoitia (Atos)	Minor changes after internal review.  Wording to address reviews recommendations

## INDEX

<b>1</b>	<b>SCOPE .....</b>	<b>9</b>
1.1	PURPOSE OF THE DOCUMENT .....	9
1.2	DOCUMENT STRUCTURE.....	10
<b>2</b>	<b>KB REPOSITORY INTRODUCTION .....</b>	<b>12</b>
2.1	INTRODUCTION.....	12
<b>3</b>	<b>REPOSITORY GETTING STARTED.....</b>	<b>15</b>
3.1	REQUIREMENTS.....	15
3.2	KB REPOSITORY TOOL LICENSE.....	17
3.3	INSTALLATION GUIDE .....	17
3.3.1	Installing the tool making use of the Windows installer .....	17
3.3.2	Installing the tool via Eclipse Update Manager .....	21
3.3.2.1	QuickImage Editor Installation (Optional).....	26
3.3.2.2	EMF Compare Plugin (Optional) .....	27
3.4	GETTING STARTED .....	28
3.4.1	KB Repository configuration .....	28
3.4.2	KB Repository workbench .....	29
<b>4</b>	<b>KB REPOSITORY USER'S GUIDE .....</b>	<b>31</b>
4.1	KB REPOSITORY WORKBENCH PERSPECTIVE .....	31
4.1.1	KB Repository Explorer View .....	37
4.1.1.1	KB Repository navigation .....	38
4.1.1.2	Ordering and filtering .....	40
4.1.1.3	KB Repository explorer view toolbar.....	41
4.1.1.4	KB Repository navigation tree contextual menu .....	42
4.1.2	KB Repository artefacts properties editor.....	42
4.1.3	KB Repository Ontology Browser .....	44
4.1.3.1	Ontology change .....	45
4.1.4	KB Repository Semantic Annotation View .....	46
4.1.5	Informative / auxiliary views.....	47
4.1.5.1	Historic view .....	47
4.1.5.2	Query results view .....	50
4.1.5.3	KB Attachments view .....	51
4.1.5.4	KB Notes view .....	55
4.1.6	KB Repository Menu and button toolbar .....	56
4.2	KB REPOSITORY ADMINISTRATION GUIDE .....	58
4.2.1	KB Repository domain model elements.....	58
4.2.2	Repository creation / removal .....	59
4.2.3	Artefact creation .....	61
4.2.3.1	Folders.....	61
4.2.3.2	Model and transformation artefacts.....	62
4.2.3.3	Notes and attachments .....	64
4.2.4	Artefact edition .....	66
4.2.5	Artefacts visualisation. Exporting Models and transformations.....	66
4.2.6	Artefact removal .....	68
4.2.7	Artefact annotation .....	69
4.3	KB REPOSITORY AUXILIARY TOOLS .....	69
4.3.1	Repository database management .....	69
4.3.2	Repository export / import tool .....	70

<b>4.4</b>	<b>KB REPOSITORY SEARCH FACILITIES .....</b>	<b>72</b>
4.4.1	Keyword matching search .....	72
4.4.1.1	Search syntax .....	74
4.4.1.2	Advanced Query creation .....	76
4.4.1.3	Browsing results.....	81
4.4.2	Semantic Search.....	81
<b>5</b>	<b>TROUBLESHOOTING.....</b>	<b>84</b>
<b>5.1</b>	<b>KNOWN ISSUES .....</b>	<b>84</b>
<b>5.2</b>	<b>BUG REPORTING.....</b>	<b>84</b>
<b>5.3</b>	<b>FAQ.....</b>	<b>84</b>
<b>6</b>	<b>APPENDIXES.....</b>	<b>87</b>
<b>6.1</b>	<b>KBR ONTOLOGY .....</b>	<b>87</b>
<b>6.2</b>	<b>SEMANTIC SEARCH AND RANKING ALGORITHM.....</b>	<b>89</b>
<b>6.3</b>	<b>SAMPLE REPOSITORY BASED ON MOMOCS TID UC'S.....</b>	<b>92</b>
<b>6.4</b>	<b>ONGOING FEATURES .....</b>	<b>92</b>
<b>7</b>	<b>ANNEXES.....</b>	<b>94</b>
<b>7.1</b>	<b>ACRONYMS AND GLOSSARY .....</b>	<b>94</b>
<b>7.2</b>	<b>REFERENCE DOCUMENTS.....</b>	<b>94</b>

**INDEX OF TABLES**

Table 1 Acronyms and Glossary..... 94

## INDEX OF FIGURES

Figure 1 KB Repository installer. Choosing components .....	18
Figure 2 KB Repository installer. eXist DB Server installation .....	19
Figure 3 KB Repository installer. Sample DB import .....	20
Figure 4 Access the Eclipse Update Manager .....	21
Figure 5 Update manager .....	22
Figure 6 MOMOCS KB Repository update site .....	23
Figure 7 Feature search results .....	24
Figure 8 Feature verification .....	25
Figure 9 Feature installation ended .....	25
Figure 10 Quick Image plugin installation .....	26
Figure 11 Configuration of EMF Compare for XSM types .....	27
Figure 12 KB Repository preference page: eXist DB client .....	28
Figure 13 KB Repository preference page: eXist DB Server .....	29
Figure 14 KB Repository workbench .....	30
Figure 15 KB Repository perspective .....	32
Figure 16 KB Repository explorer view .....	33
Figure 17 KB Repository ontology browser .....	34
Figure 18 KB Repository annotation browser .....	34
Figure 19 Eclipse properties view .....	35
Figure 20 KB Historic explorer .....	35
Figure 21 KB Query results .....	36
Figure 22 KB Attachments view .....	36
Figure 23 KB Notes list view .....	36
Figure 24 KB Repository explorer view .....	38
Figure 25 KB Repository explorer filters .....	40
Figure 26 KB Repository explorer contextual menu .....	42
Figure 27 Eclipse properties view .....	43
Figure 28 KB Ontology browser view .....	45
Figure 29 Change ontology wizard .....	46
Figure 30 KB Semantic annotation view .....	46
Figure 31 KB Historic view .....	47
Figure 32 KB Historic view (II) .....	48
Figure 33 KB Historic View (III) .....	48
Figure 34 KB Historic View (IV) .....	49
Figure 35 Models comparison process .....	50
Figure 36 KB Query results .....	51
Figure 37 KB Query results view contextual menu .....	51
Figure 38 KB Attachments list view .....	52

Figure 39 Add attachment process .....	53
Figure 40 Open Attachment process.....	53
Figure 41 Open Model Diagram attached.....	54
Figure 42 Steps to find out and open a diagram.....	55
Figure 43 KB Notes list content viewer .....	56
Figure 44 KB Repository main menu.....	56
Figure 45 DB Management menu .....	56
Figure 46 KB Artefact management.....	57
Figure 47 KB Repository tools menu .....	57
Figure 48 KB Repository button toolbar.....	58
Figure 49 New repository instance wizard.....	60
Figure 50 New wizard keyword annotation page.....	60
Figure 51 New wizard comments edition page .....	60
Figure 52 KB Explorer view contextual menu.....	62
Figure 53 New folder wizard.....	62
Figure 54 Add Model contextual menu .....	63
Figure 55 New model wizard .....	64
Figure 56 Resource selection wizard .....	64
Figure 57 New note wizard.....	65
Figure 58 New note wizard content page.....	65
Figure 59 Open model contextual menu and workspace container selection wizard.....	67
Figure 60 Model opened within the Momocs Suite workbench.....	68
Figure 61 KB Repository database management toolbar menu.....	69
Figure 62 KB Repository explorer showing database access error message.....	69
Figure 63 KB Repository export wizard .....	70
Figure 64 KB Repository import wizard .....	71
Figure 65 Search button at main toolbar .....	72
Figure 66 KB Repository query tool wizard .....	73
Figure 67 Advanced Search dialog .....	77
Figure 68 Query example .....	80
Figure 69 KB Repository query tool. Semantic search wizard .....	82
Figure 70 Semantic Search results.....	83
Figure 71 KBR Ontology design pattern .....	88

# 1 Scope

This deliverable, but also D5.2 and D5.3, does not fully comply with the organization described in the Description of work, where we were thinking of slicing our tools according to the distinction among data, \*ware, and process. While designing our tools, we understood that that division was unnatural, and would have led to conceivable overlapping among the different tools. Our tools are functionality-specific, but they can easily work on the artifacts at the different levels of the hierarchy identified by the MOMOCS Description of Work.

This is why we opted for a different partitioning based on supplied features, instead of the abstraction level. In other words, we prefer to get rid of the horizontal layers originally proposed, and identify consistent and self-contained vertical segments that easily spread across data, \*ware, and process.

## 1.1 Purpose of the Document

The purpose of the document is offer a complete guide for the XSM KB Repository Tool, which is part of the MOMOCS Tools Suite. This guide helps the users of this tool to install it, to configure it and to start using it. Besides, it offers a detailed reference to all the features provided by the tool.

The development was driven by [2007b] “D41 – XIRUP Supporting Tools Specification”.

This is an updated version of Knowledge Base Repository Tool documentation due at M17 including fundamental improvements and addressing all main issues raised during software validation.

Table 1 below presents an overview of new contributions/updates that were applied to this document along with their related sections: this information reflects the way in which the tool was fine-tuned and enhanced in order to satisfy comments and remarks expressed during the evaluation phase.

Topic	Section
Requirements (updated)	3.1
Installation guide (updated)	3.3
Domain ontology change (new)	4.1.3.1
Models comparison (new)	4.1.5.1.1
Query results view (updated)	4.1.5.2
KB Attachments view. Add attachments (updated)	4.1.5.3
XSM diagram view in graphical format (new)	4.1.5.3.1
Semantic Search (updated)	4.4.2
Know issues (updated)	5.1
KBR Ontology (new)	6.1
Semantic Search and ranking algorithm (new)	6.2

## 1.2 Document Structure

This document is organized as follows:

Section 2 introduces the XSM KB Repository describing its main features in the context of XIRUP Modernisation methodology. Section 3 is a getting started guide that explains the requirements needed to install the tool, how to install the tool and how start playing with it. Section 4 contains the main tool user's guide. Section 5 is a troubleshooting section, listing knows issues, bugs reported and a FAQ. Section 6 contains the appendixes, which describes the semantic search algorithm, additional tool ongoing features under development or forthcoming features to be

implemented. Finally, section 7 contains the annexes with the acronyms, glossary and reference documents.

## 2 KB Repository Introduction

### 2.1 Introduction

XSM Knowledge Base Repository is a container for XIRUP artefacts produced during the different phases of modernization process that converts the legacy system (TBMS) into the modernized system (MS), to promote their reuse whenever is possible. KB Repository will provide support for storing and retrieving those artefacts, which may be required later on by some activities of that modernization process, which can be performed following the XIRUP methodology 2007c. These main features facilitate the re-use of those artefacts in the same modernization process or in others, as it is suggested by the XIRUP.

The repository stores the following artefacts produced during some phases of the XIRUP methodology:

- A XSM created by the Analyst using the XSM Editor.
- A metamodel transformation rules set created by the XIRUP Analyst using the XSM Transformation tool.
- A XSM model transformation historic including a set of transformation mappings.

The XSMs produced during the modernisation process we can store into the KB Repository follow this classification:

By the system target: we can classify models according into the legacy system (TBMS) models or the modernized system (MS) models.

By the level of abstractness: we can classify the models into CIM, PIM or PSM, that is, different views describing the same system at different levels of detail or concerns.

By the scope or extension of the model: we can consider complete models by they own, describing a whole system, or partial models describing only restricted regions

of the system. In the latter case, it is included (among other ones) the “component type models”, describing abstract components, which may be instantiated by particular components in the model, and “pattern models”, a sort of well-known model solution for a particular problem .

All those kind of models are built by the XSM Editor, which can send and retrieve them to/from the XSM KB Repository.

A transformation rules set is a list of mappings between the elements of a source metamodel and those of a target metamodel<sup>1</sup>. A particular metamodel is XIRUP, which is used to describe XSM models. This transformation rules set is used to convert from one source model (as instance of the source metamodel) into the target model (instance of the target metamodel). XSM transformation rules set is created and edited by the XSM Transformation tool.

Finally, the XSM model transformation mappings are a set of links between objects belonging to the source model and those equivalent ones belonging to the target model after being apply to them the transformation rules set. They allow navigating from the objects of the source model to the objects of the target model and vice versa. A set of transformation mappings linking different models coming from an initial one constitute a XSM model transformation historic, also store into the XSM KB Repository. A XSM model transformation mapping (together with the source model, target model and XSM transformation rules set) constitutes a chain link of that transformation historic.

The main operations supported by the XSM KB Repository are the following:

- Manage the repository
- Store artefacts
- Browse the repository
- Search for artefacts
- Retrieve artefacts

---

<sup>1</sup> In general source and target metamodel are not the same. However, in MOMOCS transformations are defined between XIRUP metamodel and itself.

XSM KB Repository is organized according with a hierarchical classification determined by the user. This classification is basically a taxonomical classification consisting of semantically annotated (using metadata and/or ontological concepts) classifiers. In other words, a tree-based structure of semantic annotated folders where are stored the XIRUP artefacts. These folders are semantically annotated by the user, when they are created, assisted by the XSM KB Repository tool, which make use of a set of available domain ontologies. Domain ontologies for MOMOCS case studies are supplied by their domain experts and come with the XSM KB Repository tool bundle. Semantic annotation will be used by the repository to search for artefacts using semantic reasoning.

XSM KB Repository management mainly comprises the organization of the repository hierarchical classification above described.

XIRUP artefacts are stored within the repository and annotated in a similar way than for the repository management.

Repository browsing permits to navigate the repository structure to discover the artefacts stored within. Its main purpose is to manually discover and retrieve XIRUP artefacts.

In a same manner, repository Searching is aimed to discover services that match with a searching criteria determined by the user. XSM KB Repository will support semantic searching by creating semantic queries that are used to reason within the available domain ontologies. The discovered candidate artefacts are also ranked , using semantic algorithms that determine the best artefact matching according with the semantic query criteria and the semantic restrictions established by the domain ontologies, so only best scored candidates are shown to the user. XSM KB Repository will assist the user to determine the searching criteria on the basis of the available domain ontologies, and, according with those criteria, it will automatically build the semantic query.

## 3 Repository Getting Started

### 3.1 Requirements

This section describes the technical requirements required by KB Repository Tool

For installing and running the KB Repository Tool some minimum requirements need to be satisfied:

- Operating system: Windows 2000/XP. This requirement is mandatory in case of installing the tool by using the Windows installer. The KB Repository tool itself it is compatible with any OS compatible with JVM).
- JRE 1.6.
- eXist [[2007d]] database installed and configured, suitable versions of the DB sever are 1.0.1 to 1.1. When installing the tool making use of the Windows “all-in-one” installer, the user can select whether or not let the tool install automatically an eXist DB Server, see section 3.3.1.
- Eclipse Platform. A good starting point is the Eclipse platform version 3.3.1. Distributions are located at <http://www.eclipse.org/downloads>. The Windows “all-in-one” installer executable provides an Eclipse 3.3.1 SDK platform too.
- (Optional) QuickImage Eclipse plugin editor (<http://psnet.nu/eclipse>) to see XSM model diagrams attached to the KB Repository in graphical format (JPG, etc). It can be installed through Eclipse Update Site facility using the following site URL: <http://psnet.nu/eclipse/updates>
- (Optional) EMF Compare Eclipse plugin (<http://www.eclipse.org/modeling/emft/?project=compare#compare>) to show graphically the differences between two XSMs. See section 3.3.2.2 for installation instructions.

All these required and optional requirements are installed into the latest MOMOCS Suite version v1.5, that can be download at the MOMOCS portal, whereby, in case of installing the MOMOCS Suite a full operative KBR tool is available.

## **3.2 KB Repository tool license**

KB Repository is released under the Eclipse Public License.

See the detailed document at <http://www.eclipse.org/legal/epl-v10.html>..

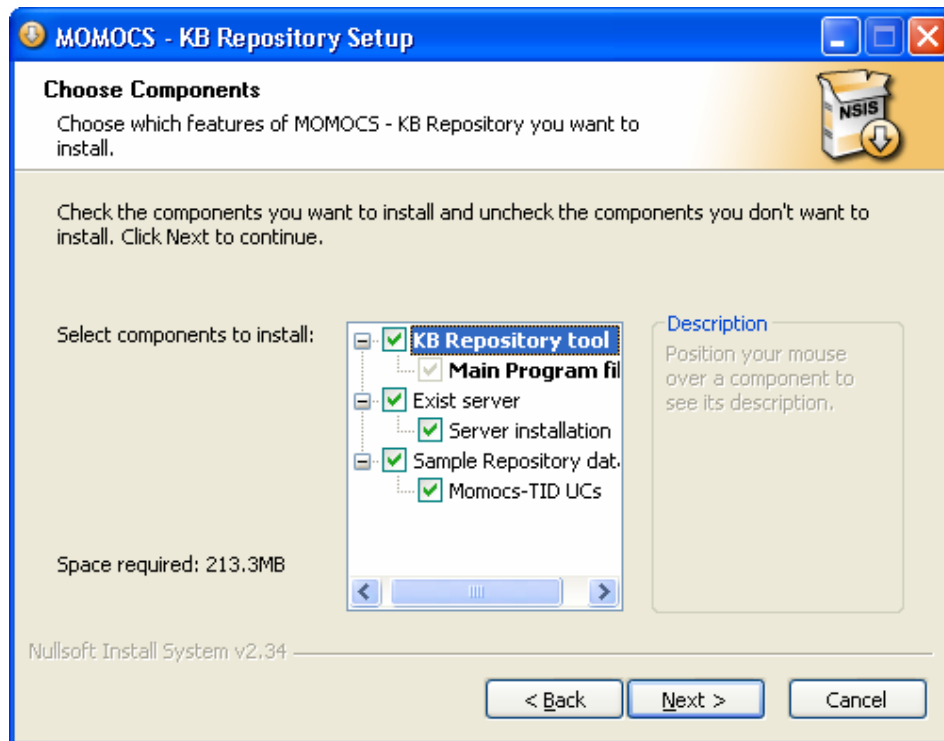
## **3.3 Installation Guide**

KB Repository is released in two flavours: as a Windows installer, packing both KB Repository Client (Eclipse platform + KB Repository plugins) and KB Repository database server (eXist database), and as Eclipse feature, installable via the Eclipse update manager mechanism.

### **3.3.1 Installing the tool making use of the Windows installer**

To start the windows installer, just double-click the executable file named: `kbrepository_v1.0.1.exe`. The installer assistant will be opened.

After the agreement of the KB Repository tool license (aforementioned) the next step consists of selecting which packages the user desires to be installed:



**Figure 1 KB Repository installer. Choosing components**

### **KB Repository tool main files (required)**

This option will install the minimum client requirements for running the KB Repository tool. That is an Eclipse 3.3.1 platform plus the KB Repository tool plugins. Please note that this minimum installation is mandatory.

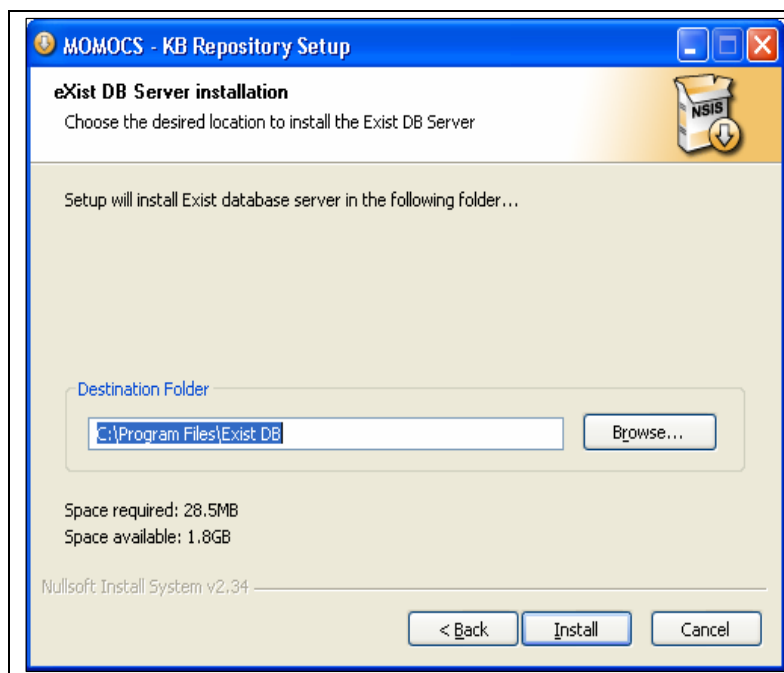
To install this section the user has just to provide to the installer the desired installation path and proceed normally.

### eXist database server (optional)

The KB Repository tools needs an eXist database server to connect to. The installer provides an easy way to install and automatically configure the eXist server to be used within the KB Repository tool.

To proceed the user has just to tell the assistant where to install the eXist server and follow carefully further instructions.

eXist installation takes few minutes to be installed.



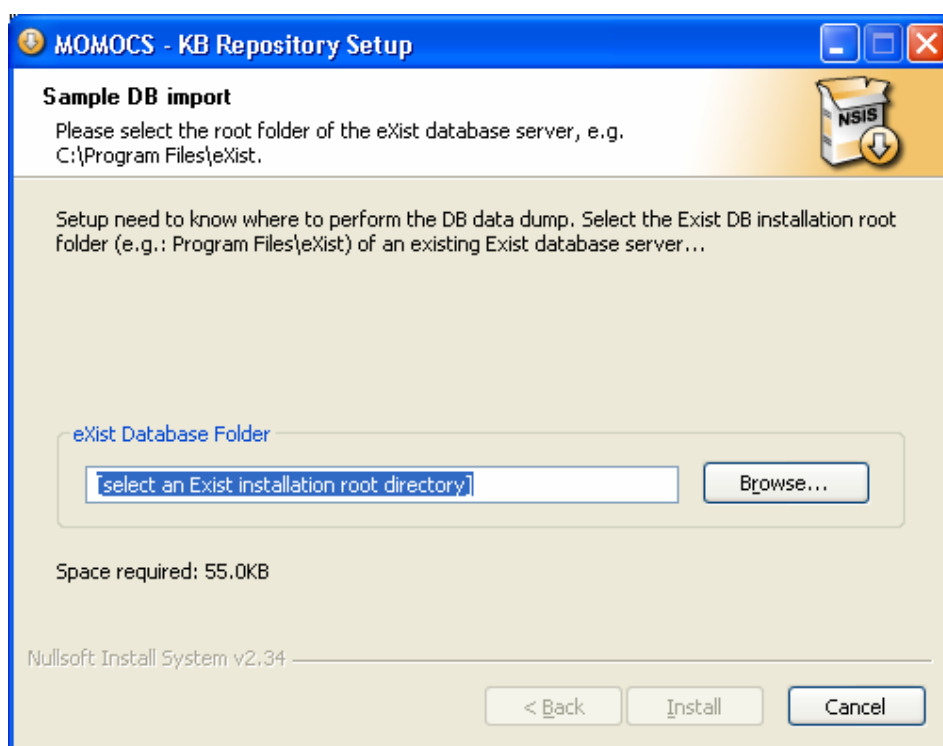
**Figure 2 KB Repository installer. eXist DB Server installation**

Important note: by default, the eXist database server is automatically configured to listen at port 8095.

### Sample repository data (optional)

Once installed the full eXist server package, is also recommended to import some sample data.

To install this package select the root folder of an existing eXist installation by pressing the button “Browse” at the eXist DB Server installation section (see figure below). When installed via this installer, the default path of the eXist server is: C:\Program Files\Exist DB.



**Figure 3 KB Repository installer. Sample DB import**

Follow the instructions given by the installer to successfully install the sample source data into the eXist server.

Note: At this step, the installer will try to start the database, import the sample data and automatically shutdown the server. Remember to, once the KB Repository tool is running, start the database server by using the Repository database management

button: . See section 4.3.1 for more details.

### **Quick Image installation (optional).**

Once installed KB Repository and eXist DB, eclipse can be launched. Then follow instructions in section 3.3.2.1.

### **EMF Compare plugin (optional).**

Install it once KB Repository has been installed. See detailed instructions in section 3.3.2.2.

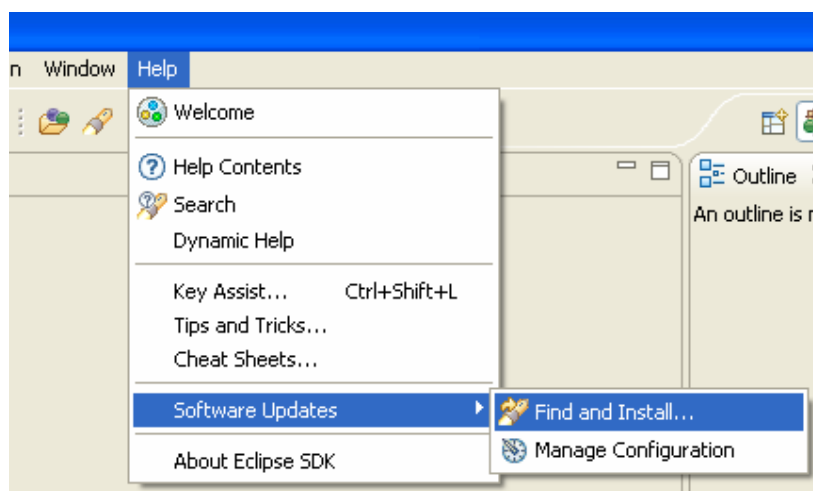
## **3.3.2 Installing the tool via Eclipse Update Manager**

First of all an Eclipse platform (3.3.1 or greater) must be installed on the target computer to be able to install the KB Repository feature (set of plugins) using the Eclipse Update Manager.

### **Feature installation steps:**

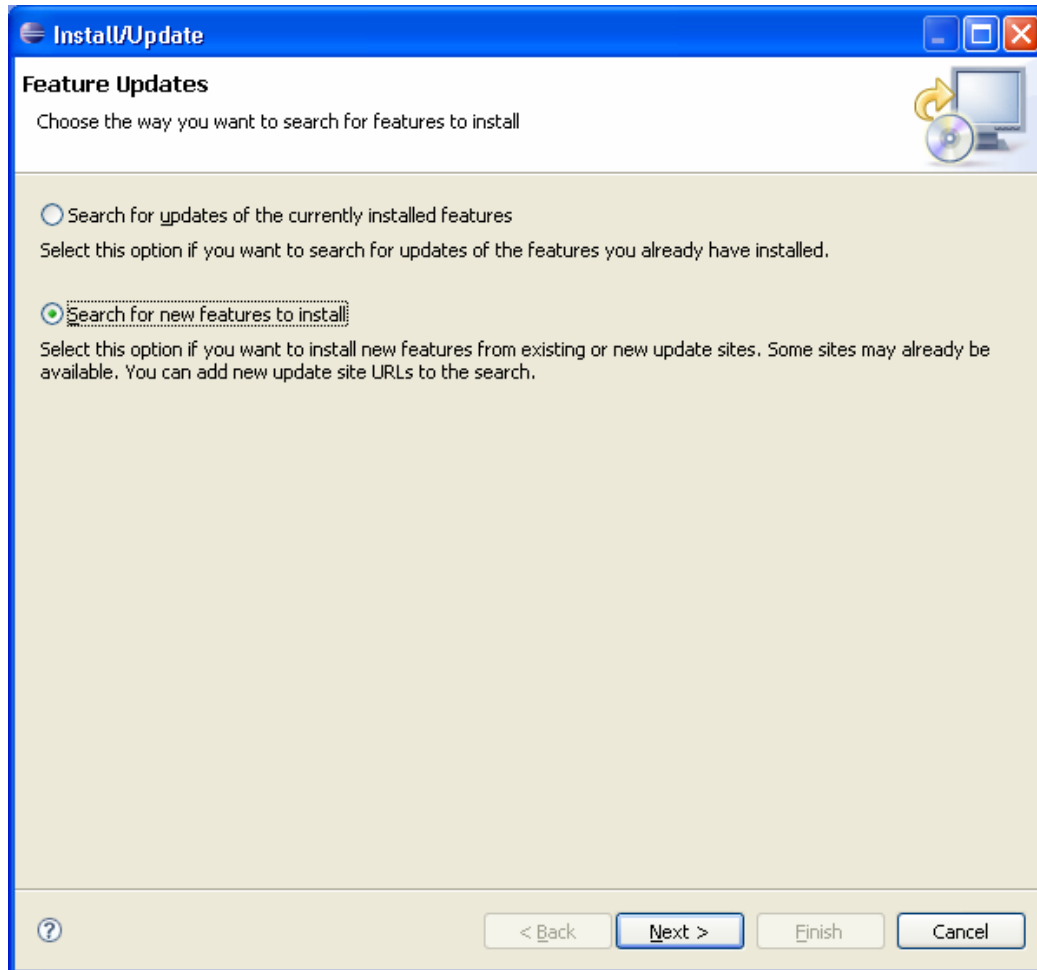
Execute the Eclipse platform and wait until the application has completely been loaded.

Go to menu Help >> Software Updates >> and clic the Find and Install... option. See the figure below



**Figure 4 Access the Eclipse Update Manager**

At Install/Update dialog, select the option Search for new features to install and press Next



**Figure 5 Update manager**

The incoming page, Update sites to visit, shows the configured update sites where to search for updates or new features to install. As far as we have not already defined the MOMOCS KB Repository update lets do it by creating a new Update Site entry. Press the button New Remote site. At the New Update Site dialog set the label for KB Repository tool update site, e.g. "MOMOCS KB Repository" and enter at the URL field the url where the update site is located: [https://services.txt.it/crs\\_subversioning/momocs/WP5/dev/atos/kbrepositorytool](https://services.txt.it/crs_subversioning/momocs/WP5/dev/atos/kbrepositorytool). Press OK, the Update sites to visit dialog page, should appear as in the figure below:

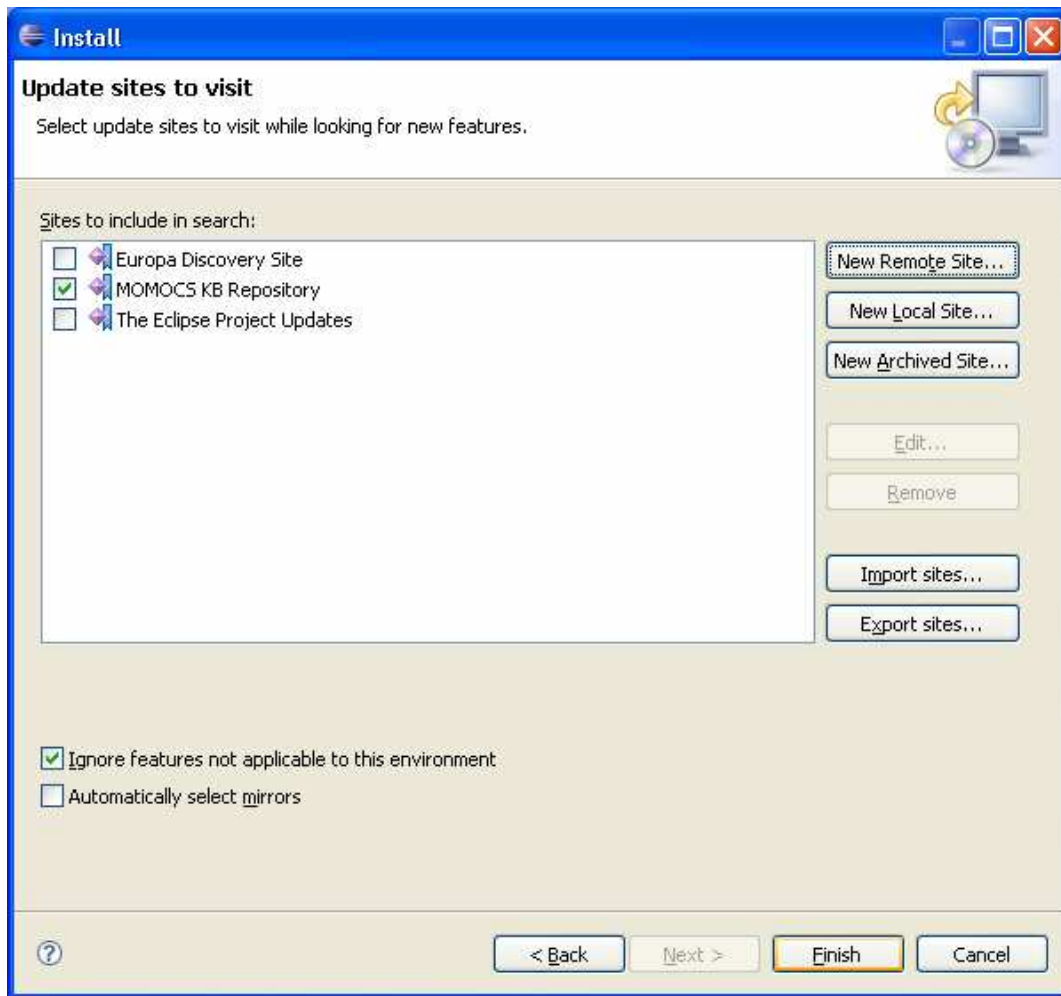
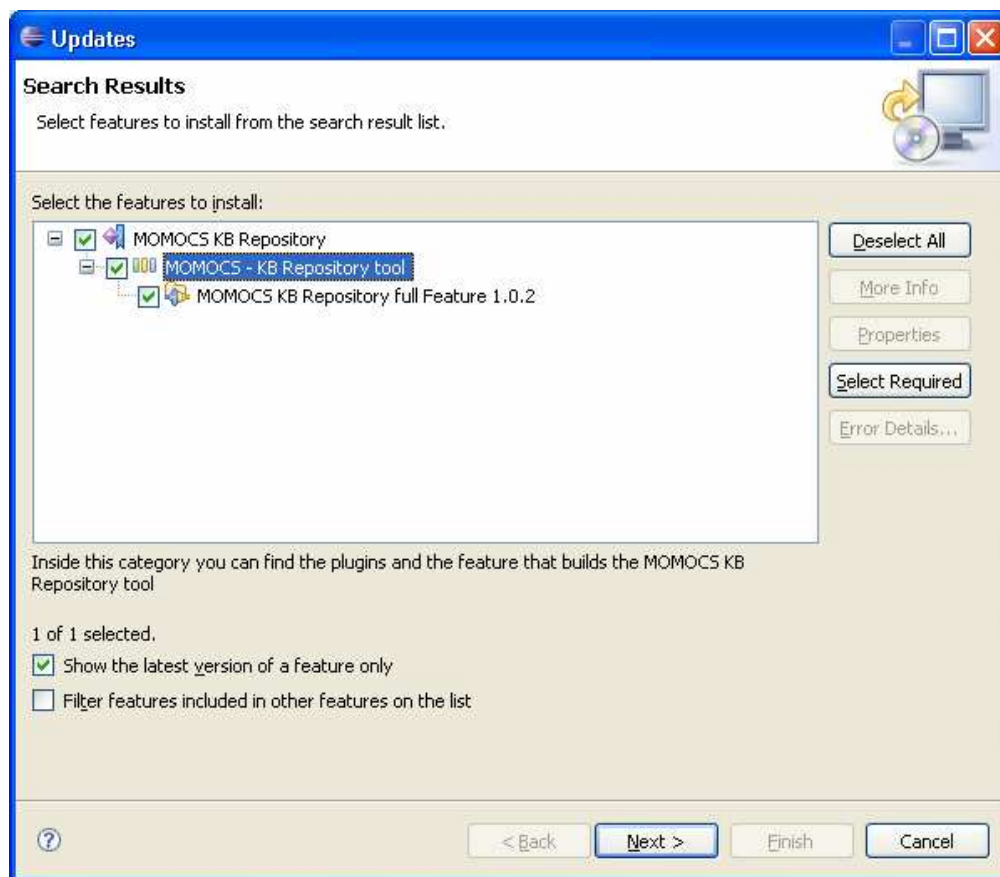


Figure 6 MOMOCS KB Repository update site

Select the MOMOCS KB Repository entry, if not done automatically after adding the new remote site and press the Finish button. Eclipse will contact the host for retrieving the update package information.

After a while the results of the searching activity are shown at Search Results page. Expand the MOMOCS KB Repository update site entry to get the founded features ready to install. Select the feature MOMOCS – KB Repository tool. Do not forget to select the checkbox Show the latest version of the feature only at the bottom of the page. Press the Next button.



**Figure 7 Feature search results**

Accept the terms of the license type and press “Next”. At the next screen, Installation, are displayed the list of features ready to be installed, in this case the MOMOCS KB Repository full feature. Press the Finish button and proceed with the installation.

Wait until the MOMOCS feature is completely downloaded from the update site host (note that this could be a long running operation). Once finished the download process just click at the “Install All” button at the Feature Verification dialog.

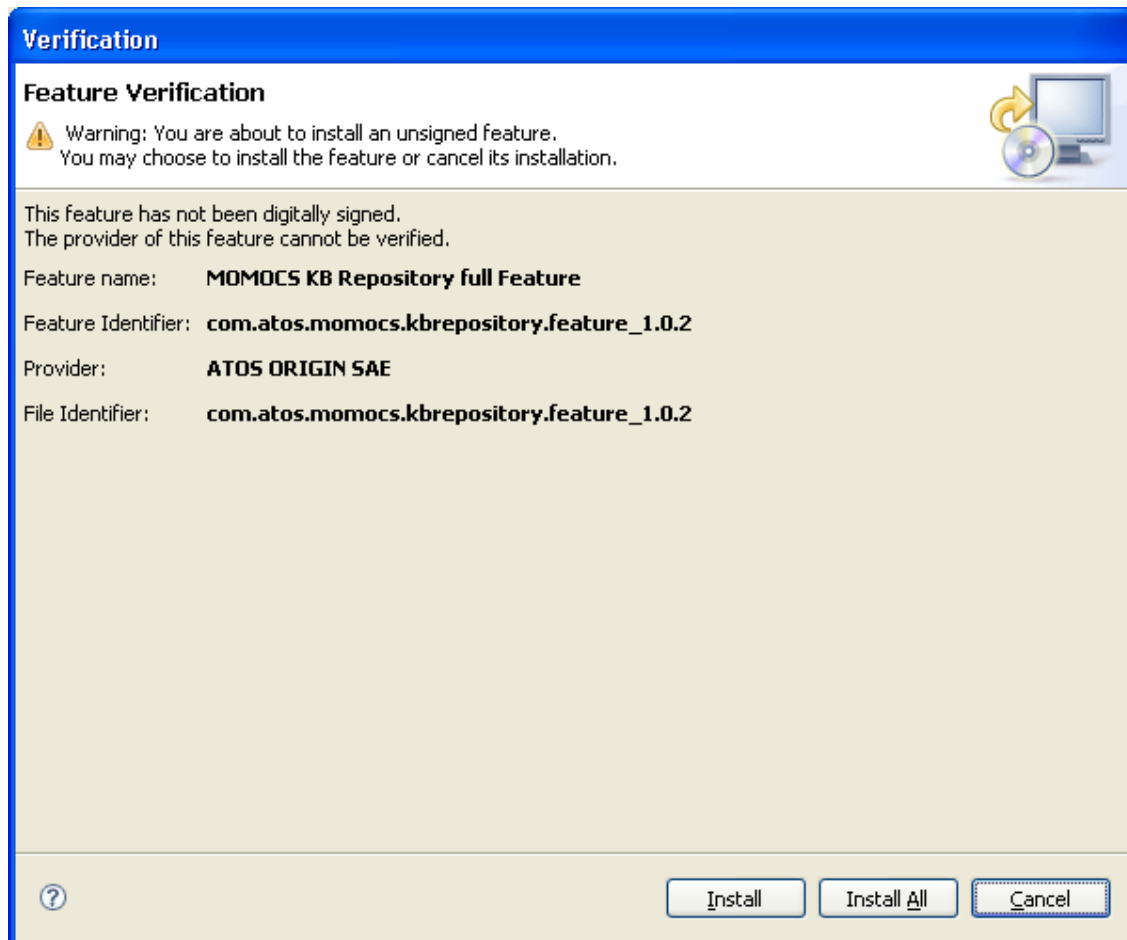


Figure 8 Feature verification

Finish the installation by restarting the Eclipse platform.

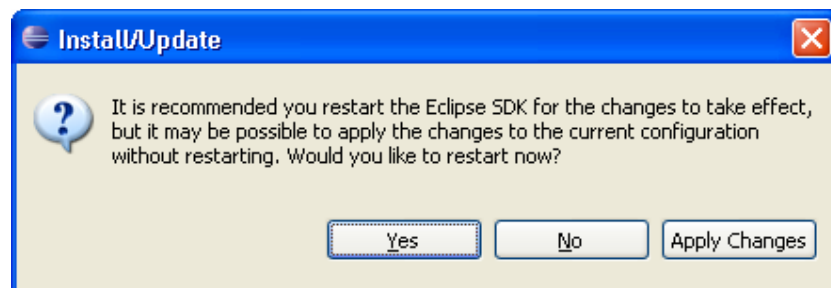


Figure 9 Feature installation ended

Important note: The KB Repository tool does not automatically recognise and configures itself in order to properly work with an existing eXist DB Server instance. For configuring the KB Repository parameters see the explained section at

### 3.3.2.1 QuickImage Editor Installation (Optional).

To install QuickImage Editor, follow the same instructions to install KB Repository through Eclipse Update Manager, but adding the remote site URL for QuickImage Editor: <http://psnet.nu/eclipse/updates>

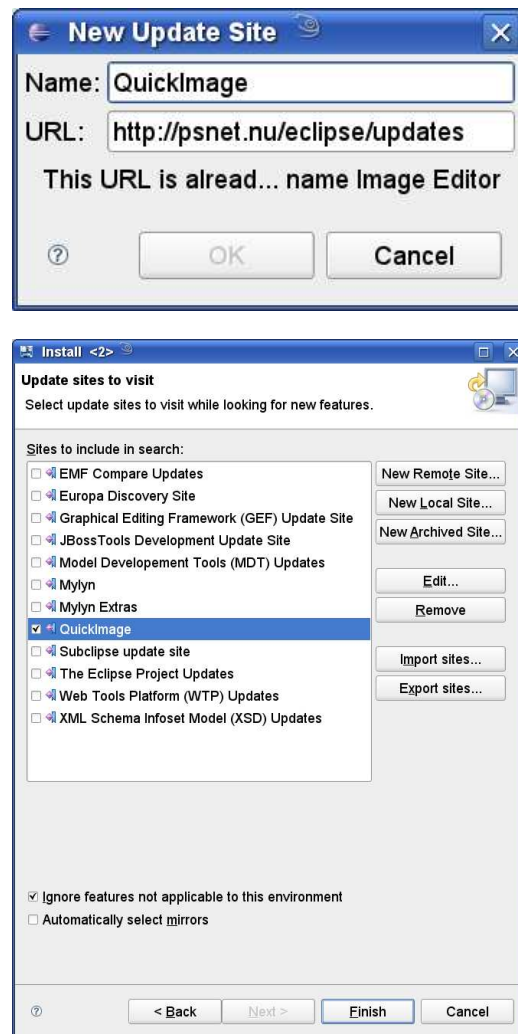


Figure 10 Quick Image plugin installation

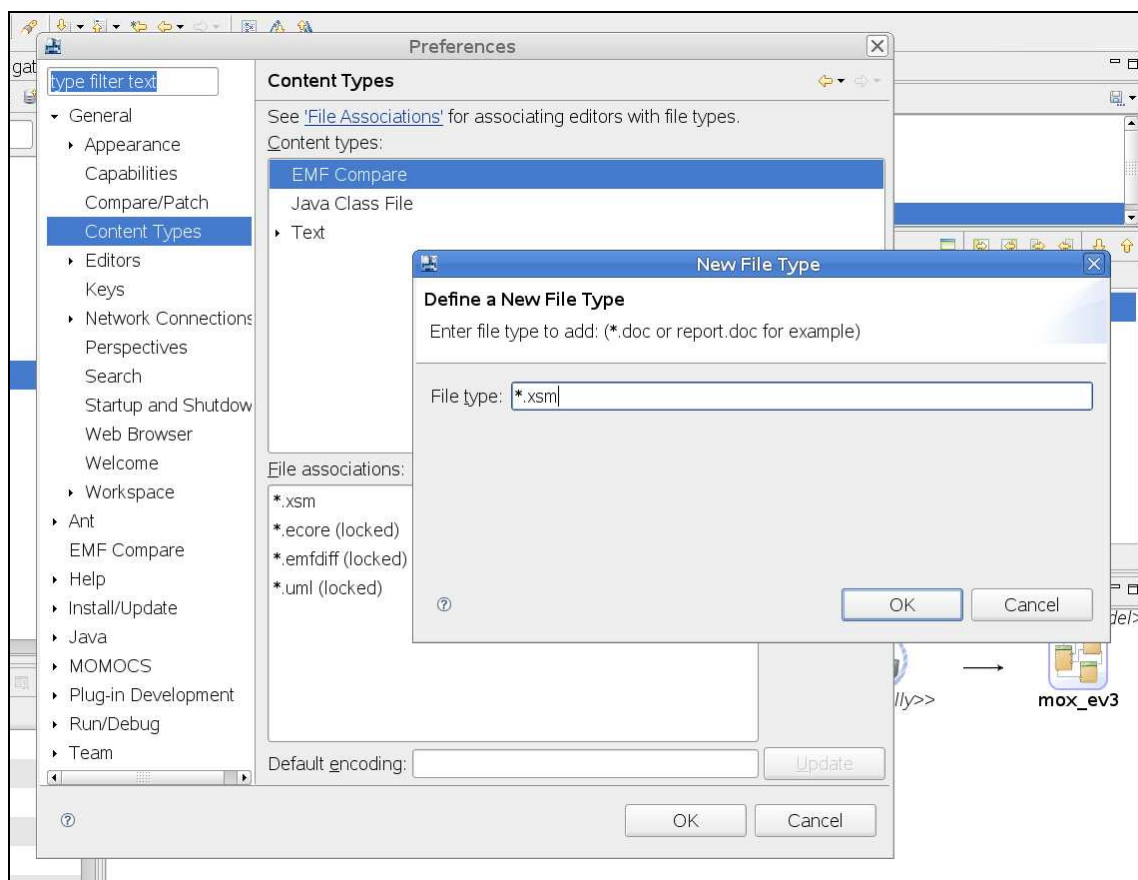
### 3.3.2.2 EMF Compare Plugin (Optional).

To install EMF Compare Plugin, download its release 0.8.0 (18 Jun 2008) at the following link:

<http://www.eclipse.org/modeling/download.php?file=/modeling/emft/compare/downloads/drops/0.8.0/R200806180301/emft-compare-runtime-0.8.0.zip>

Then, unzip that file from the eclipse installation parent directory.

Once installed register XSM extensions for EMF Compare plugin. To do that, access the preference page (Window/Preferences/General/Content Types). Select EMF Compare in the upper panel, click on “Add” button and type “\*.xsm” in the “File Type” text box. See next figure.




**Figure 11 Configuration of EMF Compare for XSM types**

Note: see section 5.1 in case you experience problems to see the EMF Compare View when comparing XSMs.

## 3.4 Getting Started

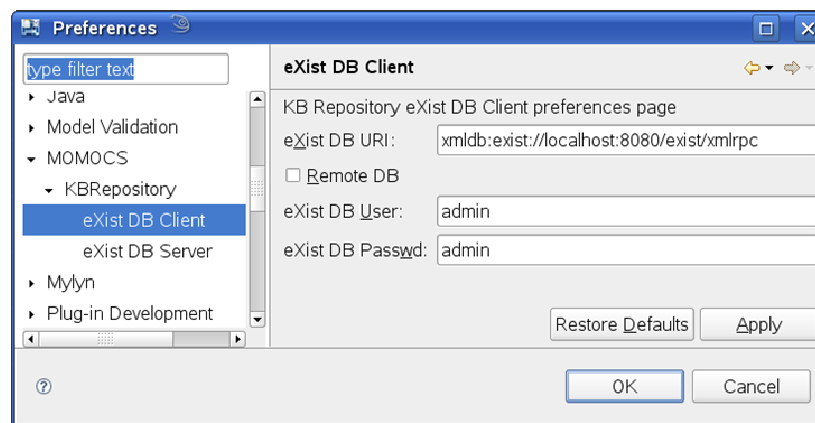
This section explains the initial steps to use KB Repository Tool.

To launch KB Repository Tool go to Start menu >> MOMOCS KB Repository tool >> KB Repository menu option or double click on the kbrepositorytool.exe icon  on the installation directory (if using Windows explorer).

Important note: it is hardly recommended to select the menu option “KB Repository tool (clean)” at the KB Repository Start menu icon group, the first time the KB Repository is started.

### 3.4.1 KB Repository configuration

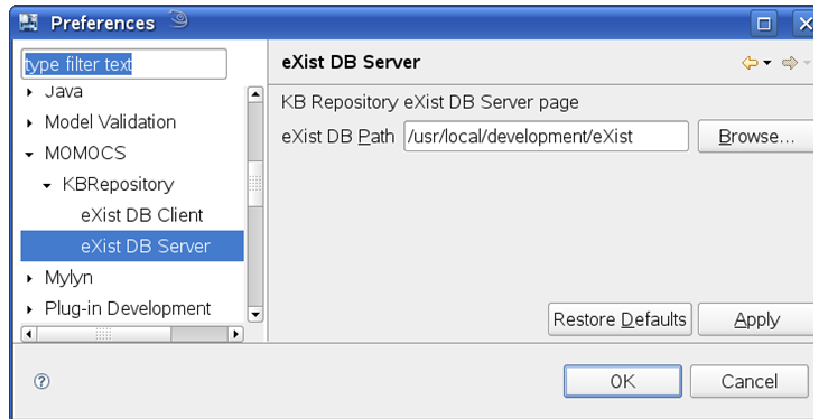
KB Repository Tool is preconfigured by the installation facility, so not additional configuration is required. However, the user can re-configure KB Repository whenever he/she wishes by accessing the Eclipse configuration. Choose Window>Preferences> to open the Preferences dialog. Then choose MOMOCS > KB Repository on the left tree view. Currently, it can be edited the eXist database configuration, both for the server and the client.



**Figure 12 KB Repository preference page: eXist DB client**

eXist DB Client configuration establishes the information required to connect with the KB Repository database: database URI, user and password. In case that KB

Repository is accessing a remote database, the checkbox Remote DB has to be selected. Thus, database management support in KB Repository tool is disabled.




**Figure 13 KB Repository preference page: eXist DB Server**

eXist DB Server configuration establishes the information required to access the local eXist database installation. This allows the KB Repository to manage the startup and the shutdown of the database (only if Remote DB checkbox is unmarked). See section 4.3.1 for repository database management.

### 3.4.2 KB Repository workbench

Within Eclipse workbench, KB Repository tool is accessible by opening the KB Repository perspective.

To open the KB Repository perspective:

- Navigate through KB Repository tool menu >> Views >> Open KB Repository tool perspective.
- Or press button  at the KB toolbar button group.
- Or choose from Eclipse menu: Window > Open perspective > Other, select “KB Repository tool” option.

Once the application engines has been started the perspective is brought to the front opening as shown in the figure below:

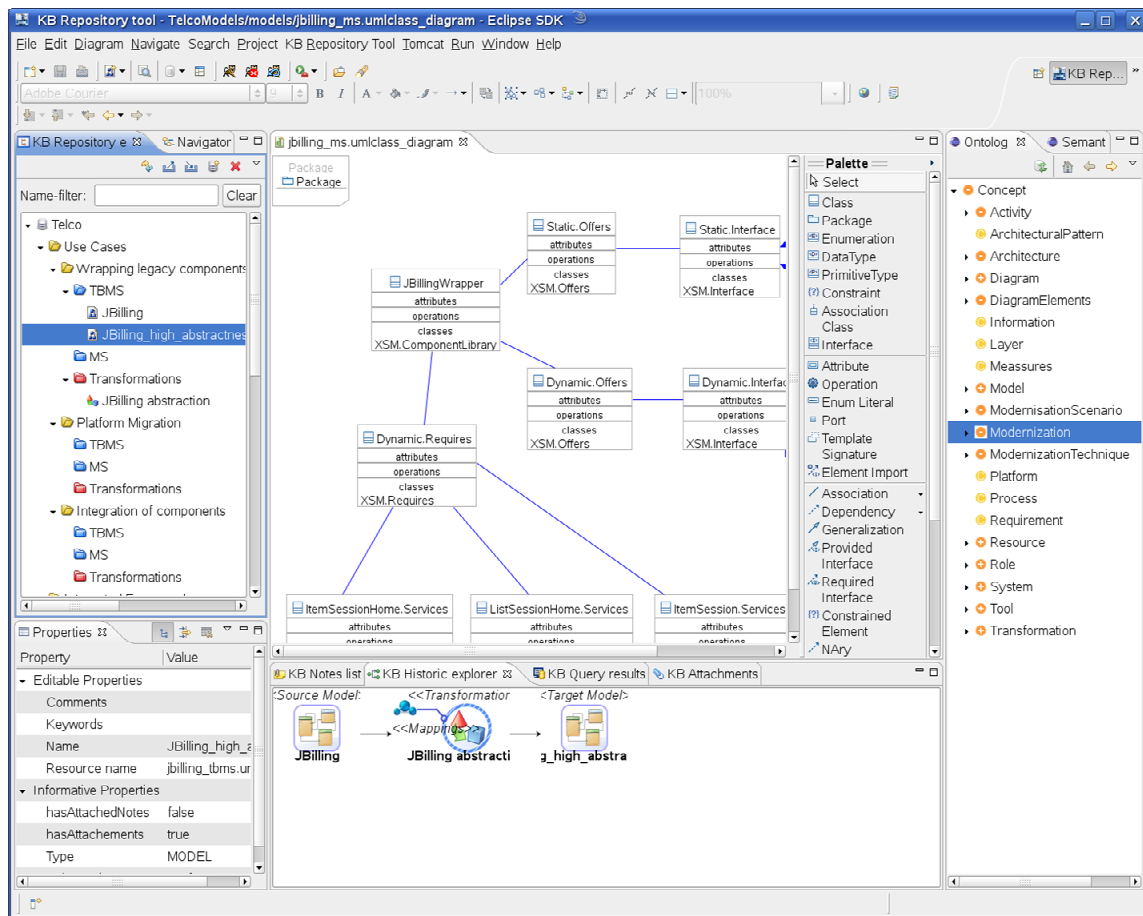


Figure 14 KB Repository workbench

## 4 KB Repository User's Guide

This section describes in detail all the features provided by KB Repository Tool. Besides, it describes the most relevant uses of KB Repository Tool.

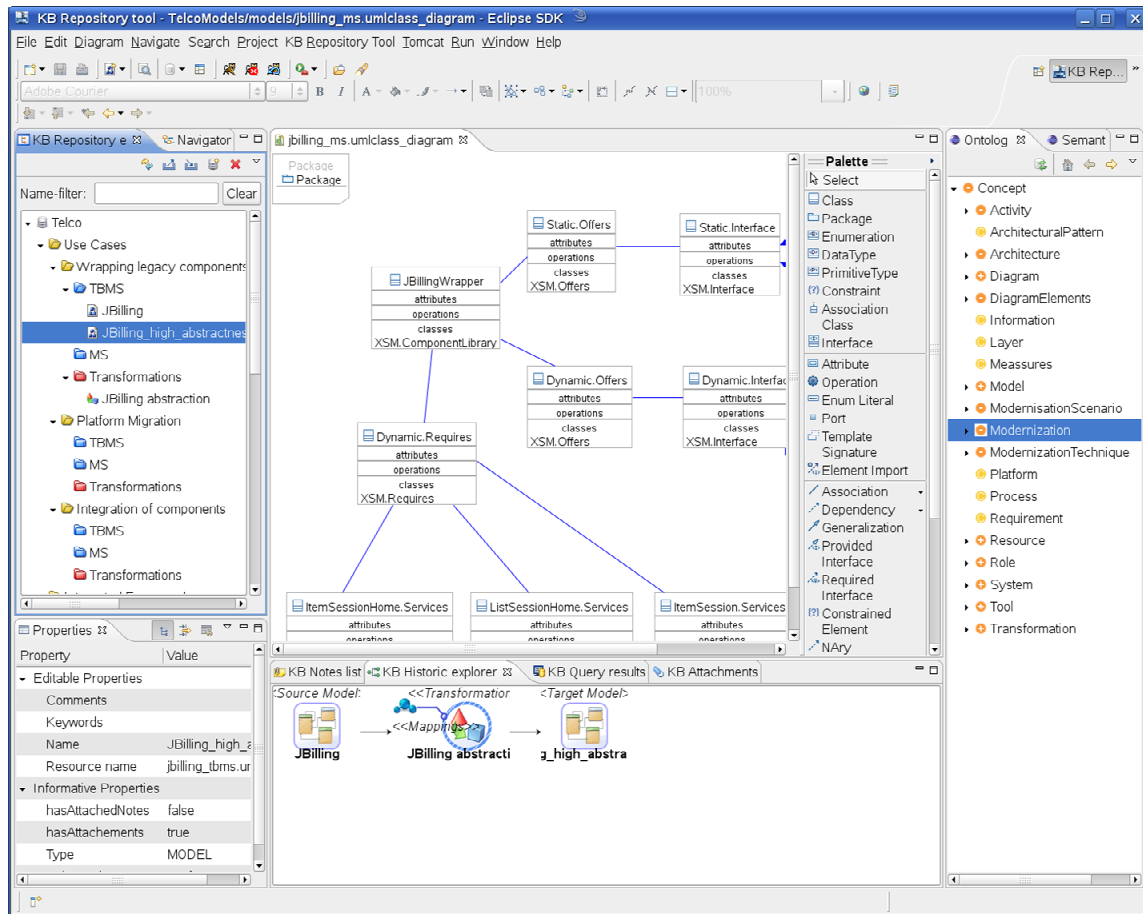
### 4.1 KB Repository Workbench Perspective

The KB Repository offers its functionality via the “KB Repository tool” perspective.

Each workbench perspective is not more than a customized arrangement of windows and buttons in a particular way.


Those windows, formerly named “Views”, are consequently distributed for providing the user a nice and “easy to use” interface to interact effectively with the KB Repository.

To open the KB Repository tool perspective see section 3.4.2: Once the application engines has been started the perspective is brought to the front opening as shown in the figure below:



**Figure 15 KB Repository perspective**

As a normal perspective within the Eclipse platform, the user is able to close the entire perspective, even to reset it (helpful mechanism when some views are hidden accidentally), and hide / show some of the “Views” that compose it.

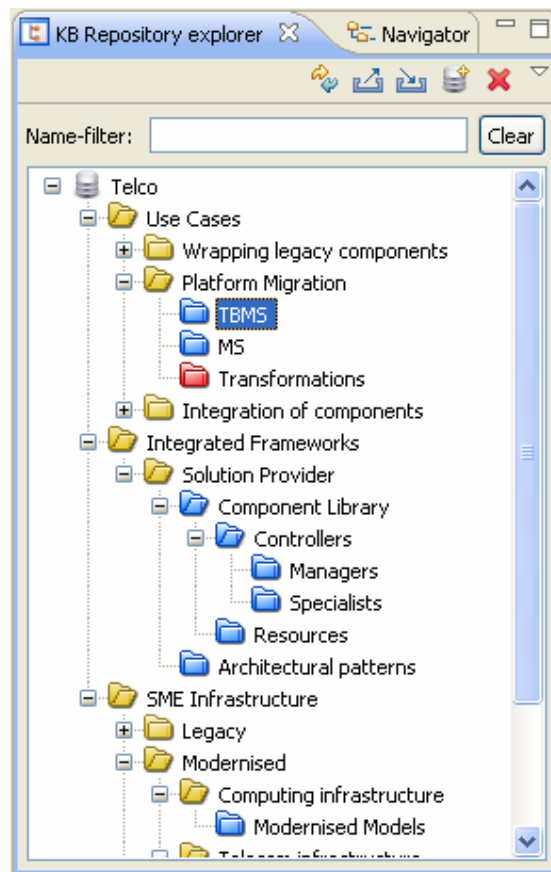
To close the KB Repository tool perspective: Right-click the KB Repository perspective button at perspective toolbar KB  KB Repository... (top-right workbench-window corner) and select the close option.

To reset the KB Repository tool perspective to default configuration: Right-click the KB Repository perspective button at perspective toolbar KB and select the reset option.

The KB Repository tool perspective is made by a set of views, some of them are key views within the tool (managing views), and the others could be considered as auxiliary (informative views).

Management views:

KB Explorer view (for detailed section see 4.1.1)



**Figure 16 KB Repository explorer view**

### KB Repository Ontology Browser (for detailed section see 4.1.3)

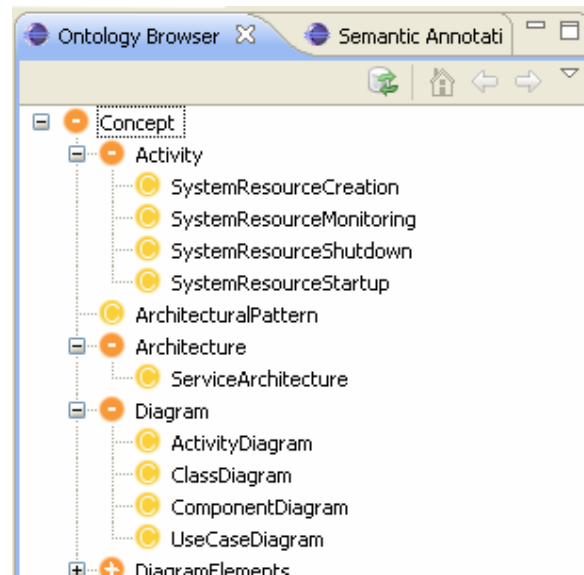


Figure 17 KB Repository ontology browser

### KB Repository semantic annotation browser (for detailed section see 4.1.4)

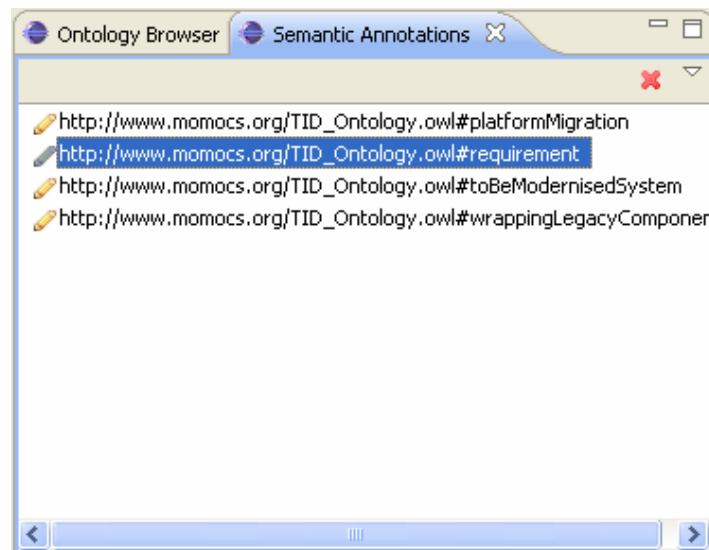


Figure 18 KB Repository annotation browser

Properties editor (for detailed section see 4.1.2)

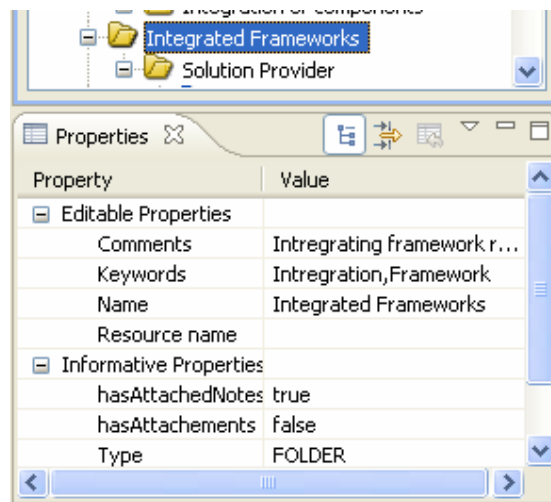


Figure 19 Eclipse properties view

Informative views:

Historic (for detailed at section see 0)

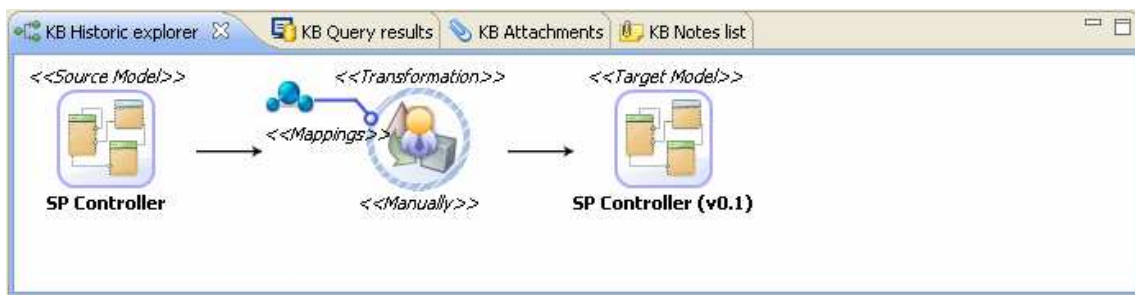
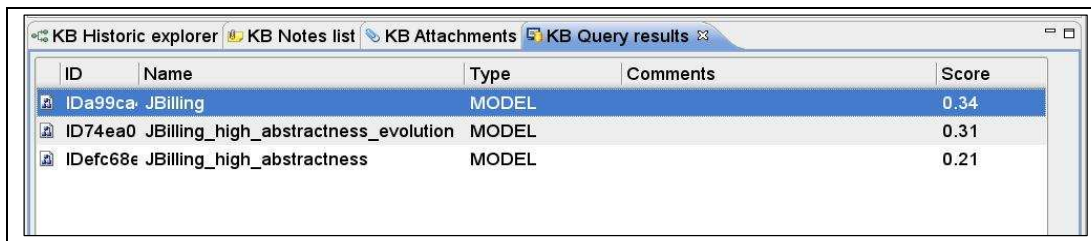


Figure 20 KB Historic explorer

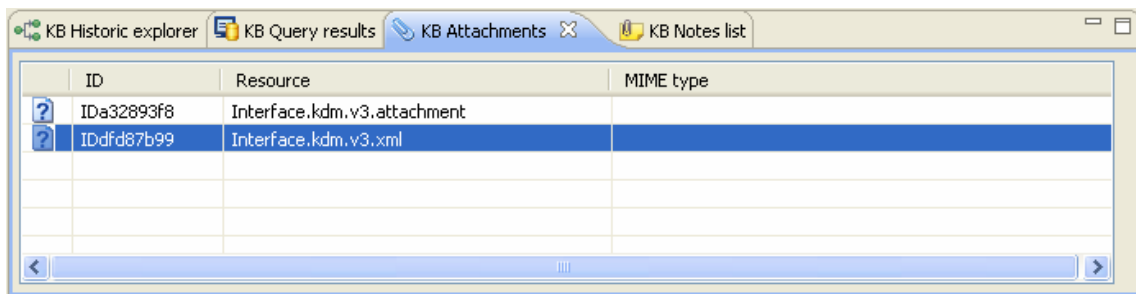
Query results (for detailed at section see 4.1.5.2)



ID	Name	Type	Comments	Score
IDA99ca	JBilling	MODEL		0.34
ID74ea0	JBilling_high_abstractness_evolution	MODEL		0.31
IDefc68e	JBilling_high_abstractness	MODEL		0.21

**Figure 21 KB Query results**

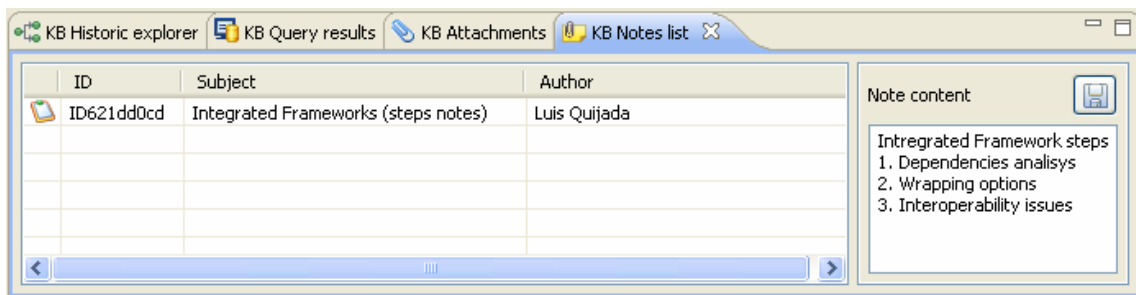
Attachments (for detailed at section see 4.1.5.3)



ID	Resource	MIME type
IDA32893f8	Interface.kdm.v3.attachment	
IDdfd87b99	Interface.kdm.v3.xml	

**Figure 22 KB Attachments view**

Notes (for a detailed at section see 4.1.5.4)



ID	Subject	Author
ID621dd0cd	Integrated Frameworks (steps notes)	Luis Quijada

**Note content**  
Integrated Framework steps  
1. Dependencies analysis  
2. Wrapping options  
3. Interoperability issues

**Figure 23 KB Notes list view**

The tool perspective is divided into 3 different main areas: left, right, bottom where the KB Repository views are docked to. This initial configuration is the default one allowing the user to freely change the perspective arrangement.

KB Repository tool has also a dedicated menu and a toolbar button group located at the upper area of the workbench window.

By default the KB Explorer view (main KB Repository management view) appears at the left side with the common Properties view located at the bottom side in the same panel. KB Ontology Browser and Annotation editor are docked together composing the right panel. The rest of the views (informative views) are located at bottom area of the workbench window.

Figure 15 shows the default perspective configuration.

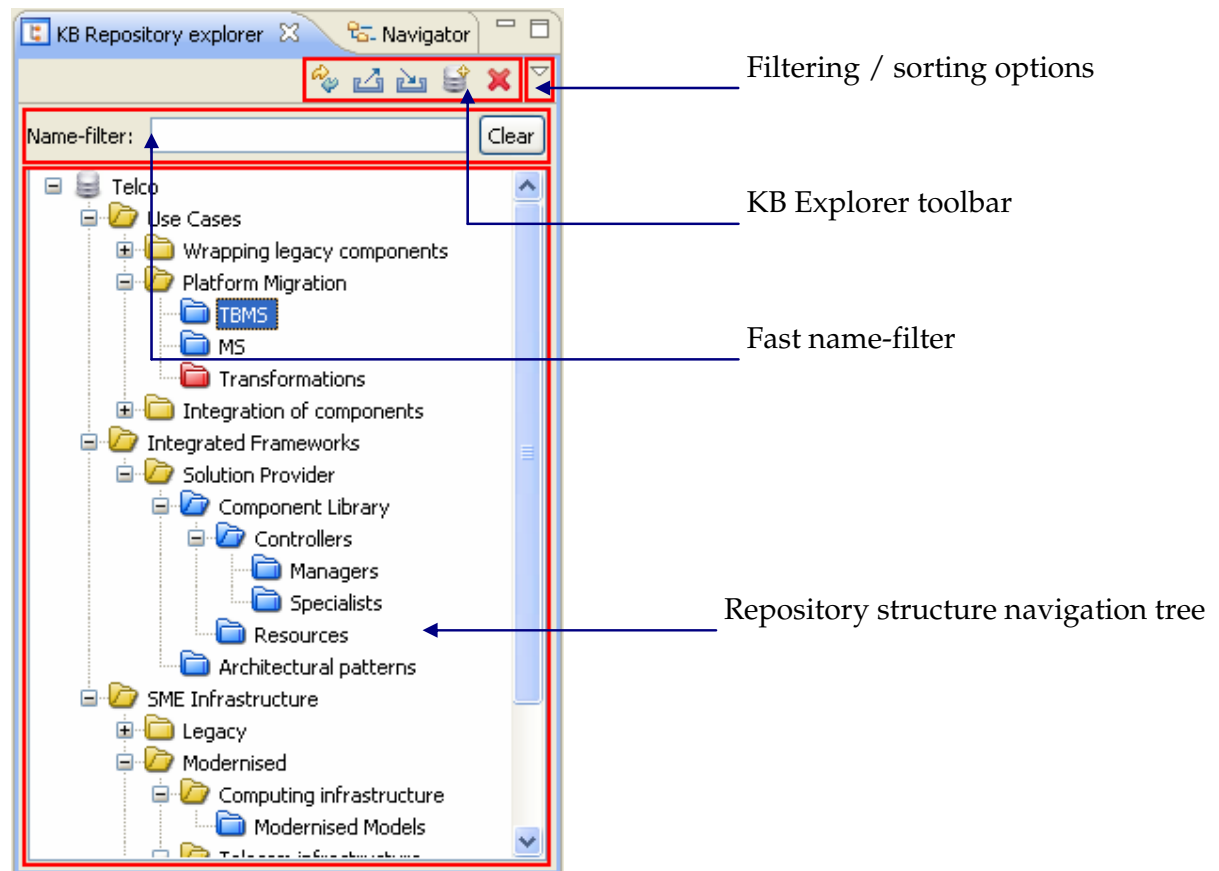
KB Repository perspective can be customised just by dragging and dropping views to another location, resizing them, closing them, open other available views.

Once customized the perspective, the arrangement configuration will be stored when closed and restored the next time the perspective is opened.

#### 4.1.1 KB Repository Explorer View

As written before, KB Explorer is the most important view of the tool allowing us a complete management of the KB Repository structure by creating and removing KB Repository domain objects (see 4.2.1) and modifying the artefact hierarchy.

## KB Repository Explorer layout description



**Figure 24 KB Repository explorer view**





### 4.1.1.1 KB Repository navigation

The repository navigation tree, the core of the KB Explorer view, is a kind of file system virtualised browser. Due to the potential huge amount of KB artefacts to be stored in the repository, the KB Explorer navigation view offers the user filter capabilities, in order to reduce the number of displayed artefacts and sorting options.

Selecting an artefact in the navigations tree produces the informative views to be refreshed showing information related to the selected artefact, like artefact properties, transformation historic (in case of models), possible attachments, notes, and so on.

As a file system navigation tree, there are folders and non-container elements, like model, transformations, notes and so on.

### **Hide / show the container contents**

In the navigation tree, each container has a small handle (/ ). Clicking on the handle when it looks like  causes folder contents to be made visible. Clicking on the handle when it is like  causes all sub-folders to be hidden.

### **Drag and drop**

The Navigation tree allow drag'n'drop of elements for easy modification of the repository structure. There are some restrictions due to the domain model and error message could appear in case of not allowed operations, e.g. when trying to drag a repository-type element.

To move an element into another folder and perform such kinds of operations just left-click on the desired element to move it, holding the mouse left button clicked, drag the element into the desired location and release the mouse left button to drop it on.

#### 4.1.1.2 Ordering and filtering

In a complex repository structure filtering and ordering is a key functionality for fast location of elements. The filtering does not modify at all the internal repository structure, just reduces the displayed elements using some criteria.

##### Built-in filters

KB Repository allows predefined filtering options like: show only models, show only transformations, hide/show notes elements (take into account that notes elements are hidden by default).

To access the predefined filters see Figure 25:

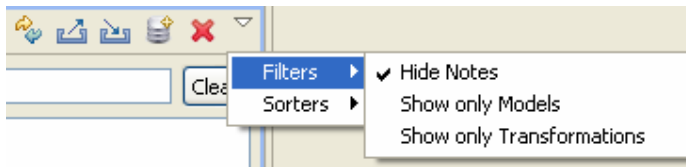


Figure 25 KB Repository explorer filters

##### Dynamic filtering

Dynamic filtering is performed through the “Name filter” command at the top of KB

Explorer view below the main view toolbar

Name-filter:

It allows the user to filter the displayed elements producing a reduced tree structure by inspecting each one of the model elements and matching its name property with the expression given at the “Name-filter” text box.

##### Wildcards and regular expressions

At dynamic filtering, two kind of standard wildcard are allowed: the asterisk (\*) and the question mark (?). As world-wide known wildcard standards, the asterisk (\*) would match any character and the question mark (?) would match 1 or 0 characters.

More in general, the dynamic filtering capabilities evaluate the contents of the text box as a regular expression, allowing the user to build powerful and complex search patterns.

E.g.: user wants to locate some models elements within the repository structure which name begins with the expression "DB". Then the user would write at the name-filter text box: "DB\*". The KB Repository explorer would refresh its data model in order to show all the elements (and their respective parents) which name starts with "DB".

#### 4.1.1.3 KB Repository explorer view toolbar

Used to perform some specific explorer view related actions, it is located at the upper side of the KB Explorer view, appearing as shown below:



Refresh button: forces de KB Explorer to refresh its contents, re-building the repository tree structure



Export-tool: opens the Export-tool wizard (see detailed description at section 4.3.2)



Import-tool: opens the Import-tool wizard (see detailed description at section 4.3.2)



New repository-type element: opens the New Repository wizard (see detailed description at section 4.2.2).



Delete selected artefact/s (see detailed description at section 4.2.6).



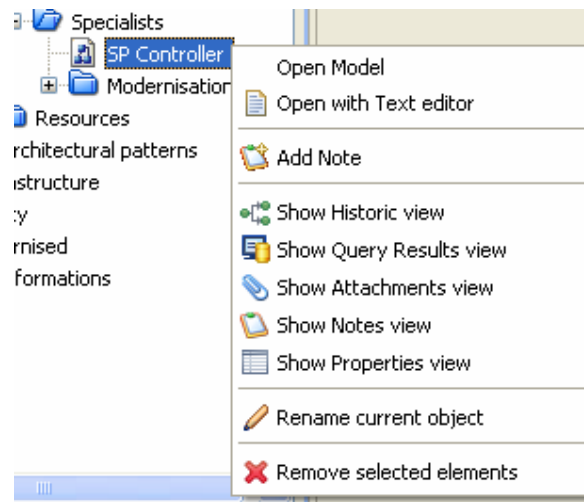
Filtering / ordering sub-menus

#### 4.1.1.4 KB Repository navigation tree contextual menu

To rapid access the allowed operations that could be applied to a KB Repository element, just select the desired element/s and right click.

The contextual menu appears showing the main actions related to the selected artefact.

See an example below:

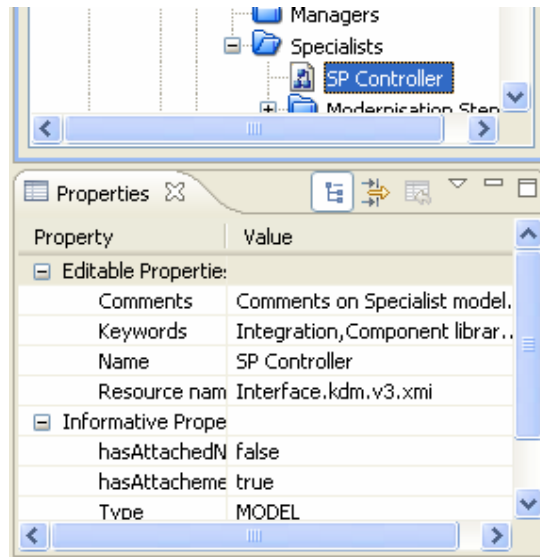


**Figure 26 KB Repository explorer contextual menu**

View the detailed command descriptions at the administration guide (see detailed description at section 4.2).

#### 4.1.2 KB Repository artefacts properties editor

The Properties view is a common view in the form of a table widely used along most of the data manipulation applications. It is shown in the picture below.



**Figure 27 Eclipse properties view**

This view (as the informative views are) is selection context sensible. Selecting a KB Repository artefact using the KB Explorer or any of the informative views causes the properties table to be refreshed displaying the predefined artefact properties and its current values.

There are two kinds of properties, or categories:


Informative: read-only properties initialized during artefact creation (e.g. "Unique Id.") or indicating some states of the object (e.g. "has Attachments").

Editable: those of them which can be modified at any time, like "Name" or "Keywords".

To edit an editable property:


1. Select the KB Repository artefact
2. Click on the actual value of the desired property

If the property is a string-value property, just write the new value and press the <<INTRO>> key or remove the focus off the field.

If it is a file system resource property, then click on the button that appears at the end of the property value field Resource nameInterface.kdm.v3.xmi  and a special resource selection dialog opens.

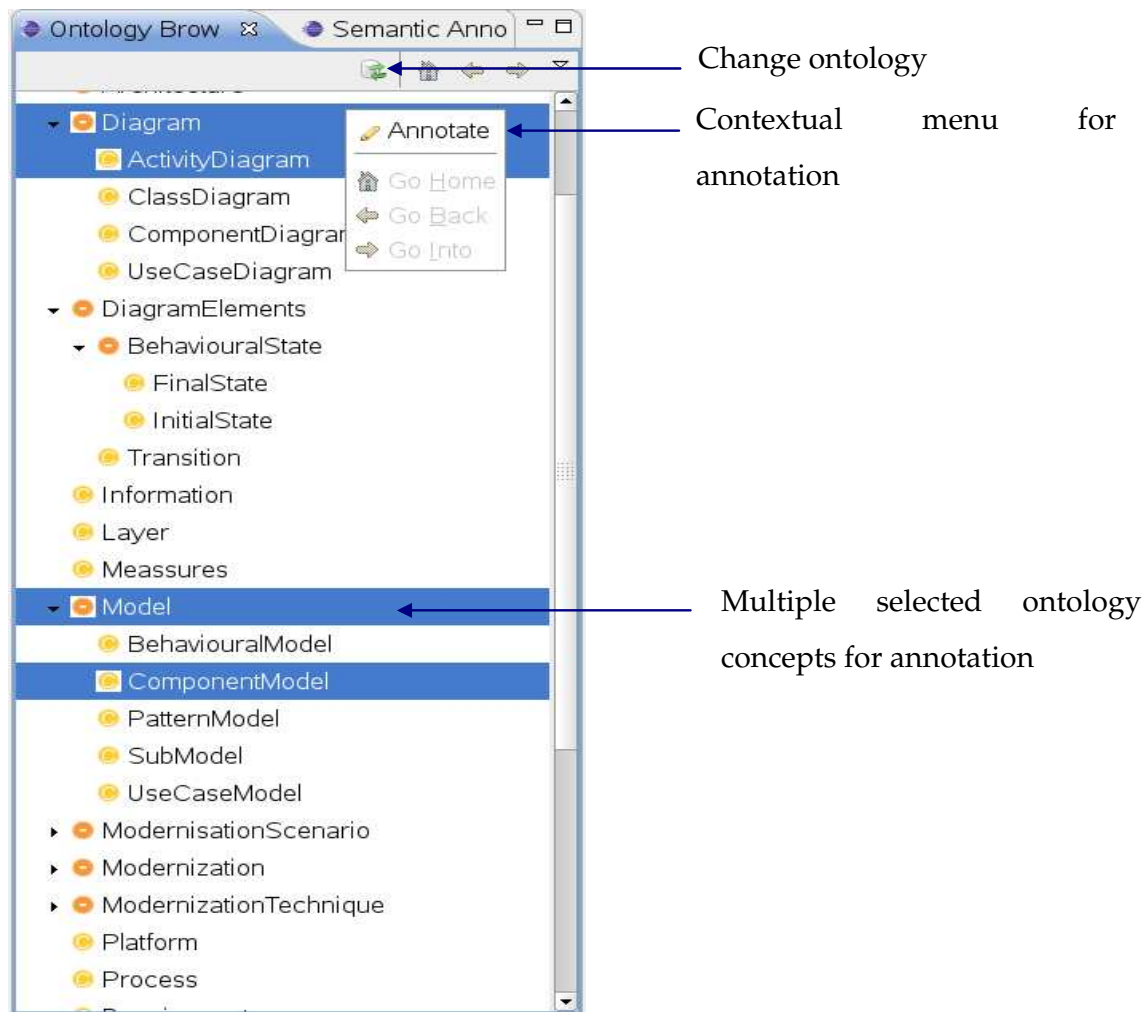
Switch to normal / categorized view:

The properties can be displayed divided into categories or ordered alphabetically.

To switch between these different modes click the button  located at the properties view toolbar.

### 4.1.3 KB Repository Ontology Browser


Ontology Browser view is a graphical browser that shows a selected ontology to be used to annotated the KB Repository artefacts selected in the KB Explorer view with semantic concepts. The reason to annotate repository artefacts with semantic metadata is to support intelligent search (see section 4.4.2). Those KB Repository artefacts can be repositories, folders, models, transformations.



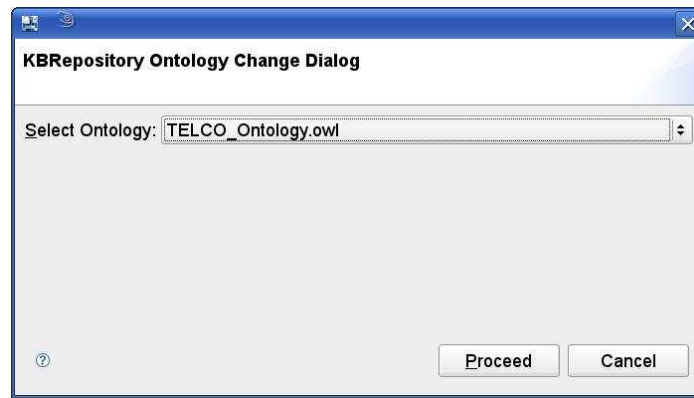
**Figure 28 KB Ontology browser view**

The user can navigate the ontology, browsing concepts and selected those that better describe the artefact selected within the repository in KB Explorer View. Then, right clicking a contextual menu appears with an option to annotate the artefact with the semantic concepts selected. The annotations of each repository artefact are shown in the semantic annotation view (section 4.1.4).

#### **4.1.3.1 Ontology change.**

The ontology shown in this view and used by the KBR to annotate and search artefacts can be changed by clicking in this view toolbar button. .

<sup>2</sup> This feature is not available in current version, but it will be implemented in next



**Figure 29 Change ontology wizard**

This button permits to change the current ontology and knowledge base used to annotate and search artefacts within the KBR. Therefore it is important to activate always the appropriate ontology, especially to perform correct semantic searches (see section 4.4.2)

#### 4.1.4 KB Repository Semantic Annotation View

Semantic annotation view shows the semantic annotations attached to the repository artefacts. The annotations shown in this view are refreshed every time another artefact is selected within the KB Explorer view or new annotations are attached by the user for that artefact. An annotation entry is a qualified ontology concept, that is, a ontology namespace prefix plus the concept name: <ontology Namespace>#<concept name>



**Figure 30 KB Semantic annotation view**

### 4.1.5 Informative / auxiliary views

KB Repository perspective owns a set of auxiliary views reflecting additional information depending on the current selected artefact.

#### 4.1.5.1 Historic view

This is the most important view for the XIRUP modernisation process and model transformation context. The “Historic view” shows graphically the model evolution relationships, transformation paths and branches, offering an easy way to visualize the particular evolution steps of the selected model artefact.

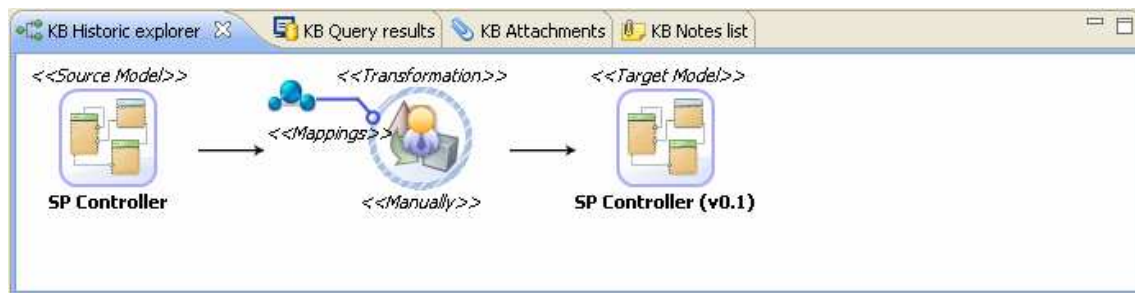


Figure 31 KB Historic view

To open a model or transformation artefact

Right click over the model icon and click the Open option at the contextual menu.



To open the transformation artefact associated to the evolution process, right click on its image icon and select the “Open” option.



The user must take into account that model transformations may occur without using a particular transformation artefact, that is: a manual transformation. In this case, the icon image of the transformation is decorated with different motives.



A special greyed effect applies to the artefacts image icon when they are deleted from the KB Repository tool. In other words, the model transformation information

persists in spite of the state (deleted or not) of the rest of the components conforming the evolution process.

### Navigating through the model evolution process

In a transformation process from a To Be Modernised System (TBMS) to a Modernised System (MS) a vast number of models may result. The system modernisation may make use of a large number of iterations to finally obtain a successfully modernised models package.

Tracking each model evolution is a key functionality in order to keep track of the changes that took place during each step of the process.

Using the graphical capabilities provided by this view it is easy for the Analyst to locate the model transformation history path. To illustrate this useful functionality let's imagine that we have a To Be Modernised model named "TBMS (1)", after applying some kind of transformation results a new model (as an evolution from the previous one) named "TBMS (2)".

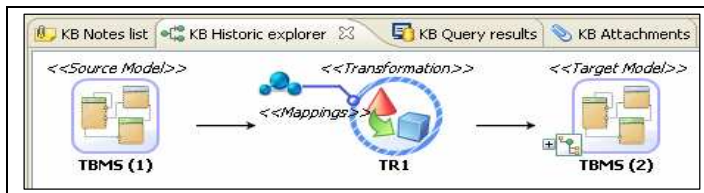


Figure 32 KB Historic view (II)

After inspecting its features, model "TBMS (2)" is considered a well-formed model and approved to be further used along the modernisation process. After that the XIRUP Analyst proceeds with the transformation of "TBMS (2)" model into a new "MS (1)" XSM modernised model.

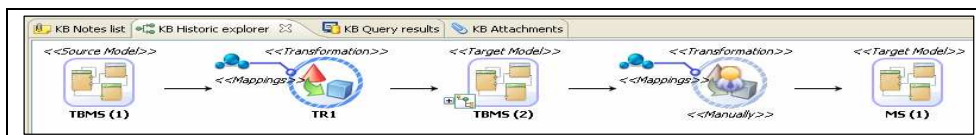
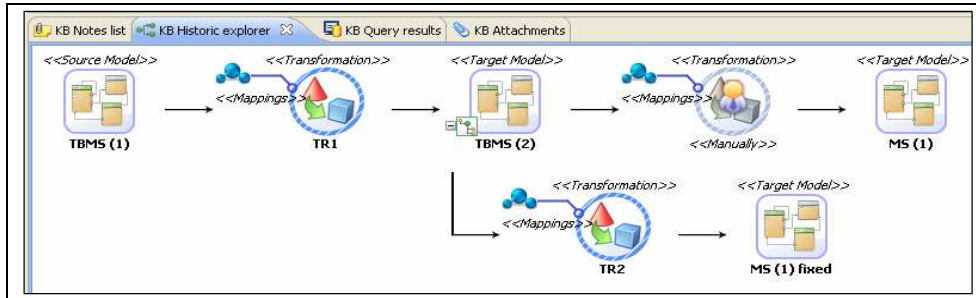


Figure 33 KB Historic View (III)

After performing some testing over the new model it is determined that is not a valid modernised one due to some constraint violations. The Analyst, using the KB

Historic view, is able to easy locate graphically the “TBMS (2)” and restart, at the desired point, the modernisation process again by creating a new branch.



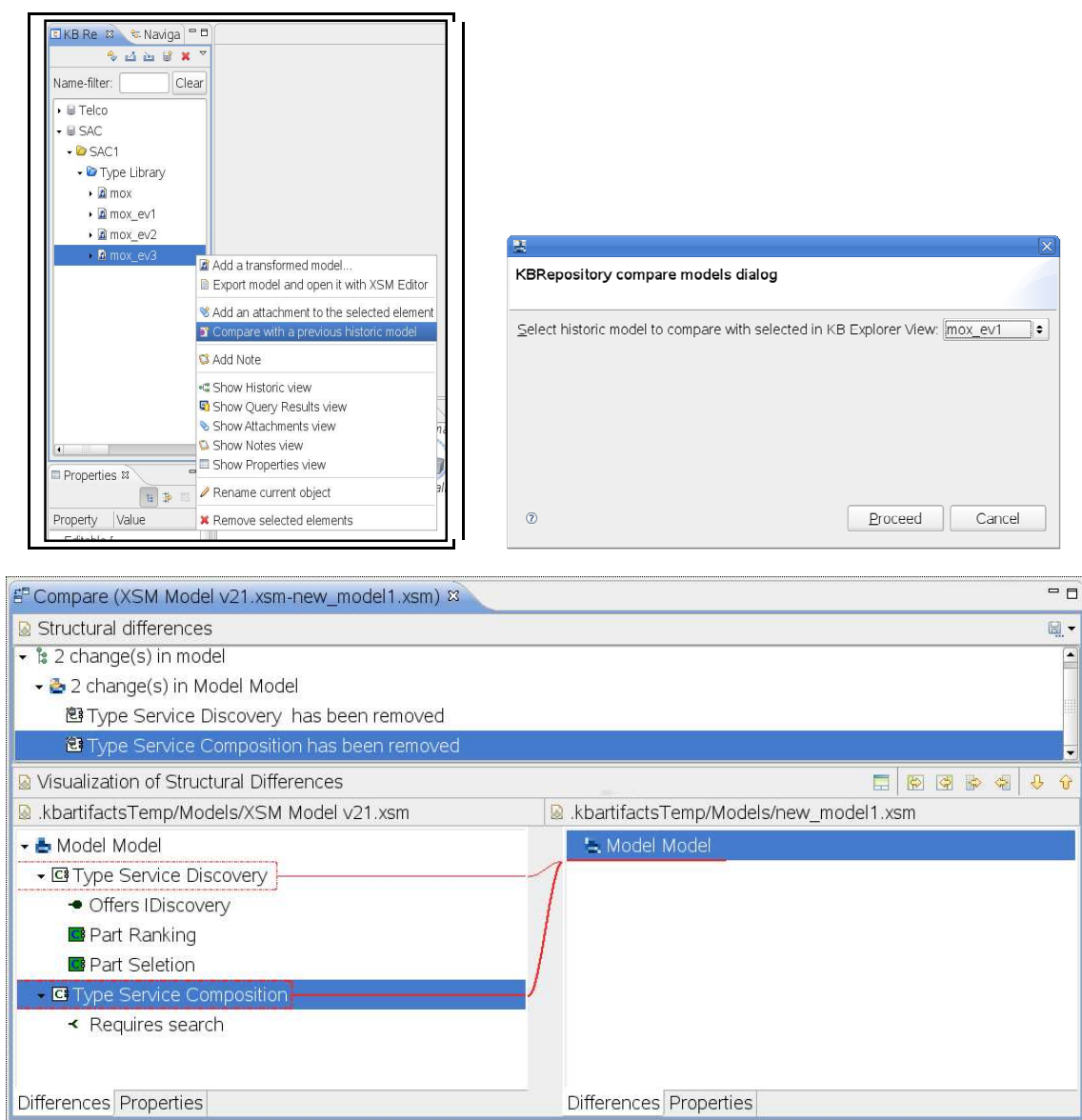
**Figure 34 KB Historic View (IV)**

#### 4.1.5.1.1 Models comparison

KBR includes a new feature, requested by reviewers and users, to graphically show the differences between two XSMs stored within the repository. This graphical model comparison shows the differences between two models that belongs to the same historic, provided that it has been consider only the analysis of the differences between evolutionary models. This feature leverages on the Eclipse compare API and the EMF Compare API, since XSMs are EMF Ecore models.

The procedure to compare two evolutionary XSMs of the same historic is described as follows (see Figure 35):

Select in the KB Explorer the XSM that will be compared with a previous version of its historic and right-click it to select the contextual menu entry “Compare with a previous historic model”. In the next dialog, unfold the combo box, select the previous historic model to compare with and accept. A new compare view appears showing the differences between both models.



**Figure 35 Models comparison process**

#### 4.1.5.2 Query results view

As a result of a searching action via the KB Repository search engine the Query result view shows the returned items satisfying the desired criteria ordered by matching score.

The results appear in the form of table where the most common properties of the KB artefacts are displayed. See the figure below.

ID	Name	Type	Comments	Score
IDA99ca	JBilling	MODEL		0.34
ID74ea0	JBilling_high_abstractness_evolution	MODEL		0.31
IDefc68e	JBilling_high_abstractness	MODEL		0.21

**Figure 36 KB Query results**

Selecting an element from the list causes the workbench to focus the selected object and reflect its properties at the Properties view.

Right-clicking on one of the items at the results list, it is displayed the contextual menu giving the user access to the common actions associated to each KB artefact.

ID	Name	Type	Comment
ID585bf676	JBilling	FOLDER FOR MODELS	
ID41ed82cf	jbilling_tbms		

Open Model  
 Open with Text editor  
 Remove selected elements

**Figure 37 KB Query results view contextual menu**

To order the result list:

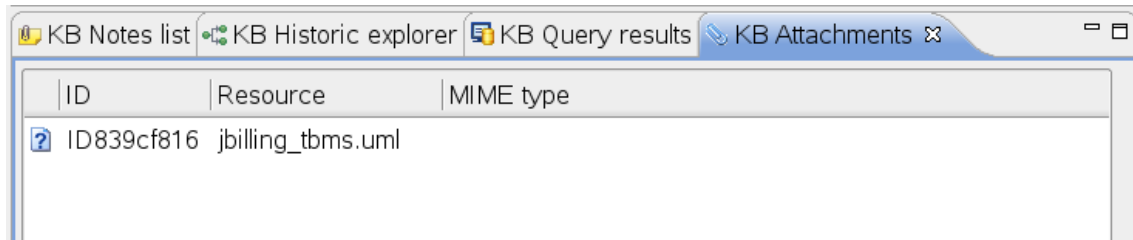
The results can be ordered by Name or by Type by clicking the respective column header.

By default, the result are presented to the user ordered by matching score, the most accurate result is displayed first.

#### 4.1.5.3 KB Attachments view

Attachments are automatically added to models or transformations when they are added to the repository tree view. These attachments are those additional files found within the same artefact source file directory sharing the same name but a different extension. For instance, this is a requirement of some UML editors which require one file for the model and another for the layout, therefore the repository should add both. Future releases of KB Repository tool will also allow attaching any file to those models or transformations added into the repository, which may complement

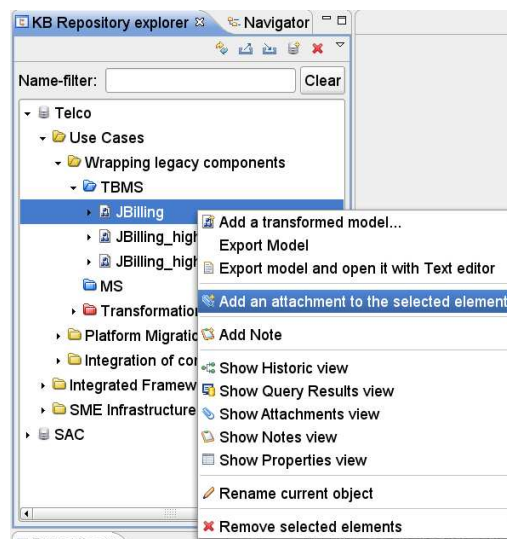
them. Attachments for a particular repository artefact are shown within the KB Attachments view.



**Figure 38 KB Attachments list view**

Besides, after adding models and transformations into the repository, new attachments can be added and linked to a particular artefact (model or transformation). To do that, right click on a particular artefact within the KBExplorer view and select “Add an attachment ...” menu entry. Select the attachment file from the local workspace and accept.

The new added attachment for an artefact is shown in the KB Attachments list view. To export an attachment from the KB repository into the local workspace and open it, right click on the attachment within that view and select “Open with registered editor”. The process is similar to export other artefacts into the local workspace.



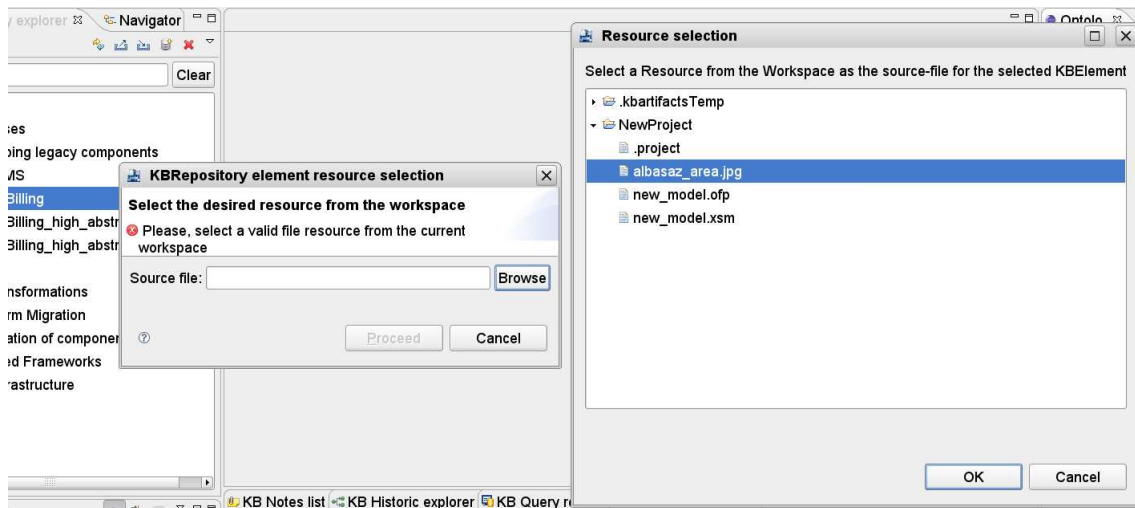


Figure 39 Add attachment process

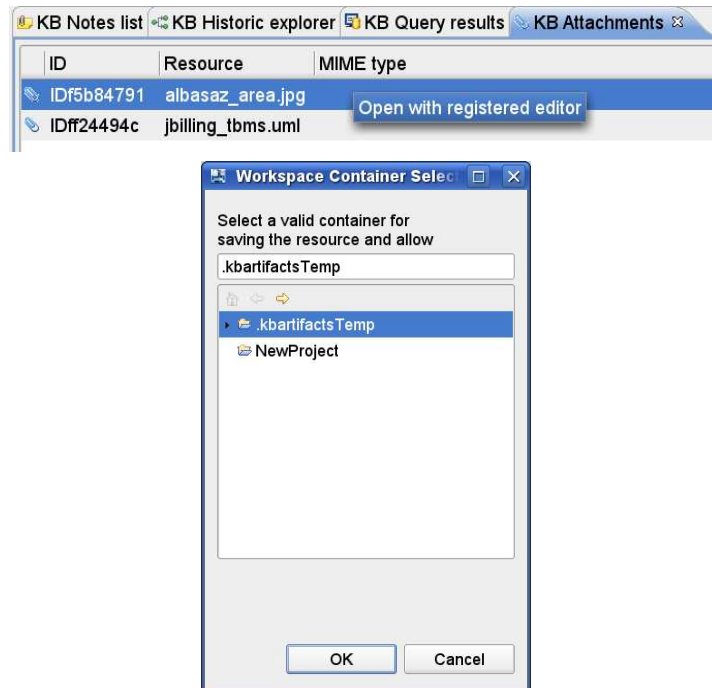
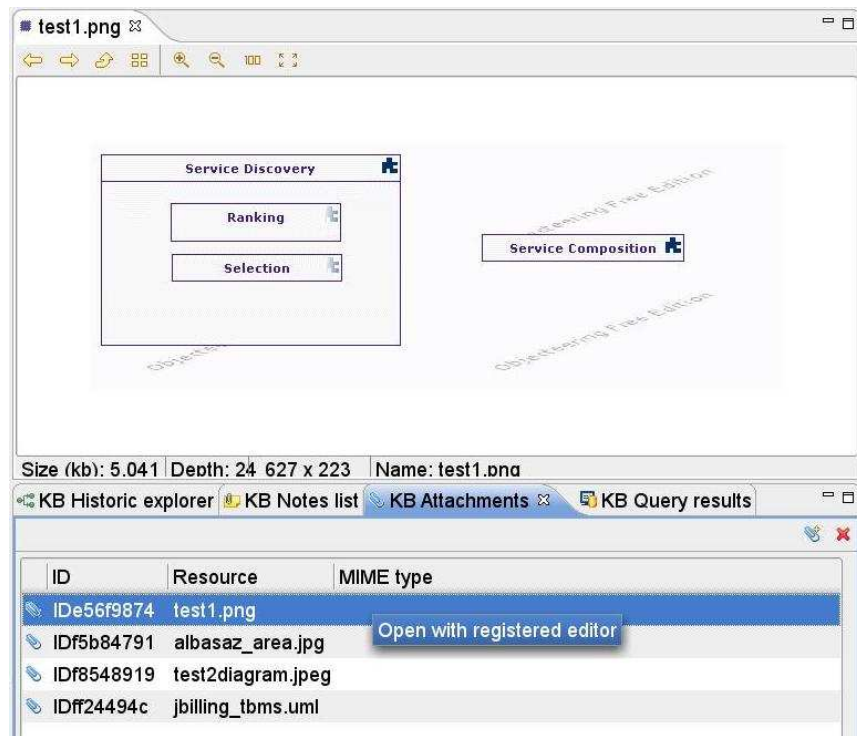


Figure 40 Open Attachment process

#### 4.1.5.3.1 KBR XSM diagram view in graphical format.

KB Attachments can be used to attach model diagrams in graphical format (JPG, etc) to concrete models stored within the KB Repository. In case it has been installed the QuickImage Eclipse Plugin Editor (see section 0), those graphical models can be exported from the repository into the workspace and opened.

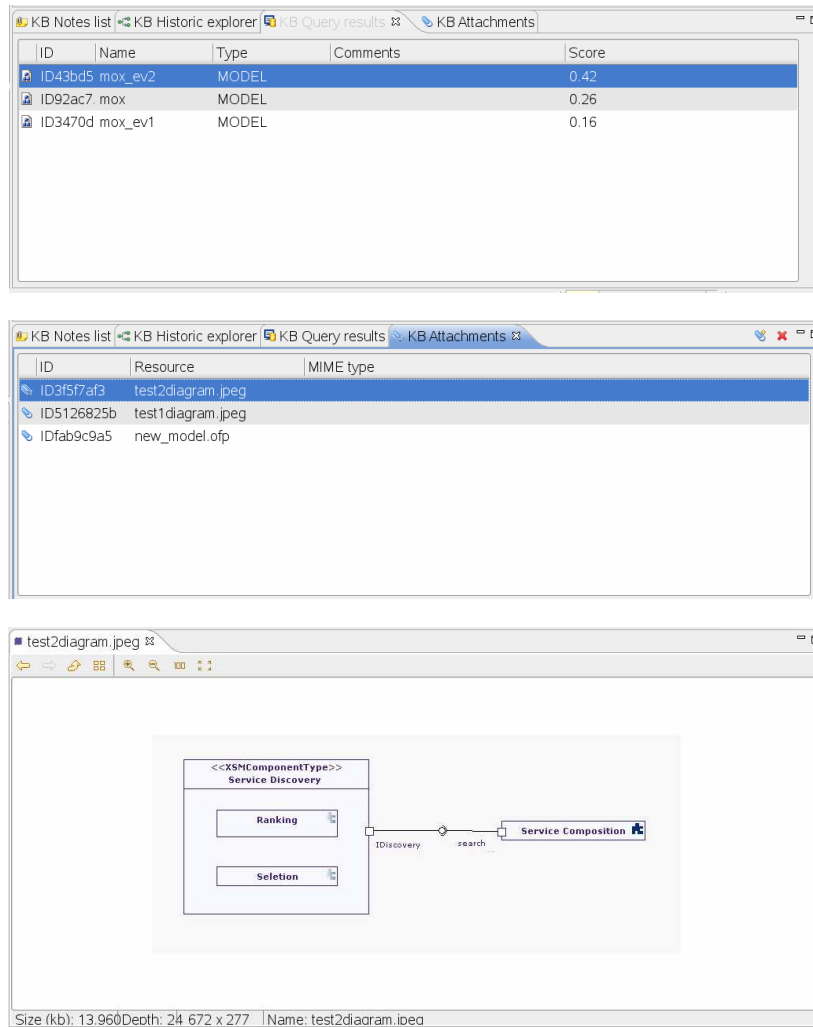


**Figure 41 Open Model Diagram attached**

Those graphical diagrams are created by the XSM Editor during the model edition phase and stored within the workspace. Then, those graphical diagrams can be attached to the model, once this has been exported to the KBR, as attachments (see section 4.1.5.3).

In a similar way, those graphical diagrams can be found and opened through the search mechanism:

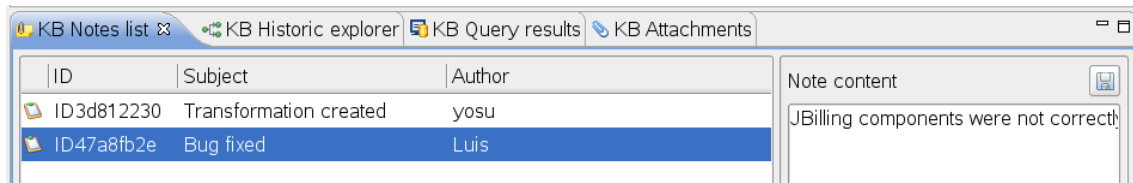
- First, perform a search to locate a particular model (see section 4.4),
- Second, select the model in the query view (the model will be highlighted also in the KB Explorer)
- Third, commute to the attachment view and select the appropriate attached diagram
- Then proceed as explained above in this section.



**Figure 42 Steps to find out and open a diagram**

#### 4.1.5.4 KB Notes view

Notes permit to annotate any repository artefact with additional information described using natural language, however this metadata are not used for searching purposes, but to complement the description of each artefact or to provide additional information. Notes can also been used to exchange messages between users. KB Notes list view in the auxiliary views panel on the bottom left lists all the notes attached to a particular repository artefact (Figure 43). By clicking on a note, the right panel show its content.



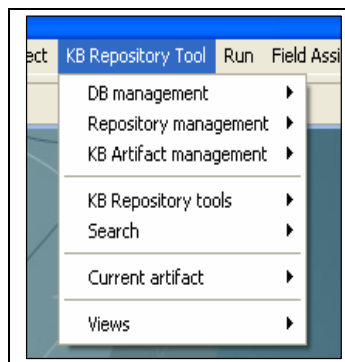
**Figure 43 KB Notes list content viewer**

#### 4.1.6 KB Repository Menu and button toolbar

Most of the KB Repository features are accessible from the KB Repository views main and contextual menus. Besides, most important features are also accessible through the KB Repository button tool bar (Figure 48) and main menu (Figure 44).

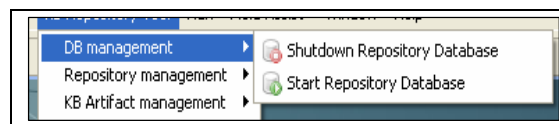
KB Repository Main menu:

The KB Repository main menu is divided into functionally differentiated submenu groups.



**Figure 44 KB Repository main menu**

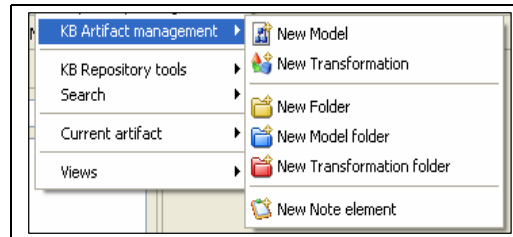
“DB Management” submenu: Access the eXist server-side control commands (see detailed section 4.3.1)



**Figure 45 DB Management menu**

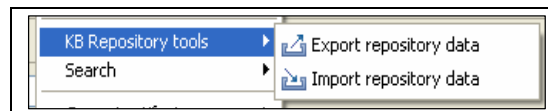
“Repository management” submenu: this menu provides a command to create KB Repository elements of type “Repository”, note that “Repository” type elements are “top level” element within the KB element hierarchy (see 4.2.3).

“Repository management” submenu: provides access for the instantiation of new KB artefacts using wizards (see 4.2.3).



**Figure 46 KB Artefact management**

“KB Repository tools” submenu: Export / Import tools commands (See 4.3.2).



**Figure 47 KB Repository tools menu**

“Search” submenu: the search command will open the Search dialog.

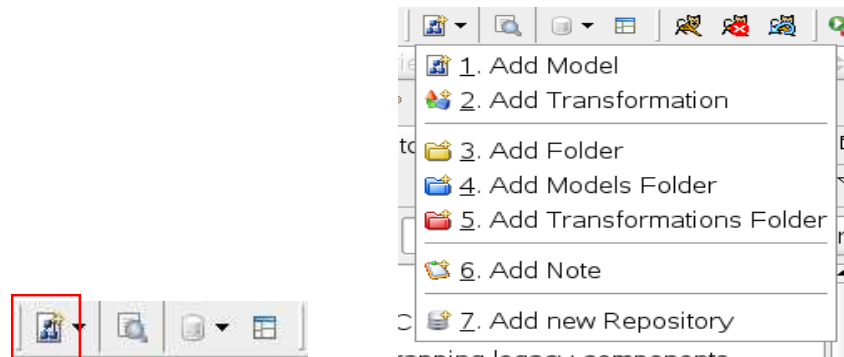
“Current artefact” submenu: offers commands to rename or remove the currently selected KB artefact.

“Views” submenu: provides the user commands for opening each view individually. Since the KB perspective is fully customizable, views can be closed, moved, etc. To display a hidden view or bring it to front just select the desired one choosing the respective command at this submenu.

### **KB Repository button toolbar**

The KB Repository button toolbar is shown in next pictures. It provides four buttons to perform some actions:

Repository artefacts creation button: opens a drop down menu supporting the creation of the commonest artefacts (i.e. any sort of folder, models, transformations, etc). See section 4.2 for more details.



**Figure 48 KB Repository button toolbar**

Repository search button: opens a wizard to perform a search under the folder selected in the KB Explorer view. See section 4.4 for more details.



Repository database management button: opens a drop down menu supporting the start up and shutdown of the local repository database. See section 4.3.1 for more details.



KB Repository perspective button: opens the KB Repository perspective.




## **4.2 KB Repository administration guide**

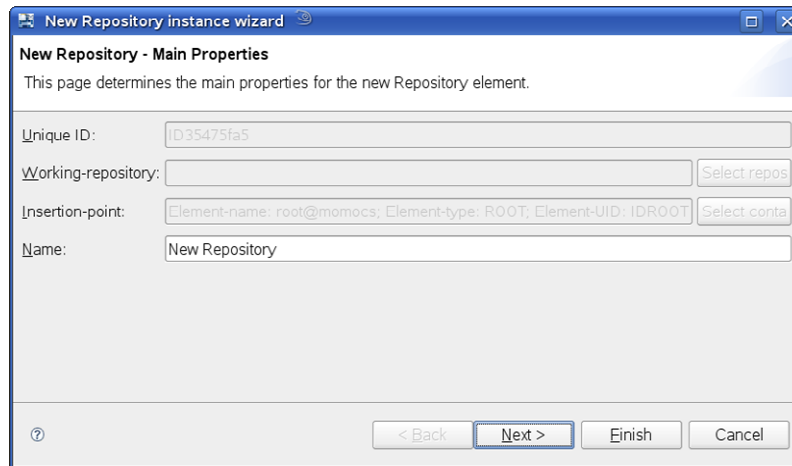
### **4.2.1 KB Repository domain model elements**

KB Repository is a centralised container of main artefacts created and used during the XIRUP Modernisation process [2007c]. KB Repository can manage several

thematic repositories. Each repository can be structured in a tree based folder structure, using normal folders (for creating the repository structure), model folders that may contain models and other model folders, transformation folders that may contain transformation rules and other transformation folders. Those elements constitute the basic artefacts contained within the KB Repository tool, which are relevant in the XIRUP modernisation process. Additional artefacts supported by the tool are notes, linked to each of aforementioned artefacts to complement them with additional information and attachments, which can also be linked with models and transformations, to complement them with additional artefacts (for instance, layout descriptions of models, matching logs on applied transformations, etc).

#### 4.2.2 Repository creation / removal

KB Repository can manage different repositories stored within the database. Those repositories are shown in the KB Repository Explorer. To create a new repository click on this icon  in the KB Repository Explorer view toolbar. Then a wizard opens (Figure 49). In the first wizard page we can edit the name of the repository. Next wizard pages are common to most of the KB Repository Explorer view wizards for creating artefacts. Second wizard page (Figure 50) allows the user to annotate the just created artefact (a repository in this case) with metadata: a set of keywords. Keywords can be added by clicking onto the “Add to list” tab button. To remove keywords or to move them up or down in the list, click “Remove”, “Move up”, “Move down” tag buttons respectively. Third and last wizard page (Figure 51) allows users to add comments describing the artefact to be created.



**New Repository instance wizard**

**New Repository - Main Properties**

This page determines the main properties for the new Repository element.

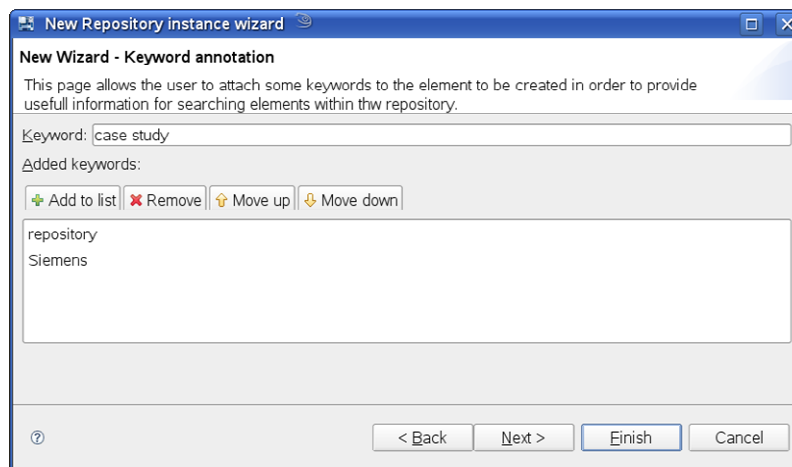
Unique ID: ID35475fa5

Working-repository:

Insertion-point:

Name: New Repository

**Figure 49 New repository instance wizard**



**New Repository instance wizard**

**New Wizard - Keyword annotation**

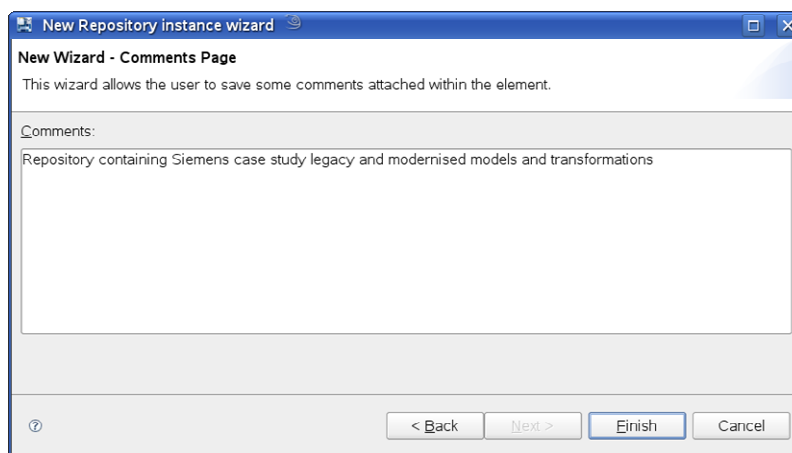
This page allows the user to attach some keywords to the element to be created in order to provide usefull information for searching elements within thw repository.

Keyword: case study

Added keywords:

repository  
Siemens

**Figure 50 New wizard keyword annotation page**



**New Repository instance wizard**

**New Wizard - Comments Page**

This wizard allows the user to save some comments attached within the element.

Comments:

Repository containing Siemens case study legacy and modernised models and transformations

**Figure 51 New wizard comments edition page**

### 4.2.3 Artefact creation

KB Repository organised in a tree based structure those artefacts produced and consumed by the XIRUP modernisation process. This section explains how to create and organised those artefacts within the repositories.

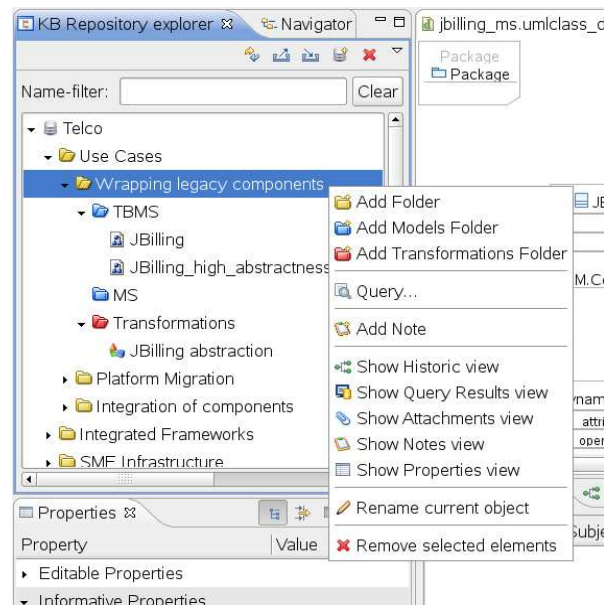
#### 4.2.3.1 Folders

Each repository managed by the KB Repository explorer view can be organized within a structure of folders and subfolders up to any depth. KB Repository manages three types of folders:

- Normal Folder: used to organise the repository structure. It may contain any other kind of folder.
- Model Folder: used to contain only model artefacts. It can contain only other model folders or models.

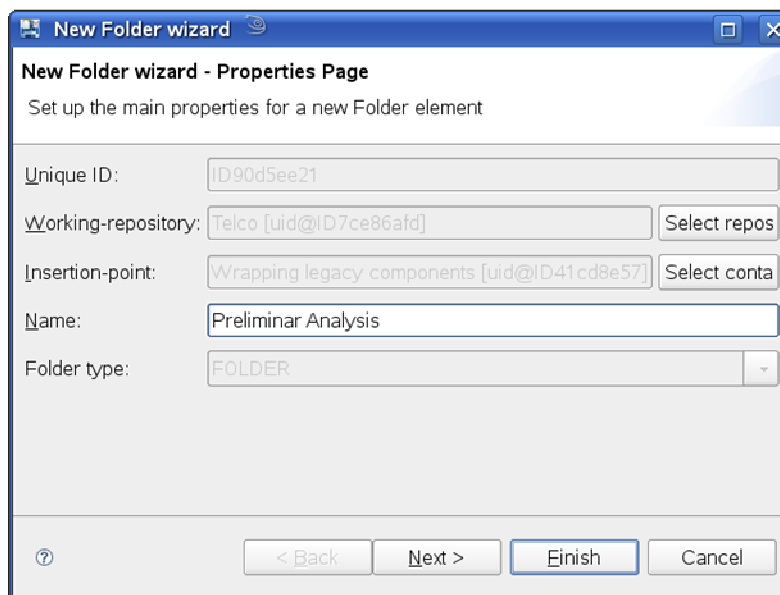
Transformation Folder: used to contain only transformation artefacts. It can contain only other transformation folders or transformations.

To create a folder of any type select the appropriate parent folder (or a repository) in the KB Repository explorer tree view and then right clicks to open the contextual menu (Figure 52). Select the appropriate menu entry, depending on the type of folder the user wants to create. The contextual menu entries change depending on the tree view item selected.



**Figure 52 KB Explorer view contextual menu**

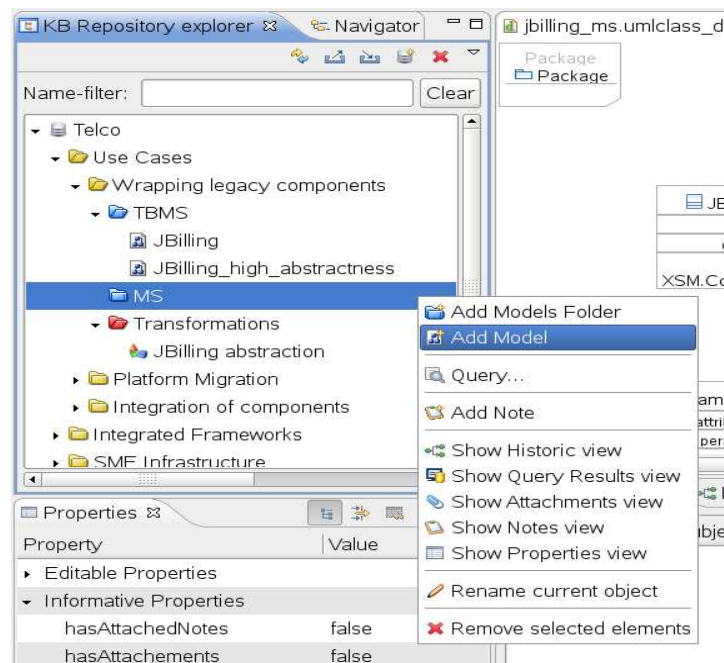
After selecting the contextual menu item, a New Folder wizard opens. The first page allows the user to edit the name of the new folder (Figure 53). Next pages allows to annotate the new folder with metadata and comments as explained above (see section 4.2.2)



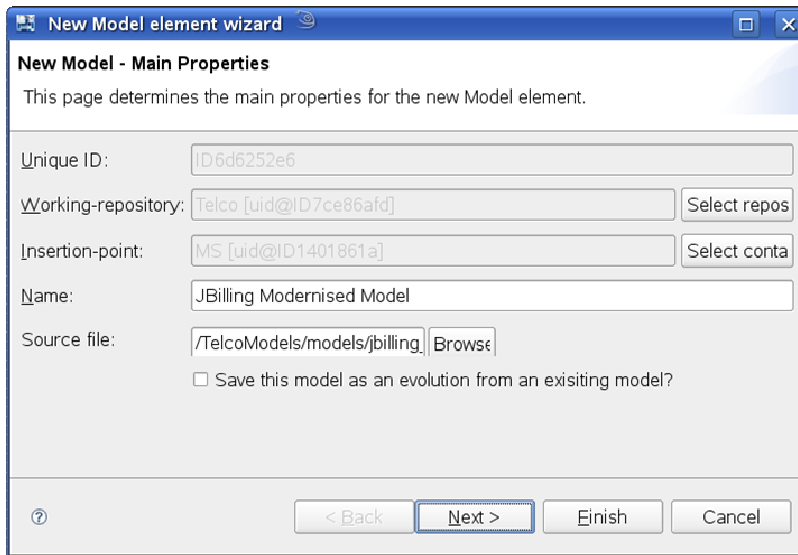
**Figure 53 New folder wizard**

#### 4.2.3.2 Model and transformation artefacts

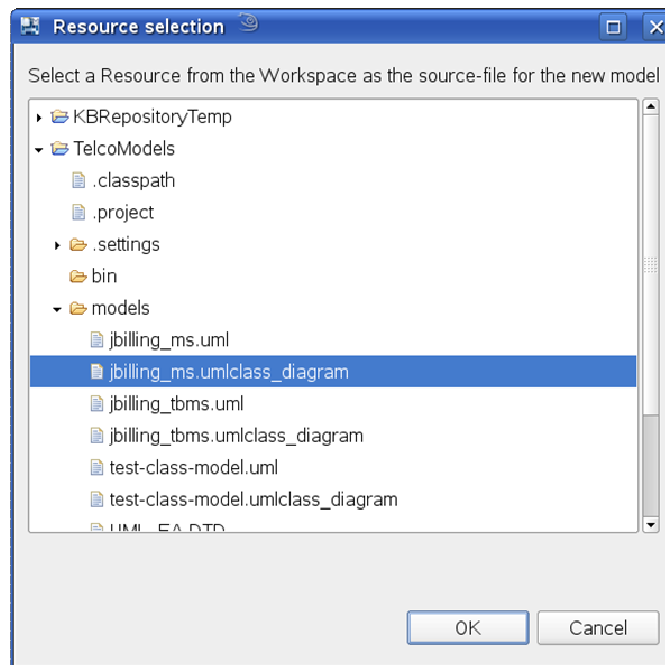
Within models or transformations folders the user can add models or transformations respectively. Just by right clicking onto a models or transformation folder the opened contextual menu shows the appropriate menu item to create a new model or transformation (Figure 54). Selecting “New Model” or “New Transformation” opens a wizard. The first page (Figure 55) permits the user to give a name for the model/transformation and selects the source file from the workspace (Figure 56). Rest of wizard pages are the same than in case of other artefact creation wizards aforementioned.



**Figure 54 Add Model contextual menu**



**Figure 55 New model wizard**



**Figure 56 Resource selection wizard**

#### 4.2.3.3 Notes and attachments

Notes can be created by selecting the appropriate repository artefact within the KB Repository explorer tree view, right clicking and selecting “Add Note”. The first “New Note” wizard page allows editing the note author and the subject (Figure 57). Second page is used to create the note content (Figure 58). Once created, the note is added to the KB Note list view in the auxiliary views panel (see section 4.2.3.3).

**New Note element wizard**

**New Note wizard - Main properties page**

This wizard allows the user to set up the note main properties.

Unique ID: ID6918e7c2

Working-repository: Telco [uid@ID7ce86afd] Select repos

Insertion-point: MS [uid@ID1401861a] Select target

Author: Luis

Subject: Include only final modernised models

< Back Next > Finish Cancel

**Figure 57 New note wizard**

**New Note element wizard**

**New Note wizard - Content page**

This wizard allows the user to set up the note content.

Content:

Models describing the process to modernise JBilling models should be put into the TBMS folder while final modernised models should be added to MS folder.

< Back Next > Finish Cancel

**Figure 58 New note wizard content page**

Regarding attachments, this tool release does not support to add user-selected attachment, but those described in section 4.2.3.3. However, future releases of KB Repository tool will also allow attaching any file to any model or transformation

added into the repository, to complement them. Attachments for a particular repository artefact are shown within the KB Attachments view (see section 4.1.5.3).

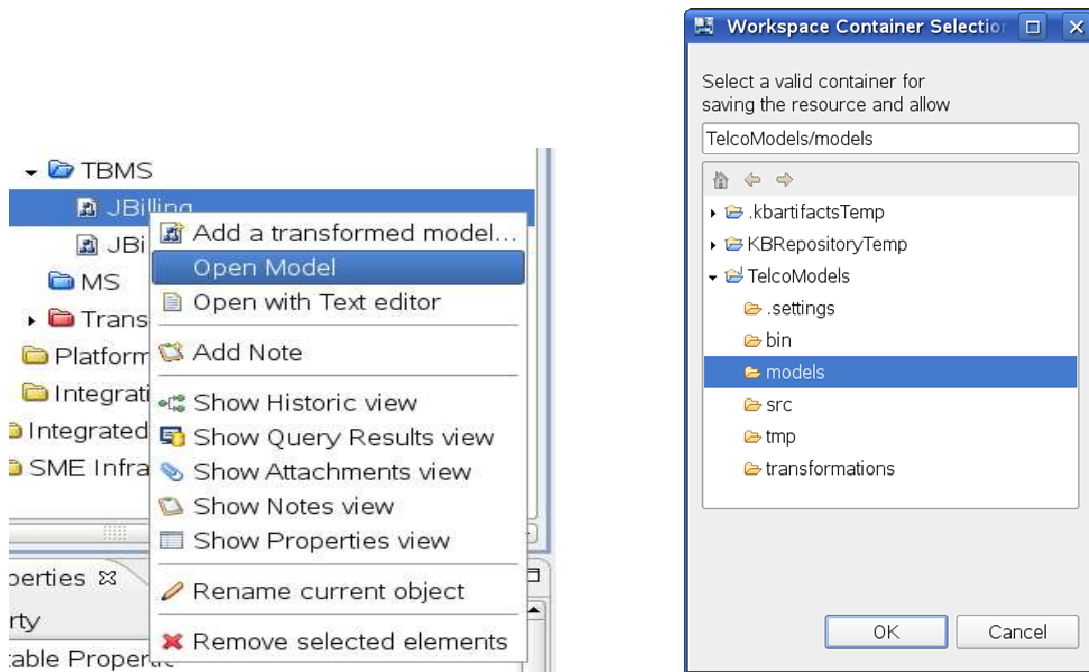
#### 4.2.4 Artefact edition

Repository artefacts like folders, model folders, transformation folders, models, transformation, notes can be edited whenever is required to change some properties as name, keyword based metadata annotation, comments and so on, by selecting the artefact within the KB Repository explorer tree view and accessing the Eclipse properties view (usually located on the left bottom panel, see section 4.1.2). In the properties view, the user can modified all the editable properties.

#### 4.2.5 Artefacts visualisation. Exporting Models and transformations

Some artefacts like models and transformations can be visualised by exporting them within the appropriate MOMOCS Suite tool: to visualise models, they have to be exported into the XSM Model Editor, and, for transformation into the XSM Transformation editor. Exporting models and transformations for visualisation means to export them into the current workspace, afterwards the appropriate tool is invoked passing to it the artefact path relative to the workspace, in order to be opened.

For example, to export a model within the workspace and to open it within the appropriate editor (similar procedure applies to open a transformation) right click on the model, select “Open Model” in the contextual menu.



**Figure 59 Open model contextual menu and workspace container selection wizard**

In the Workspace Container Selection wizard select the folder within some project of the workspace where to export the model (and all its attached files). Then, those files will be exported into the selected workspace folder and the appropriate editor invoked. The model appears in the KB Repository perspective central view, opened by the editor (Figure 60).

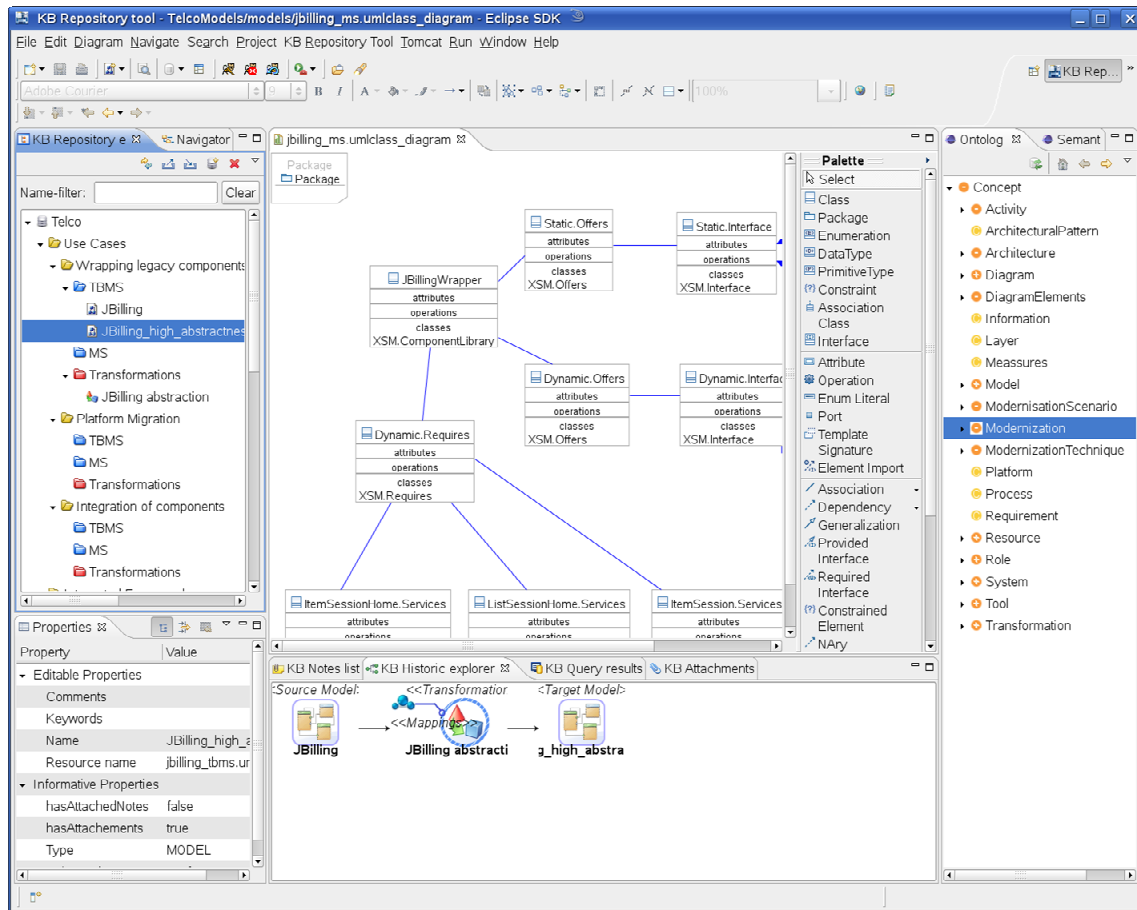


Figure 60 Model opened within the Momocs Suite workbench

#### 4.2.6 Artefact removal

In order to remove any KB Repository artefact: repositories, folders, model folders, transformation folders, models, transformation, notes, etc. just right click on the artefact and select the contextual menu entry "Remove selected elements". Confirm in the dialog. Note that this procedure can not be undone, so once deleted the artefact can not be restored.

### 4.2.7 Artefact annotation

Repository artefacts can be annotated with keyword based metadata when they are added into the repository, using the wizard (see section 4.2) or anytime after by using the Properties Editor view (see section 4.2.4). Additionally artefacts can be annotated anytime with semantic metadata using the Ontology Browser view (see section 4.1.3).

## 4.3 KB Repository auxiliary tools

### 4.3.1 Repository database management

KB Repository tool offers a basic management support for the local repository database:

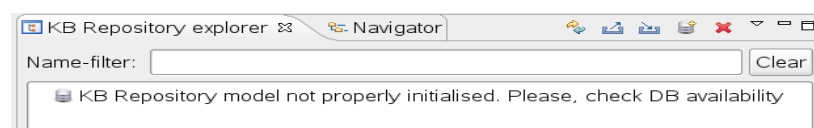
To start the local database, choose from the eclipse menu: KB Repository Tool>DB Management>Start Repository database or the same option from the KB Repository tool bar (see picture below)

To shutdown the local database choose from the eclipse menu: KB Repository Tool>DB Management>Shutdown Repository database or the same option from the KB Repository tool bar (see picture below)



**Figure 61 KB Repository database management toolbar menu**

After shutting down the local repository database or in case that the local or remote database is not accessible, the KB Repository Explorer shows an error message.




**Figure 62 KB Repository explorer showing database access error message**

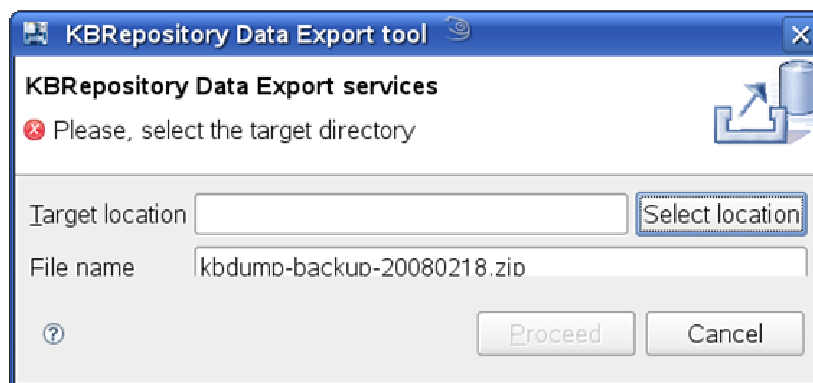
After starting the local repository database up or after refreshing the KB Explorer view once the remote makes available, the KB Explorer view removes this error message and shows the appropriate KB Repository content.

### 4.3.2 Repository export / import tool

KB Repository provides a facility to export the local database, creating a dump which can be later used to import its contents. This can be useful to migrate from one local database to another or to make a backup.

The procedure to export the local database<sup>3</sup> is as follows:

From “KB Repository Tool” menu select “KB Repository Tools/Export repository data” or click on the button  of the KB Repository explorer view. In the opened wizard select a target location for the database dump backup (click “Select location” button to navigate through the file system) and “proceed” button. A zip file containing the database dump backup will be place into the selected directory.



**Figure 63 KB Repository export wizard**

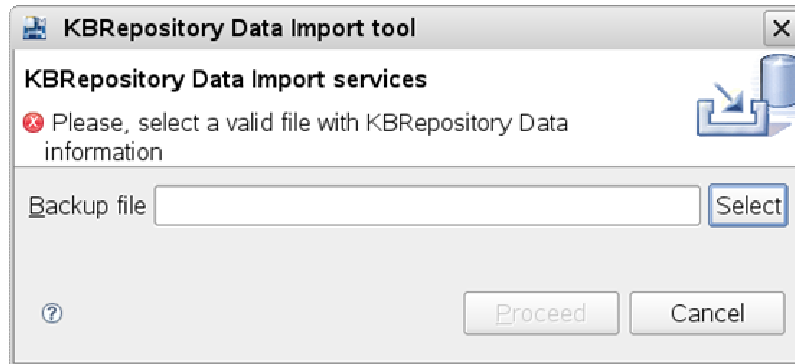
The procedure to import the local database is as follows:

From “KB Repository Tool” menu select “KB Repository Tools/Import repository data” or click on the button  of the KB Repository explorer view.

---

<sup>3</sup> The local database must be started both during the export and import processes.

In the opened wizard, select the database dump backup file by clicking onto the “Select” button and navigating through the file system looking for the appropriate backup. Then click on “Proceed” button.



**Figure 64 KB Repository import wizard**


## 4.4 KB Repository Search Facilities

One of the most important aspects covered by this tool is information retrieval.

As said before a medium transformation process could result in a vast number of KB artefacts like models or transformations. Search facilities are given to users to easily locate an artefact within the repository structure and take advantage of the tool.

Both semantic (still in development) and keyword search engine are provided. Taking expertise in both of them engines will leverage the modernisation process by easily store, track, and retrieve the different KB artefact during its life-cycle and furthermore. Keyword search uses advanced information retrieval algorithms based on keyword matching.

In order to recover a KB artefact effectively the user must annotate either with keyword metadata the artefact (see section 4.2.3) or using semantic metadata annotation (see section 4.1.3).

To open the search facilities dialog go to menu KB Repository tool >> Search >> clic the Search menu option, or press the search button  at main application toolbar

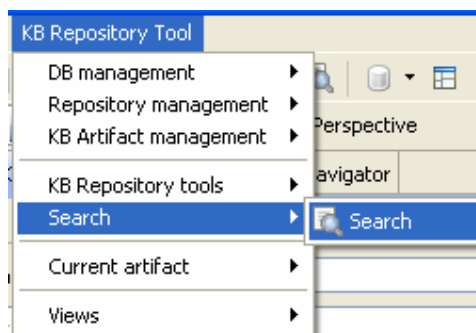


Figure 65 Search button at main toolbar

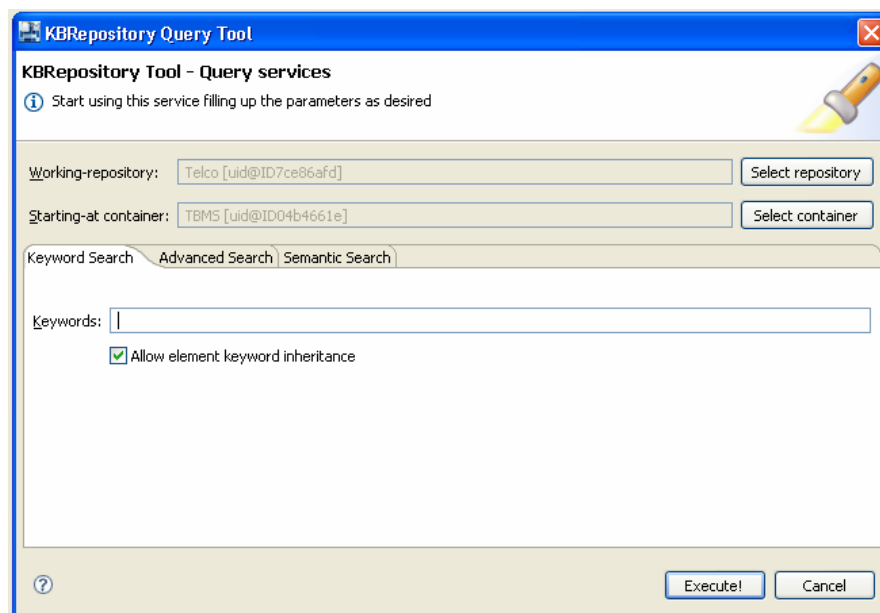
### 4.4.1 Keyword matching search

KB Repository uses the well-known and full-featured Open Source search engine named "Lucene" ([lucene.apache.org](http://lucene.apache.org)), from the Apache group, to perform the

keyword matching and KB artefacts indexing. In complex repository structures efficient information (object) retrieval workflows are vital to reduce time and mandatory to simplify the object location and selection. For this matter it is very important to take into account that the information (in the form of keywords in this case) the user attaches to a KB artefact is of vital importance for an optimal object recovery.

To start performing a keyword based search:

- Open the Search dialog:
- Select the “Keyword Search” tab, if not selected, and the keyword dialog will be ready to use:



**Figure 66 KB Repository query tool wizard**


- Select the container to search within:

Before searching, select a container where to start. Most of the times selecting an optimal starting point is the best option rather than querying the whole repository structure, by doing this the potential results and query time are significantly.

To proceed, first select the target repository by pressing the “Select repository” button and choose the desired one; second, select a KB container from within the repository.

This procedure applies to all the searching methods offered by the KB Repository tool.

Continue by pressing the button “Proceed” at the Search dialog. The results will be automatically shown at the KB Query results view.

Tip: It is easier to select the “Starting-at” target container while browsing the repository using the KB Repository Explorer view. Select a container (where it is supposed to be located the requested element/s) and press the “Search” button from the KB Repository main toolbar  or right-click the target container and select the option “Query...”.

#### 4.4.1.1 Search syntax

KB Repository tool uses Apache’s Open Source Lucene search engine ([lucene.apache.org](http://lucene.apache.org)), one of the most widely used OSS search engine that offers a lot of indexing and searching capabilities. Thus, the Lucene special syntax applies for querying the KB Artifacts.

Here below some examples to illustrate how to query and find inside the KB Repository.

A Simple query:

- To match all KB artifacts having the keyword “DB”, just insert the text “DB” at the “Keyword” text field and press “Proceed”.
- To match exactly a phrase within a keyword use the quotation marks. Executing the query: “Only Modernised models” will search those elements having a keyword with the exactly the given text.

Important note: Space characters inside a query string are used as a Boolean “OR” clause connector (see usage below), e.g. the query string: DB CORE, will match those elements owning a keyword with the value “DB” or the value of “CORE”.

Wildcards:

KB Repository tool keyword search facility supports single and multiple character wildcard searches.

- To perform a single character wildcard search use the "?" symbol.
- To perform a multiple character wildcard search use the "\*" symbol.

The single character wildcard search looks for keyword that match that with the single character replaced. For example, to search for "text" or "test" you can use the search: te?t

Multiple character wildcard searches looks for 0 or more characters. For example, to search for test, tests or tester, you can use the search: test\*

You can also use the wildcard searches in the middle of the keyword. te\*t

Important note: You may not use a \* or ? symbol as the first character of a search string.

AND, OR clauses: Boolean clause connectors are available

- AND: The AND operator matches documents where both keywords exist anywhere at the KB artifact keyword list. The symbol && can be used in place of the word AND.

To search for elements owning both keywords "modernised" and "j2ee" use: "modernised" AND "j2ee" query string.

- OR: this is the default conjunction operator. The symbol || can be used in place of the word OR.

To search for documents that contain either "fixed model" or just "modernised" use the query: "fixed model" OR modernised.

Complex queries:

- Just play with mixing up all of the explained query methods above. Supposing we have divided a modernisation process into different step and annotated each element with "Step XX" keyword and also with "Modernised" or "To Be Modernised" keyword, for recovering all modernised KB artifacts involved at modernisation step 10 to 20 we could use the query string: Step 1? AND Moder\*.

#### **4.4.1.2 Advanced Query creation**

This feature provides several facilities for building complex queries. By filling some fields at the Advanced Query dialog page the user will be able to match any KB artefact from the repository of a specific nature, like Model, Transformation etc., specify at the same sentence some keywords to match (using the same syntax explained at the previous section) or search for an specific element property value.

All KB repository artefacts have some common properties like Unique ID and Keywords and some other are specific fields of the element type. For example a note element has Author and Subject properties and the Model element does not; in contrast the Model element has a Name property and the Note element does not.

To access de Advanced Search dialog page:

As for the basic keyword search, rather than perform a search job over the whole repository structure, when possible, it is better to restrict the search scope by selecting a more specific container where to search within.

Once the user has selected an element from a KB repository structure the easiest way to go the search dialog is using the contextual menu, right-click the selected element and selection the Query menu option from the menu.

The search dialog appears, click at the Advanced Search tab and the query builder form will be displayed as follows:

The screenshot shows a window titled "KBRepository Query Tool" with a subtitle "KBRepository Tool - Query services". Below the subtitle is a help icon and the text "Start using this service filling up the parameters as desired". The window contains two input fields: "Working-repository:" with the value "Telco [uid@ID7ce86afd]" and a "Select repository" button; and "Starting-at container:" with the value "Use Cases [uid@IDff6695f6]" and a "Select container" button. Below these are three tabs: "Keyword Search", "Advanced Search" (which is selected), and "Semantic Search". The "Advanced Search" tab contains a query builder form with four fields: "Operator:" (a dropdown menu), "Element-type:" (a dropdown menu), "Field:" (a dropdown menu), and "Value:" (a text input field). Below these fields are two buttons: "Clear clauses" and "Add clause". At the bottom of the tab is a "Query clauses preview:" label above a large empty text area. At the very bottom of the window are two buttons: "Execute!" and "Cancel".

**Figure 67 Advanced Search dialog**

The top row of the form is where the users establish the parameters for the desired clause. The text area at the bottom row shows the actual query to be sent to the search engine.

**Searching by element types:**

- Choose the element type you want to match at Element-type dropdown list.
- Press the Add clause button
- Press the Execute! button

**Searching by element field value:**

- Select the desired element field by first choosing a specific element type. Note that for searching within the common artefact properties Keyword or Unique ID the element type labelled “[does not matter]” must be selected.
- Set the value or expression to match at the Value text field.
- Press the Add clause button
- Press the Execute! button

**Searching by both element type and field value in one clause:**

This search case implies the use of both previous procedures.

- Select at the target element type to match.
- Select the desired element property.
- Set the value or expression to match at the Value text field.
- Press the Add clause button and note that the newly created clause is composed by two clauses: the first one would match the element type and the second one the specified element property.
- Press the Execute! button

Mixing two or more clauses:

Creation of complex queries is allowed by joining them using a logical operator. That operator must be established once the first clause has been added, at this moment the Operator list is enabled in order to let the user choose the right one for the joining the clauses. A simple example of this kind of queries could be a set of clauses to join two different element types, e.g. Models and Transformations. Here are the steps required to perform the suggested action:

- Select the element type “Model” from the element type list
- Press the button Add clause, after that the Operator list will be enabled.
- Select the logical operator “OR” at operator list
- Select the element type “Transformation” at element type list
- Press the button Add clause.
- Press the Execute! Button

Bellow is the screenshot to illustrate the previous example:

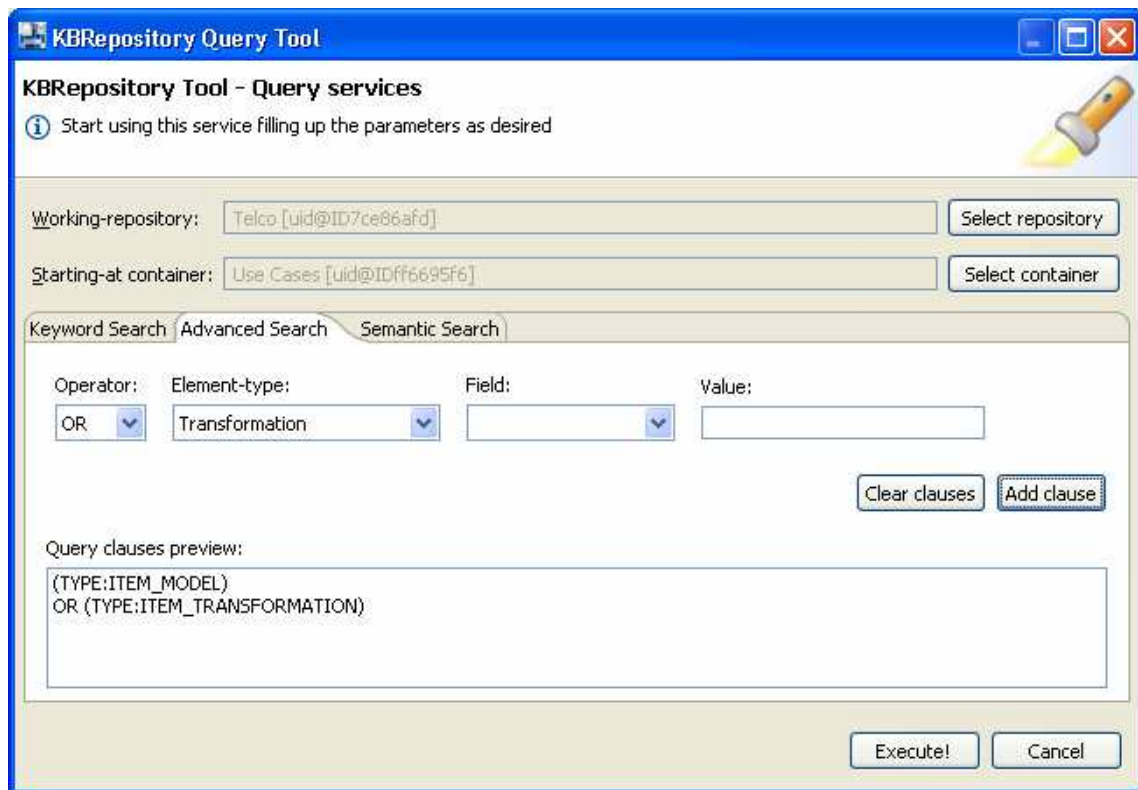


Figure 68 Query example

The results of the executed query are shown at the KB Query results list, see the next section.

#### **4.4.1.3 Browsing results**

Results of querying into the repository can be browsed within the KB query results view (see section 4.1.5.2).

### **4.4.2 Semantic Search**

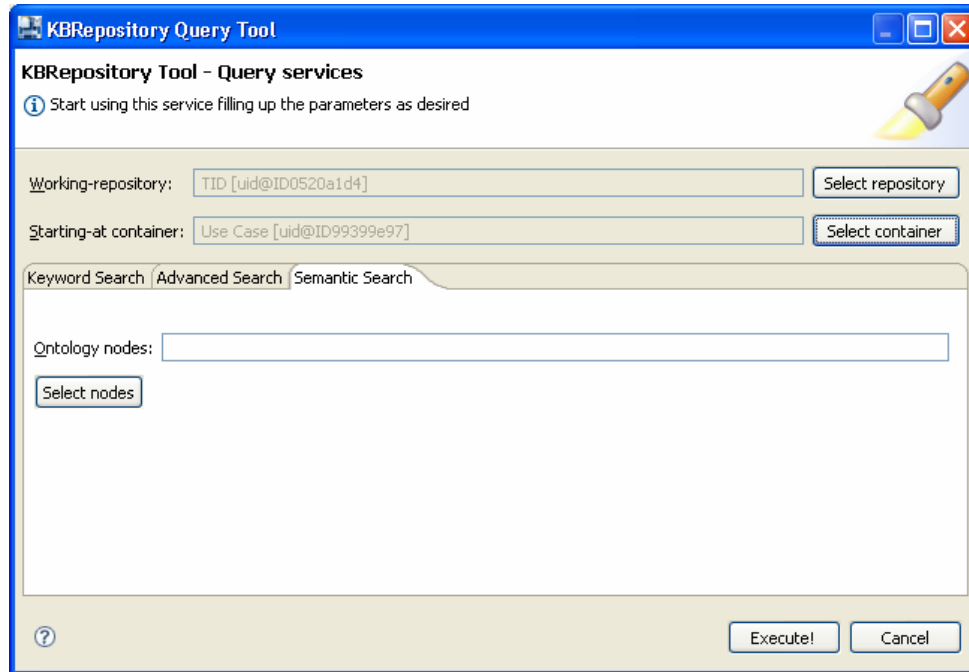
KB Repository tool provides a Semantic Search feature that allows the user to search within the repository making use of some semantic concepts taken from the current loaded ontology. It is important to warn that in order to perform a semantic search, the appropriate ontology used to annotate those KBR artefacts we are searching for (that is, the current domain ontology, ie. Telco or Industrial case ontologies) should be activated using the Ontology Browser view (see section 4.1.3). Normally, XA will be working on one of those specific domain with the appropriate ontology activated.

To access de Semantic Search dialog page:

Go to menu KB Repository tool >> Search >> and select the Search option.

Select a KB Container from the KB Explorer and display its contextual menu by making right-click over it. Select the Option Query...

Once the Search dialog is brought to the front, select the Semantic Search tab, if not currently selected. The figure bellow illustrates the semantic search dialog page.



**Figure 69 KB Repository query tool. Semantic search wizard**

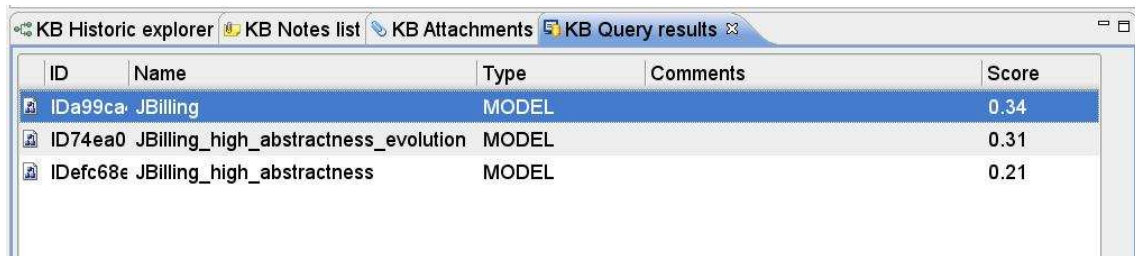
#### **Procedure to execute a semantic query:**

Like in case of keyword-based search, we have to select a KB container where to start searching. Push the button Select repository and select the desired repository instance at the KB selection service tree, after that select the target KB container by pressing the Select container. This is automatically accomplished by launching the search dialog using the contextual menu at the desired KB container in the main KB Explorer.

The next step for searching is to select those ontology concepts we want to match. It is allowed to select more than one concept from the ontology.

Press "Execute" button.

The results of the query are displayed at the KB query results view (see section 4.1.5.2) ordered by score. The semantic search and ranking algorithm implemented by KB Repository is explained in section 6.2.



ID	Name	Type	Comments	Score
IDa99ca	JBilling	MODEL		0.34
ID74ea0	JBilling_high_abstractness_evolution	MODEL		0.31
IDefc68e	JBilling_high_abstractness	MODEL		0.21

**Figure 70 Semantic Search results**

Results can be ordered by any of KB Query results columns by just clicking on each column header (the order direction will be swapped after each click).

## 5 Troubleshooting

### 5.1 Known issues

Under certain circumstances on Windows platforms, it has been observed sometimes that the internal KB Repository database (an instance of eXist xml database) hangs up, when it is managed from the KB Repository. Attempts to shut it down from KB Repository tool fail, therefore it is required to restart the KB Repository itself. This phenomenon has not been observed for KB Repository tool running on Linux operating systems.

EMF Compare plugin does not work with XSM if at least one XSM has not been previously opened from the workspace with the XSM Model Editor. To do that, open the Navigator View, right click in one XSM you have in the workspace and select the menu entry “Open With/XSM Model Editor”. Once opened, you can close the XSM Model Editor view and the EMF Compare feature will work correctly. It seems that XSM types are not internally registered for a proper EMF Compare behaviour if at least one of them has not been opened before with the XSM Model Editor.

### 5.2 Bug reporting

This first version of KB Repository Tool has not been yet assessed by final users, hence, even if a bug tracking facility has been supplied for such reporting, there have not been reported bugs. Incoming versions of this document will summarise the bugs reported and additional information about fixing.

### 5.3 FAQ

**Question:** After installing the KB Repository tool, I restarted the Eclipse platform as suggested, I try to open the KB Repository tool perspective and I get an error

message telling me: "An error occurred while trying to load Repository model from DB. Check DB availability".

**Answer:** The eXist database server is down. If your eXist instance is installed at the same machine as the KB Repository tool, select the option "Start Repository DB" from the KB Repository toolbar.

**Question:** When I try to start the database I get the following error: "Error trying to start the repository persistent-layer[...] Cannot run program[...] the system cannot find the path specified.

**Answer:** When working locally the KB Repository tool must be properly configured to work with an existing eXist database server. Go to menu Window >> Preferences >> and select the MOMOCS preference category. Expand the KB Repository sub-category and select the eXist DB Server option. At the right panel configure the root folder of the eXist DB installation, press Apply, and OK. Try to restart the database again.

**Question:** When I try to start the database I get the following error: "Error trying to start the repository persistent-layer[...] Cannot run program[...] the system cannot find the path specified".

**Answer:** that means that the tool has not been properly configured. When working locally the KB Repository tool must be properly configured to work with an existing eXist database server. Go to menu Window >> Preferences >> and select the MOMOCS preference category. Expand the KB Repository sub-category and select the eXist DB Server option. At the right panel configure the root folder of the eXist DB installation, press Apply, and OK. Try to restart the database again.

**Question:** When I try to start the database I get the following error: "Error trying to start the repository persistent-layer[...] KBPersistence could not find the KB

Repository root collection. Please, check if the collection has been correctly installed into the server”.

**Answer:** The server is already installed and ready to run but the DB structure has not been yet initialised to work with KB Repository tool. Press Ok at the error message and you will be prompted to initialise the DB with some sample data. Press OK to proceed and wait until the perspective is refreshed. Another way to initialise the DB consist of importing an existing KB DB dump file, see the section 4.3.2 for instructions on how to do it.

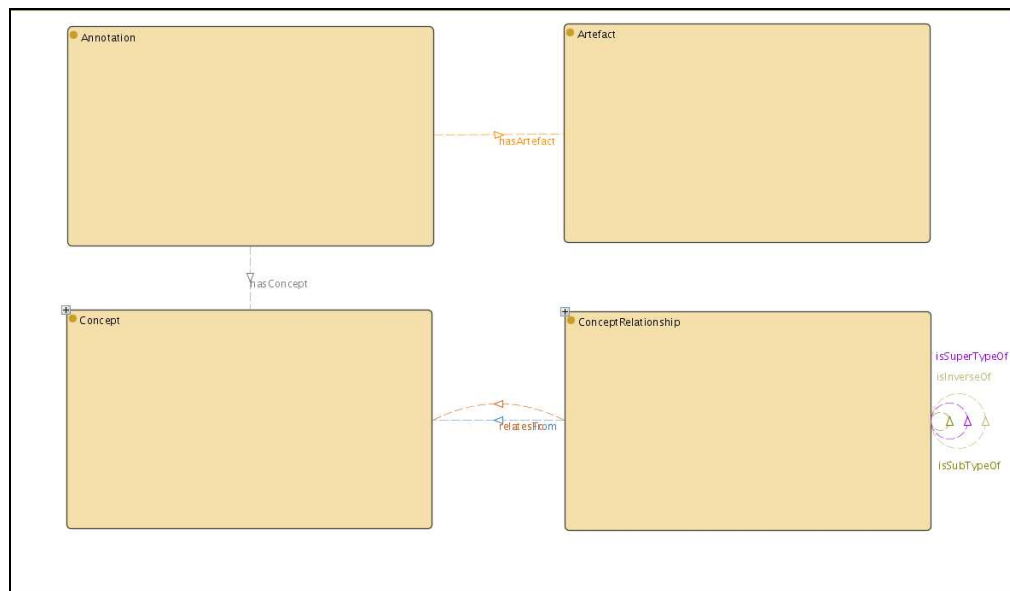
## 6 Appendixes

### 6.1 KBR Ontology

KBR uses a set of ontologies (one for each MOMOCS case study domain) to describe the available vocabularies used in the semantic annotation of KBR artefacts and in the building of the knowledge base that stores the artefacts annotations. The schema behind those ontologies follows a common design pattern which is depicted in Figure 70.

This design pattern is motivated by the work described in [2007d].

There are four main ontology classes: Artefact, Concept, Annotation and ConceptRelationship. Individuals of the Artefact class correspond to the annotated artefacts stored in the KBR. Individuals of the Annotation class correspond to the attached annotations. Concepts subclasses (not shown in Figure 70) are the domain concepts (for instance, of the Telco or Industrial use cases domains) that are visualised in the KB Ontology Browser and whose individuals are used to annotate artefacts. Those subclasses of the Concept class constitute the concrete domain ontology. The individuals of the ConceptRelationship class represent relationships amongst the domain concepts (individuals of subclasses of the Concept class).



**Figure 71 KBR Ontology design pattern**

The Annotation class is linked to the Artefact class by the “has Artefact” property (and its inverse property, “isAnnotatedBy”, not show in Figure 70) and linked to the Concept class through the “hasConcept” property (and its inverse property, “isLinkedTo” not show in Figure 70), whereby there is bidirectional navigation among those classes.

The individuals of the Concept subclasses are related through ConceptRelationship individuals, using the “relatesTo” and “relatesFrom” properties, which allow bidirectional navigation among concepts. This design pattern allows to weight concept relationships.

Concepts are also weighted through the “hasWeight” property (not shown).

When the user annotates a KBR artefact using the KB Ontology Browser, new individuals of the Annotation class are created and the properties that link those individuals with those of the Artefact and Concept classes are fulfilled. That allows the retrieval of the annotations for a particular artefact or the retrieval of the artefacts annotated with a particular concept (as used in the semantic search process, as explained in next section).

Reviewers in the review recommendation 4 showed their concern on the usage of KBR ontology as taxonomy rather than as a true ontology. However, once

considering the above description, we try to maintain that the KBR ontology is a true ontology and not a simple hierarchical organization of concepts, since we have created a shared formal conceptualisation of some specific domains (TELCO and INDUSTRIAL) with relations between concepts, where it is possible to make some reasoning, using the JENA API and OWL DL. After reviewing the semantic algorithm and ranking algorithm (see next section), the reader can accept that the usage of the KBR ontology goes beyond a simple taxonomy, since the relations among concepts are considered. However, it is true that due to the search nature within the KBR where the artefacts are annotated with several ontology concepts as metadata, it is not possible to place a particular artefact as a concrete individual of a particular ontology concept, since that exact matching is not feasible in most of the cases. Therefore, real semantic reasoning to extract concrete knowledge (artefacts in this case) is not possible, because of which, an approximate semantic search algorithm is required, giving the most accurate result with some imprecision (given by the score). This situation is common where some knowledge can not be included in the Knowledge Base as individuals but dispersed along some of them.

Maybe the KBR ontology usage as taxonomy misconception may be influenced by the mechanism to annotate artefacts with semantic metadata using the KBR Ontology Browser view, provided that that view only shows the ontology concepts in a hierarchical tree-based representation. That approach was adopted to simplify the view, and the relations among concepts were deliberately omitted for the sake of simplifying the view and usage for final users. Otherwise, it would be quite difficult to show in a proper and manageable way all ontology concepts and their relationships, which are determined by domain experts, but not for XAs.

## **6.2 Semantic Search and ranking algorithm**

KBR has improved its semantic search capabilities through the improvements provided in its semantic search and ranking algorithm as compared to the previous KBR release. Part of the KBR development work among M17 and M24 was allocated to address the reviewers' recommendation 4 on this regards.

This section describes the semantic search and ranking algorithm applied by the KBR during the search process. This algorithm and the KBR ontology design pattern have been inspired by the work presented in [2007d].

The search and ranking algorithm description starts with some definitions:

KBR stores a set of artefacts  $\alpha_n \in A \equiv \{\alpha_n\}_{n=1}^N$  where  $A$  constitutes the repository of  $N$  artefacts. Each artefact  $\alpha_n$  is annotated by a set of concepts  $c^n \equiv \{c_m^n\}_{m=1}^M \subset O$  among those selected from the ontology  $O$ . A query criteria provided by the user consists of a set of concepts selected from the ontology  $O$ ,  $q \equiv \{\omega_k q_k\}_{k=1}^K$ , where the weights  $\omega_k$  are determined by the ranking algorithm after reasoning upon the own ontology (see below where the ranking algorithm is explained). A candidate artefact  $\tilde{\alpha}_i$  found out by the search process has been annotated by some concepts  $\tilde{c}^i \equiv \{\tilde{\omega}_j \tilde{c}_j^i\}_{j=1}^J \subset O$ , where the weights  $\tilde{\omega}_k$  are determined as afore explained.

The search algorithm determines the set of candidate artefacts,  $\tilde{A} \equiv \{\tilde{\alpha}_i\}_{i=1}^I$ , that have been annotated with at least one of the concepts of the query criteria  $\tilde{c}_i \in q$ .  $\tilde{c}_q^i \equiv \{\tilde{c}_l^i\}_{l=1}^L$ ,  $\tilde{c}_l^i \in q$  is the set of annotated concepts that belong to the search criteria concepts for a particular candidate artefact. The original search results set dimension,  $I \equiv \dim(\tilde{A})$ , can be reduced before applying the ranking algorithm, by sorting the  $\tilde{A}$  elements according to the decreasing dimension of  $\tilde{c}_q^i$ , taking those with higher dimension.

Then the candidate artefacts  $\tilde{A}$  are sorting according to their higher computed similarity, which is defined as:

$$sim(q, \tilde{\alpha}_i) \equiv \frac{q \cdot \tilde{c}_q^i}{|q| |\tilde{c}_q^i|} [1]$$

That is, the similarity between a candidate artefact and the search query is defined as the normalised Euclidean scalar projection (scalar product) between the candidate artefact  $\tilde{\alpha}_i$  and the query  $q$ . The norm  $||$  is the Euclidean one. The scalar product is

calculated as follows:  $q \cdot \tilde{c}_q^i = \sum_{k=1}^K x_k$  where  $x_k$  is  $\begin{cases} 0 & \text{if } \tilde{c}_k \notin q \\ \omega_k \cdot \tilde{\omega}_k & \text{if } \tilde{c}_k \in q \end{cases}$ , where  $\omega_k$  is the weight of the query concept  $c_k$ , and  $\tilde{\omega}_k$  is the weight of the candidate artefact annotation concept  $\tilde{c}_k$ . Sum terms contributing to the similarity are only those build by candidate artefact annotation concepts belonging to the query concept space.

The weights  $\omega_k$  and  $\tilde{\omega}_k$  are calculated by the ranking algorithm applying the following algorithm:

Each ontology concept  $c_k$  has its internal weight  $\omega_k^0$  determined by the ontology designer (by default all concepts have  $\omega_k^0 = 1, \forall c_k \in O$ ). The calculated concept weight is:

$$\omega_k = \omega_k^H + \omega_k^{\mathfrak{R}} [2]$$

Where

a)  $\omega_k^H$  is the hierarchical weight, calculated by the aggregation of the internal weight of all the hierarchical ancestors (related by “isPartOf” relationships) of the concept  $c_k$  in the ontology. That is, the deeper is the concept in the ontology hierarchy (the specialised the concept is) the higher is its weight:

$$\omega_k^H = \sum_{h=1}^H \omega_k^{0^h}$$

where  $\omega_k^{0^h}$  is the internal weight of  $c_k^h$ , the ancestor in position h of the concept  $c_k$  in the ontology hierarchy.

b)  $\omega_k^{\mathfrak{R}(q)}$  is the connectivity weight between the concept  $c_k$  and the rest of query concepts or candidate artefact concepts set, accordingly.

$$\omega_k^{\mathfrak{R}} = \sum_{h=1}^H \omega_k^{0 \mathfrak{R}(k,j)}$$

where  $\omega_k^{0 \mathfrak{R}(k,j)}$  is the internal weight of an explicit relationship (excepting “isPartOf” relationships) between the concept  $c_k$  and another concept  $c_j$  belonging to the same query criteria or candidate artefact annotation concept set, accordingly.

The first term of the weight sum [2] tries to make prevailed specialized concepts upon those less specialized in the search query and artefacts annotation concepts set. The second term tries to make dominant those concepts with more connections with other concepts of the same search query or artefacts annotation concepts set.

The internal concept weight can be set after some experimental tuning, in order to achieve a better search precision.

Search results are sorting according to the increasing calculated similarity, so better scored candidate artefacts are shown on the top of the list.

### **6.3 Sample Repository based on MOMOCS TID UC's**

As an add-on for the tool a KB Repository dump is also released. It contains the both the KB Repository data structure of folders, folder for models, etc. and a base Ontology source file based on the TID use cases for MOMOCS.

The KB repository dump compressed file is available under the location: [https://services.txt.it/crs\\_subversioning/momocs/WP5/dev/atos/db-dump/kbdump-backup-20080123.zip](https://services.txt.it/crs_subversioning/momocs/WP5/dev/atos/db-dump/kbdump-backup-20080123.zip)

To import the KB dump into the KB Repository tool just download the file and proceed with the import process as seen on Section 4.3.2.

### **6.4 Ongoing Features**

This section summarises main incoming features not yet implemented into the current release of KB Repository Tool:

- Concurrent access to local or remote KB Repositories database instance: current KB Repository implementation support both local and remote eXist database instances but only with one user access. Incoming releases will support concurrent access to the database instance, managing user's account, user's concurrent changes, commits, updates and so on.

- Query storage: at Search dialog, a new command will allow the user to execute and store, using a familiar name, the query being processed. Later the user would be able to easily locate and run his stored queries.
- Keyword inheritance: in a top-down inheritance sequence a KB artefact could inherit the parent's keywords. This could help the user to not repeat the keywords introduced before and to give an element more information in order to evaluate the relevancy level in queries.

## 7 Annexes

### 7.1 Acronyms and Glossary

Acronym	Meaning
XSM	XIRUP system model
TBMS	To be modernised system
MS	Modernised system
UC	Use case
TID	Telefonica I+D
KB	Knowledge base
RCP	Rich client platform
SWT	Standard widget toolkit
PIM	Platform independent model
CIM	Computer independent model
PSM	Platform specific model
JVM	Java virtual machine
XIRUP	eXtreme end-User dRiven Process
FAQ	Frequent answered questions
JRE	Java Runtime Environment
URI	Uniform Resource Identifier
DB	Database
UML	Unified Modelling Language
OS	Operating System
OSX	Macintosh Operating System

Table 1 Acronyms and Glossary

### 7.2 Reference Documents

[2007a], F. Garijo, J. Pavón, C. Rodriguez: Momocs Telco use cases.

[2007b], Momocs team, D4.1 XIRUP supporting tools specification.

[2007c], Momocs team, D31 XIRUP Methodology

[eXist] Open Source Native XML Database. <http://exist.sourceforge.net/>

[2007d] P. Castells et al. An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval, IEEE Transactions on Knowledge and Data Engineering. February 2007 (Vol. 19, No. 2) pp. 261-272