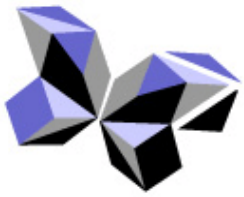


Digital Business Ecosystem

Contract n°507953

Workpackage WP20: User Interface

Deliverable D20.9: DBE Studio UI Analysis and Recommendations

Information Society Technologies

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

Contract Number: 507953

Project Acronym: DBE

Title: Digital Business Ecosystem

Deliverable N°: D20.9

Due dates: 30/11/2006

Delivery Date: 04/12/2006

Short Description:

This document evaluates the DBE Studio's user interface (UI) with respect to human interface guidelines issued by the Eclipse foundation. These guidelines are built upon platform human interface guidelines (HIGs), which include the Windows HIG, Apple HIG and the Gnome HIG.

Author: Intel Ireland Ltd.

Partners contributed: Jukka Hutamaki, TTY/DMI Hypermedia Laboratorio, Tampere University of Technology.

Made available to: Public

Versioning		
Version	**Date**	**Author, Organisation**
0.1	05/07/2005	Andy Edmonds, Intel Ireland Ltd. **Draft**
0.2	20/11/2006	Andy Edmonds, Intel Ireland Ltd.
0.3	15/12/2006	Andy Edmonds, Intel Ireland Ltd.

Quality check:

1st Internal Reviewer: Fotis G. Kazasis, Technical University of Crete.

2nd Internal Reviewer: Pierfranco Ferronato, Soluta.

TABLE OF CONTENTS

LISTING OF FIGURES

LISTING OF TABLES

Executive Summary

This document is designed to address the second objective of the Interaction Design task C19: "To ensure a consistent, coherent UI across all infrastructural components of the DBE by collaborating with the infrastructural developers of those components".

Initially entitled "User Interaction Design for DBE Studio", it became clear that due to the structure, practicalities and time-lines of the implementation and integration of the various DBE Studio components, a comprehensive top-down interaction-design process could not be usefully employed. This document describes the bottoms-up approach to enhancing the user interaction of the DBE Studio that was adopted instead…hence its updated, more specific title.

The process of creating this document has been iterative in its approach. A detailed UI evaluation of the 0.2.1 version of the DBE Studio was performed, and recommendations were made in a draft of this document. This document was distributed [2] to all partners involved in the implementation of the DBE Studio through the use of the DBE Studio mailing lists, hosted on sourceforge [1]. The current, final version of this document is a combination of evaluations performed both for 0.2.1 and the latest (as of 27/11/2006 [3]) release, 0.3.0.

It should be noted that the implementation of the individual recommendations is at the discretion of the appropriate partners, and subject, naturally, to the availability of their resources. Although it can not be expected that all recommendations will be adopted, significant enhancements to the user interaction within DBE Studio have already been made as can be seen in the chapter "UI Enhancements to Date".

The final output and suggestions contained within this document are being distributed to the relevant partners by both using the original distribution mechanism and also by logging all suggestions as feature requests (available here [4]) on the sourceforge project site.

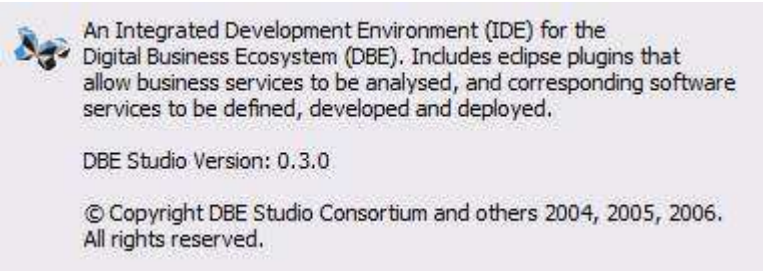


Figure 1 DBE Studio Version Information

1 Introduction

The goal of this document is to provide recommendations that will help the user's interaction with the DBE Studio as a whole. The suggestions contained in the feedback are purely recommendations and developers are under no obligation to implement them. Due to the structure, practicalities and time-lines of the implementation and integration of the various DBE Studio components, it was not possible to deliver a top-down user-centred interaction design or analysis within the time frame of the project. Of particular concern was the possibility that such an exercise would require significant reengineering or even merging of components in the DBE Studio - work whose outcomes could not be predicted: they may have required merging of components (and possibly merging of tasks) of different partners, for example, if not significant reengineering. To guide this bottom-up approach, this document evaluates the DBE Studio's [1] user interface (UI) with respect to human interface guidelines issued by the Eclipse foundation [5]. These guidelines are built upon platform human interface guidelines (HIGs), which include the Windows HIG [6], Apple HIG [7] and the Gnome HIG [8]. These platform-specific HIGs are valuable references and guidelines for creating the user interface components that comprise the Eclipse UI frameworks. However, For the needs of this evaluation, they are not as relevant as the DBE Studio builds on the Eclipse UI frameworks. These frameworks, whose guidelines are specified in the Eclipse HIG can be seen as a meta-HIG with respect to the platform HIGs.

By adopting both guidelines and suggestions within the Eclipse HIG and this document, visual and functional consistency throughout the DBE Studio will be promoted. It is important to maintain a reasonable level of consistency with the visual and behavioral aspects of the application, as a user builds up expectations about how an interface works. Where inconsistencies are exhibited by a user interface, for example any UI in the DBE Studio, a user can become confused and unnecessary complexity is added to the process. This confusion is something to be avoided at all costs when we consider that many of the users of the DBE Studio may not have much experience and/or training with such technical environments. As such to counter this, there is the ever present need for improved simplicity. Simplicity can be a difficult attribute to reflect in user interfaces, especially in highly complex modeling applications, but it is highly desired. Simplicity is not the same

as being simplistic. Creating something simple requires a good deal of code and work and, hence, why it can be difficult to convey to the user.

This evaluation is only part of a larger process of user interface design. Within the act of user interface design there are a number of phases that comprise it. Those phases are:

- *Requirements Gathering* – in this phase the required functionality of the system to be implemented are collected. This phase is based on potential requirements and needs of the user of the system.
- *User analysis* – in this phase discussions are initiated with users that will have direct contact with the system or will manage such users of the system. At this stage, typical questions asked would include "What do you want the system to do?" and "How would the system fit into your daily activities" and "Describe what your work involves, step-by-step". This is an important phase as user interface design needs an excellent and in-depth understanding of a user's needs.
- *Information Architecture* – in this phase a model of how information, within the proposed system, is created. At this stage optimizations of that information model can be suggested or indeed planned.
- *Prototyping* – in this phase basic prototypes using rapid prototyping processes are created. Such processes include the forms of paper prototypes or simple interactive screens implemented using such systems as Adobe Flash [9] and Director [10]. Typically, these prototypes' look and feel is not considered, instead focusing on the functionality and purpose of the interface.
- *Usability Testing* – in this phase the prototypes developed are tested upon the user-base that was used in the user analysis phase. For example, users are allowed navigate the prototypes and are asked to speak through their actions using a technique known as "talk-aloud protocol" [11]
- *Graphic Interface Design* – in this final phase, the look and feel of the system is created based on the results from the usability testing. This phase is carried out by a graphical designer.

Aspects that are considered right throughout this process can be summarized from work done by Nielsen [12] with system frameworks for system acceptability. Such a framework can define usefulness[1] as being composed of:

- Learnability (e.g. intuitive navigation)
- Efficiency of use
- Memorability
- Few and non-catastrophic errors
- Subjective satisfaction

As mentioned in the first paragraph of this introduction, the DBE Studio is built on the Eclipse platform [13]. There is plenty documentation written on the actual platform regarding its architecture but for the needs of this evaluation, this information is somewhat mute. What is more of relevance to this evaluation is how the Eclipse platform presents its user interface to the user. An overview and description of this is now presented.

The remainder of this document makes recommendations on DBE Studio plugins. Those recommendations and correspondingly, preceding chapters, are organized by category. In chapter 2 recommendations are made that are applicable to general DBE Studio wide aspects. Chapter 3 details recommendations on DBE Studio perspectives. Following that, chapter 4 makes recommendations based on the editors contained in DBE Studio. Related to those editors in chapter 4, DBE Studio views are analyzed and resultant recommendations are made in chapter 5. Chapter 6 makes recommendations on DBE Studio wizards, chapter 7 details recommendations on documentation within the DBE Studio and chapter 8 contains recommendations on DBE Studio preferences. Penultimately, UI enhancements are detailed in chapter 9 and finally we conclude in chapter 10.

[1] Incidentally, an ISO standard defines this too. The document *ISO 9241-11 (1998) Guidance on Usability* issued by International Organization for Standardization defines usability as:
"The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use."

1.1 Eclipse

1.1.1 Overview

Eclipse [13] is a universal tool platform - an open, extensible IDE for "anything". The Eclipse platform is very flexible and extensible. As this document is oriented around the UI framework and services that are provided by Eclipse, it is a good idea to introduce the main concepts that comprise the Eclipse UI which is presented to DBE Studio users.

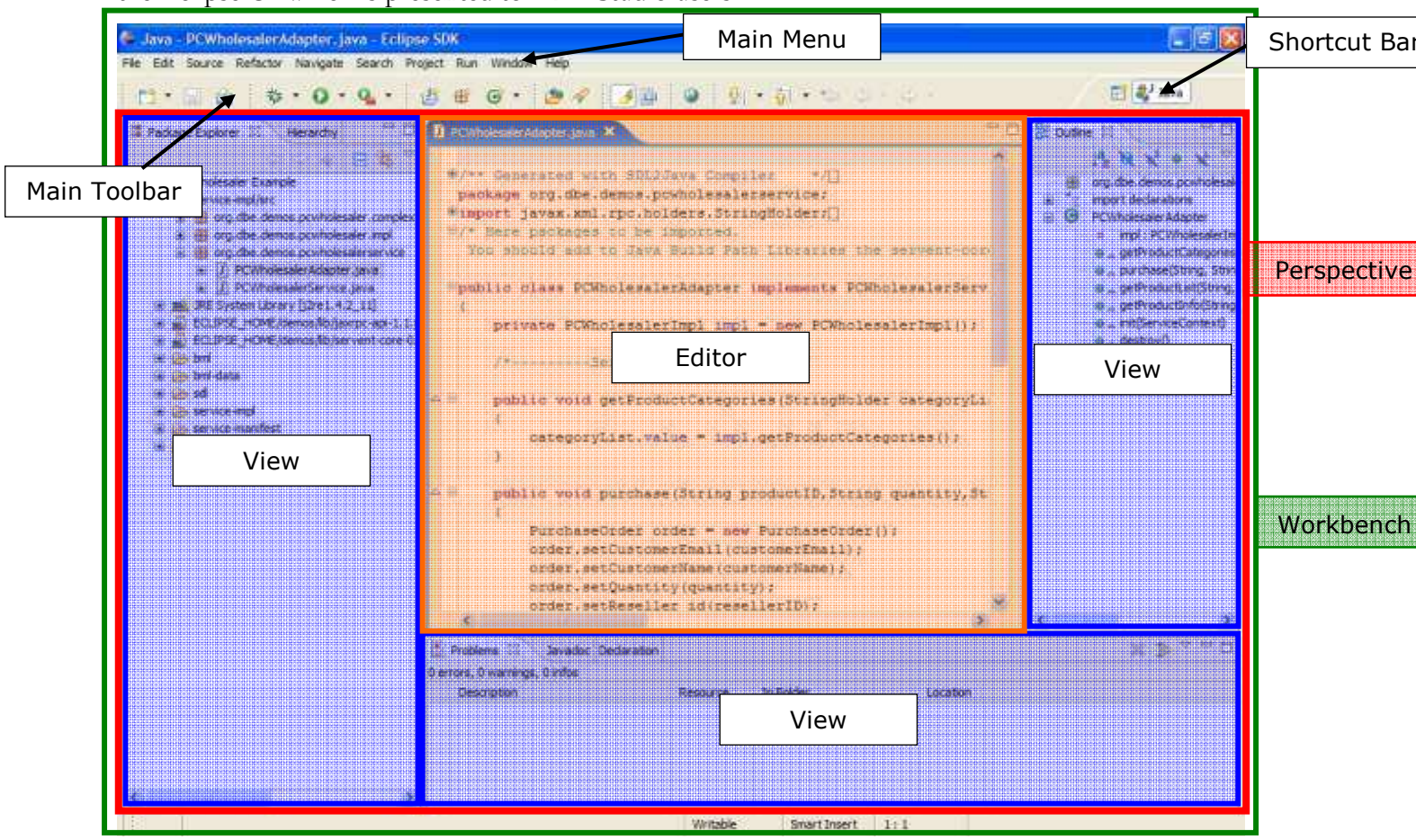


Figure 2 DBE Studio UI Concepts

In the diagram above (Figure 2) the main user interface of the DBE Studio is shown in a manner suitable for Java code development. Around the UI are a number of box overlays. These overlays highlight how the UI is composed. The DBE Studio in its entirety is known as a Workbench and this

is represented by the green overlay in the diagram. Composing the workbench are a main menu, a main toolbar (both pointed to by the arrows) and a perspective. A perspective groups an editor, the orange overlay and its related views, the blue overlays. There can be many perspectives open in a workbench and they can be switched between using the shortcut bar.

1.2 DBE Studio

The DBE Studio is an Integrated Development Environment (IDE) for the Digital Business Ecosystem (DBE) [14]. The DBE is a semantically-aware, service oriented architecture for Small and Medium sized Businesses (SMEs) whose services run within a peer to peer network. The architecture of the DBE is divided into three main logical components:

- Service Factory: This compromises of a set of development tools to allow users create and deploy DBE services and models.
- Execution Environment (ExE): This is the DBE's runtime environment and is the application container for the execution of DBE services. More information can be found at [15].
- Evolutionary Environment (EvE): This is the environment where DBE services, especially composed services, are optimized and evolved using user preferences and service usage data. More information can be found at [16].

The DBE Studio is the Service Factory incarnated and implemented upon the Eclipse platform. In Eclipse, all tools that compose the platform are implemented as Eclipse plugins and correspondingly all DBE Studio tools are implemented in the same fashion.

These plugins developed for the DBE Studio allow business services to be analysed and modeled based on the Business Modeling Language (BML), version 1 [17]. Those modeled services have functional interfaces generated and their corresponding implementation as Java skeletons. With these skeletons, developers, using the standard Java development tools also available with Eclipse, then can add business logic to them and using more DBE Studio plugins publish semantic advertisements and deploy live service within the ExE.

This description of the DBE Studio in fact defines a general workflow within the DBE Studio, which is useful to illustrate in the following diagram (Figure 3).

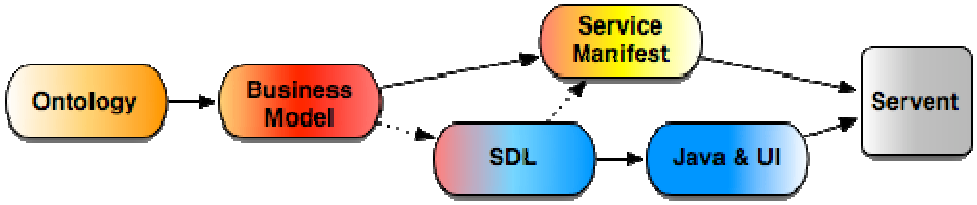


Figure 3 DBE Studio General Workflow

The above illustrated workflow moves from the right to the left. The initial stage of defining an ontology is optional. A more detailed look at this workflow is illustrated below (Figure 4).

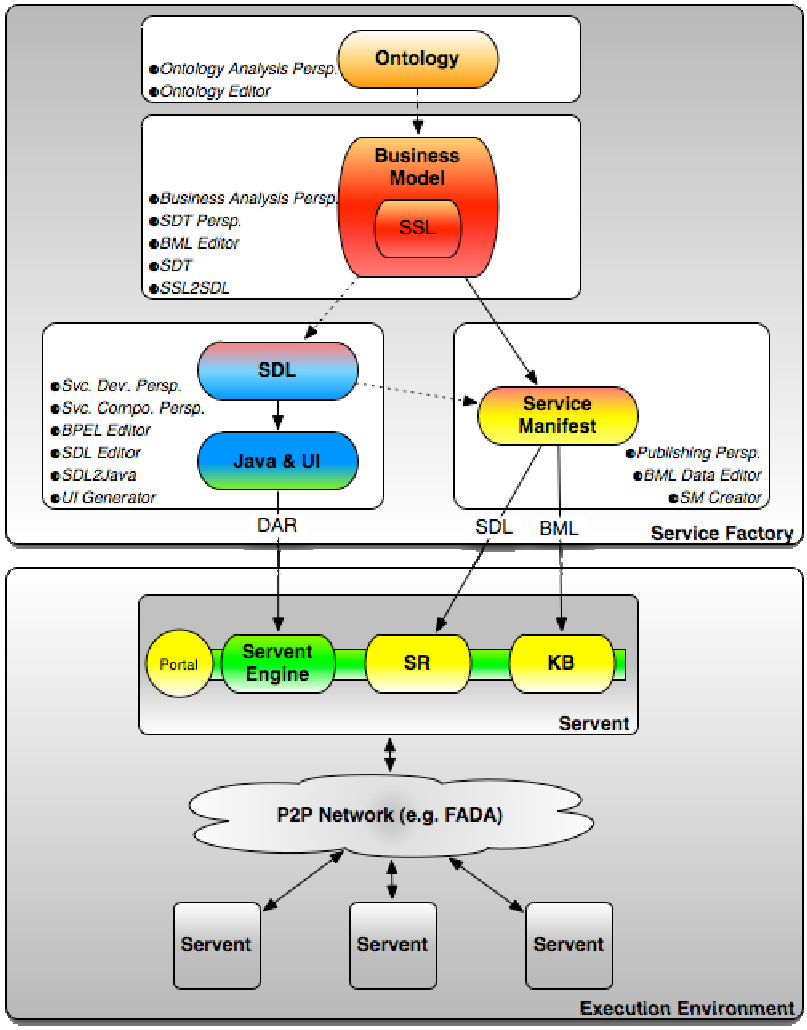


Figure 4 DBE Studio Detailed General Workflow

Typically the target users of the DBE Studio are software developers. These may vary from salient SMEs to 3rd party business analysts who need to model their business and software services using the business modeling tools, to SME software developers or indeed 3rd party developers, who need to expose new software services or existing legacy systems as a software service using the DBE.

2 General DBE Studio-Wide Considerations

Contained in the following section are general considerations that are applicable right throughout the DBE Studio and its plugins.

2.1 Startup and Welcome

When the DBE Studio starts up for the first time, the user should be able to begin working with the Studio, without having first to conquer a steep learning curve. In the Eclipse development environment this is accomplished using a welcome page which briefly shows, graphically, the main features that are available to developers. This welcome page is shown in Figure 5 and shows an easy and terse way to introduce a potentially confusing and overwhelming tool that eclipse could be. With the introduction of eclipse 3.2, it is now possible [23] to easily extend this page to accommodate 3rd party products, such as the DBE Studio. It is a recommendation of this UI evaluation that the DBE Studio takes advantage of this feature in eclipse and implement its own DBE Studio welcome page contribution.

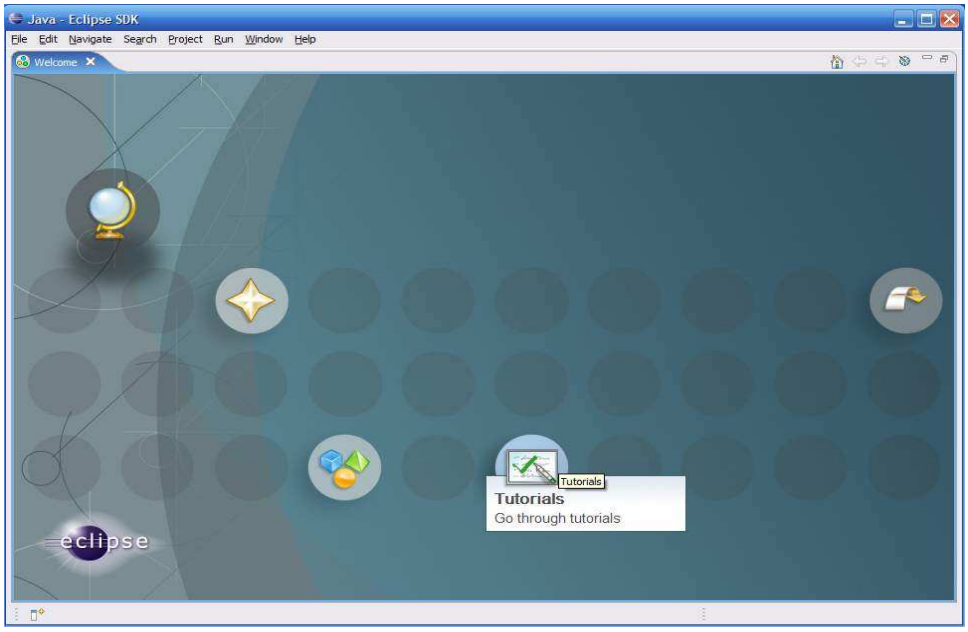


Figure 5 Eclipse Welcome Page

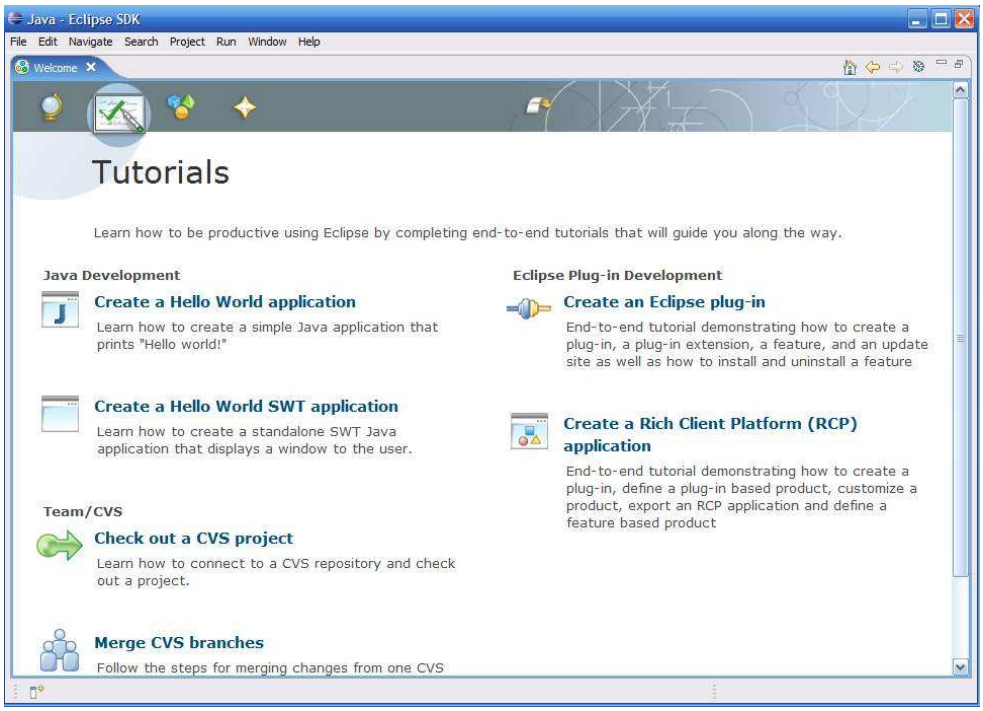


Figure 6 Eclipse Welcome Page: Clicking the main entry shows detailed tutorials

Further to implementing this welcome screen, one of the very first steps that a user has to do is to configure the DBE Studio. Currently this requires the user dismissing the welcome page and navigating a number of menu entries. This may initially, to a new user, be confusing and inconvenient especially when presented by the configuration parameters the DBE Studio has (see section 8). As such a recommendation of this evaluation is to present the bare necessary parameters that need configuration through the use of a simple and explained welcome page. From evaluating the preference pages of the DBE Studio the bare necessary parameters that need to be set are:

- SME ID
- Servent (ExE) URL
- Proxy settings

Based on the above two suggestions, the first time startup of the DBE Studio should follow the following workflow:

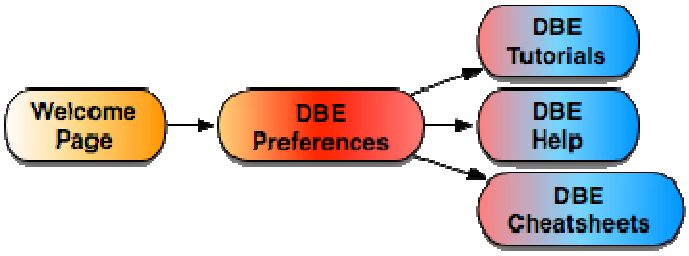


Figure 7 DBE Studio - Initial Startup Workflow

2.2 Error Handling and Logging Management

On evaluating the source code of a sample of DBE plugins it is apparent that there is ample logging code within plugins. However the mechanism for logging within these plugins is only suitable for those developing plugins and does not allow easy inspection of logging messages by anyone else other than the developers of that plugin. The reason for this is that most logging messages are displayed to the user (the developer in this case) through the use of the log4j logging framework [18]. In log4j there are appenders, sinks to output logging messages, and the most commonly used appender is the console appender. This appender will only output messages to a process that has been launched using a console or has access to one. Normally DBE Studio end users will launch the Studio within a graphical window manager, like explorer in Windows or Gnome in Linux. As such these logging messages cannot be viewed to the end user in the case that the end user wants to provide support information on issuing a bug submission. To remedy this problem there are two approaches that can be taken:

- Use the in-built eclipse logging mechanism [21] as a replacement for the log4j mechanism in all DBE Studio plugins.
- Configure log4j in such a way that it uses an appender that is compatible with eclipse, such as the Ganymede plugin [19] or the log4j integration plugin [20].

These two solutions to the above problem should be seriously considered to enable logging to be an effective part in reporting errors back to the DBE Studio community.

2.3 Connection Management

Ultimately the DBE Studio will always interact with the DBE Network through the ExE (Servent) [15]. All dialogs and indeed plugins of the DBE Studio need to account for this in their design. All development within the DBE Studio should be carried out so as not to be dependent on a network connection. That development should initially performed and persisted in the local workspace. To make these resources available to the DBE community, they can be then exported to their respective online repositories. Where a DBE Developer wishes to work with resources that are currently stored in a repository of some sort (Knowledge Base, Semantic Registry, Recommender), the DBE Studio should operate so as to retrieve the requested resources, save those resources to the local workspace and then present them to the user as expected.

There will be the case where not all DBE Developers will share the same network resource-types – some maybe on a high-speed xDSL connection, others on an ISDN line or perhaps a 56K modem. This may result in longer delays for developers, especially when importing a large BML model. To counter this and to provide feedback to the developer, progress meters should always be used, allowing the developer to know where in the process he/she is and allow that user to cancel a network action at any stage.

There should be no automatically initiated connections to the DBE Network either, those connections required should be allowed by the user through the use of an appropriate prompt.

A major deficiency of the DBE Studio was not to provide or at least re-use proxy server settings. Without the provision of such a feature, SME's who must for one reason or another use a proxy server for access information over HTTP were limited.

To orchestrate all of the connections within the DBE Studio so connections are managed easily and existing connections are reused, the creation of a connection manager is recommended. This would be implemented as an Eclipse plugin that all plugins of the DBE Studio requiring a network connection would use. This connection manager is then the only plugin that requires a preference page to configure the servent URL. In doing so would have the knock on effect of making the preference pages of the DBE Studio easier to use through the reduction of the number of fields that need to be configured.

These suggestions have already been accepted and are now introduced in version 0.3.0 of the studio with the inclusion of a connection manager plugin.

2.4 DBE Project Structure and Layout

Each DBE project contains a collection of folders and files. These objects are known as resources. The layout chosen for a DBE project is the result of consultation with all consortium members involved in DBE Studio plugin tasks. The layout of a DBE project is as follows:

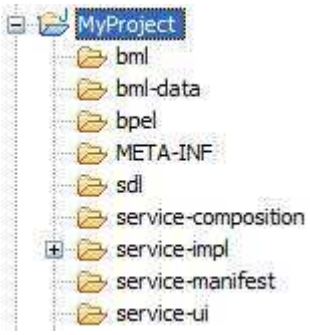


Figure 8 DBE Studio – Project Structure

Considerations that need to be taken into account when designing DBE plugins that manipulate a DBE project are as follows:

- Root folders, i.e. those at the same level of DBE projects, should not be created automatically without the explicit permission of the user.
- Any newly created resource, including any resource imported from an outside source should be stored in the current and active DBE Project folder. If this is not an option or cannot be determined then the process storing the resource should ask the user to specify the location of where the resource is to be stored. This includes the creation of ontologies, BML models and semantic search templates and queries.

2.5 Persistency within the DBE Studio

A great many of the DBE Studio plugins require that the output of their actions be persisted to some form of storage. In all cases these forms of persistency are either:

- Local file system
- Remote storage
  - Knowledge Base
  - Semantic Registry
  - Distributed Storage System

To improve reliability and independence on an external network connection all DBE Studio plugins should always persist their data and output to the local file system.

Persisting to the local file system should always be accomplished using standard eclipse mechanisms. Where editors create and or modify documents they should persist this data by extending the eclipse extension points so that the menu entry "File -> Save…" and "File -> Save As…" are respected. Also the standard eclipse short-cut key combination of "CTRL + S" should also be implemented and respected. In respecting these two persistency mechanisms, user preconceptions on saving resources are respected and confirmed.

When persisting files to the local file system, **all** DBE Studio plugins should respect how a DBE Studio project is laid out. What this means for DBE Studio plugins is that as they know the layout of a DBE Project they should always select the correct location for the output of what they generate and only ask the user for confirmation that the output location is correct. In the case that the expected output location does not exist the DBE Studio plugin should proactively solve the problem by either alerting the user to this fact or actually creating the location. Saving resources should always be performed by the eclipse project-focused file selection dialog. An example of this dialog is shown in Figure 9.

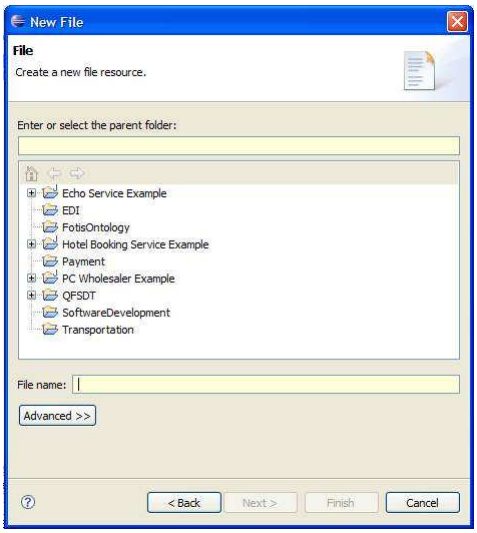


Figure 9 DBE Studio – Standard Project-specific File Saving Dialog

With regard to saving resources to a remote system, such as those listed above, this should always be performed using the eclipse metaphor of "Exporting" a resource. Never should a DBE Studio plugin replace this metaphor with those used to save files locally e.g. overriding the <CTRL + S> short-cut key combination to save a resource locally. Exporting resources is performed by right clicking on the resource to be exported and then selecting "Export…" from the menu. The DBE Studio plugin should then have all the information (file type, file name, file path etc) to be able to carry out this operation in a manner that users have and will come to expect from the eclipse environment.

2.6 Long Running Operations within the DBE Studio

Within the DBE Studio there are a number of operations and tasks that can take up a noticeable period of time that is easily perceived by the user. In cases such as this there are two necessary mechanisms to **always** abide by.

The first is that if the operation, which is estimated by the developer to complete in a perceivable time frame that would be considered long, then that operation should be implemented in an asynchronous fashion so as not to interrupt the multi-tasking nature of a user interface.

The second and related mechanism to adopt is that for all long running user-initiated operation within the user interface of the DBE Studio, progress and feedback should be supplied to the user. This is normally done with the use of progress bars that display the amount of an operation

completed and still left to do. This is vitally important in order that the user remains in control and informed about his/her initiated operations and that there is visual feedback supplied telling the user that the operation is still working.

2.7 Capitalization

For consistent capitalization of components that provide a textual description to the user, the guidelines set out in [1] are to be adhered to. By following these guidelines, a more polished feel and coherent experience can be presented to the user. There are two main styles of capitalization to be used within the DBE Studio as follows:

- Headline capitalization – this style should be applied to all titles, menus, tooltip, tabs, and push buttons within dialogs and windows. The first and last words, and all nouns, pronouns, adjectives, verbs and adverbs should be capitalized. Do not include ending punctuation. An example of this style is shown below:

  "This is Headline-Style Capitalization"

- Sentence style capitalization – this style should be applied to all check boxes, radio buttons, and group labels. The first letter of the first word should be capitalized and ending punctuation should be included. An example of this style is shown below:

  "This is sentence-style capitalization."

3 Perspectives

In this chapter the perspectives contained within the DBE Studio are analyzed and recommendations are made upon that analysis. A perspective is a visual container that aggregates a set of related views and content editors. When two or more perspectives have the same view opened, that view is the same instance. When there is more than one workbench open neither views or editors are shared.

Within the DBE Studio there are six perspectives currently defined that group related views and editors together. Below is a table of all the perspectives contained in the DBE Studio along with a brief description of each.

Name	Function	Partner
Ontology Analysis	Groups all related views and editors related to ontology modeling activities.	Technical University of Crete
Business Analysis	Groups all related views and editors related to BML modeling activities.	Technical University of Crete
Semantic Discovery	Groups all related views and editors related to service and model query activities.	Technical University of Crete
Service Composition	Groups all related view and editors related to BPEL modeling activities.	Trinity College Dublin
Service Development	Groups all related views and editors related to DBE service implementation activities.	Soluta
Service Publishing	Groups all related views and editors related to service publishing and deploying activities.	Intel Ireland Ltd.

Table 1 DBE Studio Perspectives

What are listed in the above table (Table 1) are the DBE Studio perspectives that are to be considered for evaluation in this document. These very same perspectives can be accessed within the DBE Studio through the menu "Windows->Open Perspectives->Other…" which is illustrated below:

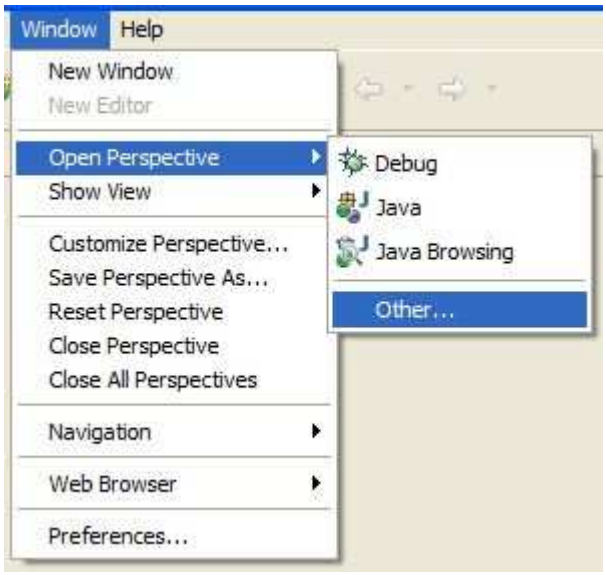


Figure 10 DBE Studio Perspectives - Access

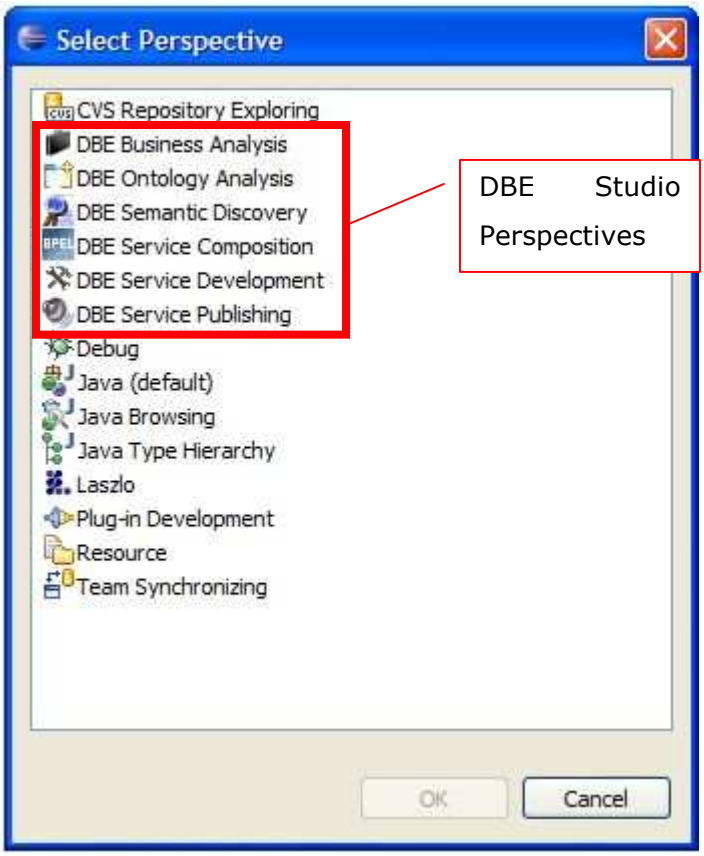


Figure 11 DBE Studio Perspectives - Selection

Before considering each individual perspective the general Eclipse recommendations are detailed which should always be considered when designing and implementing Eclipse perspectives.

3.1 UI Recommendations

When designing perspectives and defining what will be grouped by them, it is prudent to consult [1] for guidelines. These guidelines are found in [1] and expanded upon there and also form the basis for the evaluation of DBE perspectives. Due to their extensive nature, it is useful to summarize them for reference:

Ref.	Description
PG1	Create a new perspective type for long lived tasks, which involve the performance of smaller, non-modal tasks.
PG2	If you just want to expose a single view, or two, extend an existing perspective type.
PG3	The size and position of each view in a perspective should be defined in a reasonable manner, such that the user can resize or move a view if they desire it. When defining the initial layout, it is important to consider the overall flow between the views (and editors) in the perspective.
PG4	If a perspective has just one part, it may be better suited as a view or editor.

Table 2 Recommended Perspective Guidelines

These guidelines are used throughout this section on the evaluation of DBE Studio perspectives and the above table will serve as a reference point for this evaluation and also developers creating or refactoring DBE Studio plugins.

3.2 Ontology Analysis

The DBE Studio ontology analysis perspective groups views and editors that are related to creating, importing, editing and exporting ontologies that are used in the BML modeling tools (available in the DBE Studio Business Analysis perspective). Below is a screenshot (Figure 12) of how the DBE Studio ontology analysis perspective appears once it has been opened by the user.

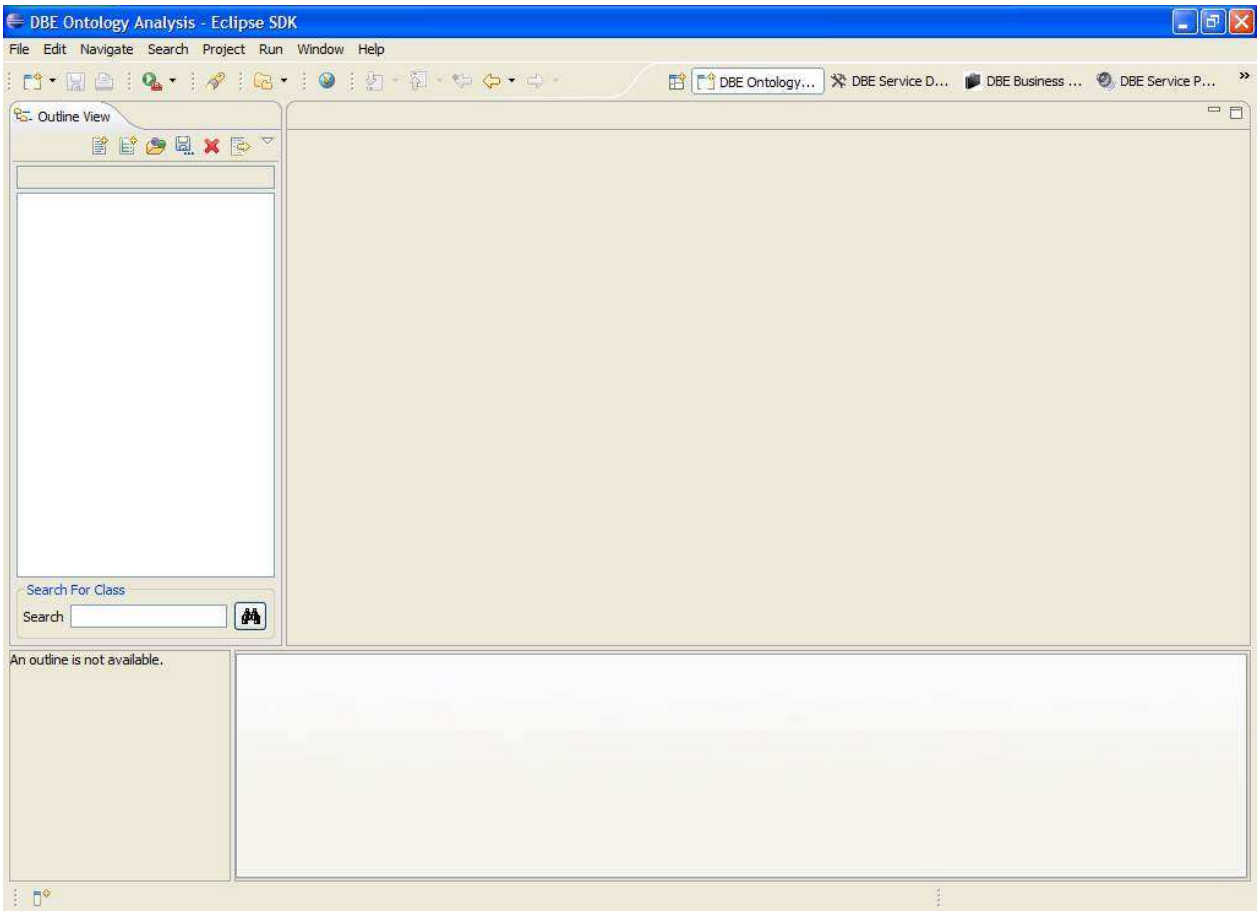


Figure 12 DBE Studio Ontology Analysis Perspective – Initial Presentation

With the DBE Studio ontology perspective visually introduced, an evaluation of it is now presented:

- There are no shortcuts available within the "File->New" menu as illustrated below (Figure 13). An entry "Ontology" and "Diagram" should be inserted.

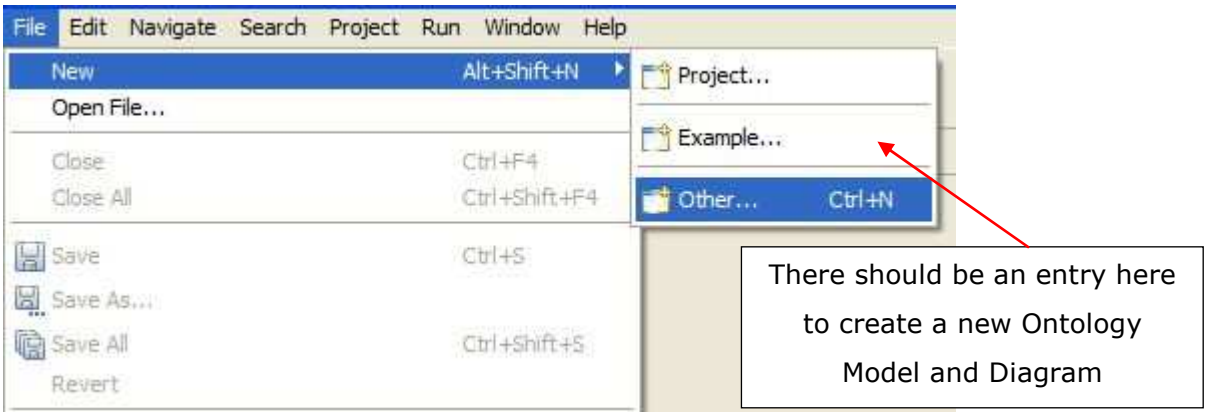


Figure 13 DBE Studio Ontology Analysis Perspective – New Menu Entries

- There are no view shortcuts available within the "new" menu as illustrated below (Figure 14). This should be corrected to show the views "Outline View" and "Properties".

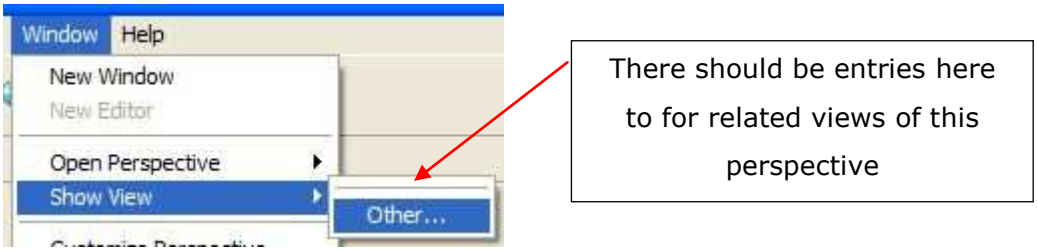


Figure 14 DBE Studio Ontology Analysis Perspective – View menu

3.3 Business Analysis

The DBE Studio Business Analysis perspective groups views and editors that are related to creating, importing, editing and exporting BML models that are used to model a SME's business. Below is a screenshot (Figure 15) of how the DBE Studio Business Analysis perspective appears once it has been opened by the user.

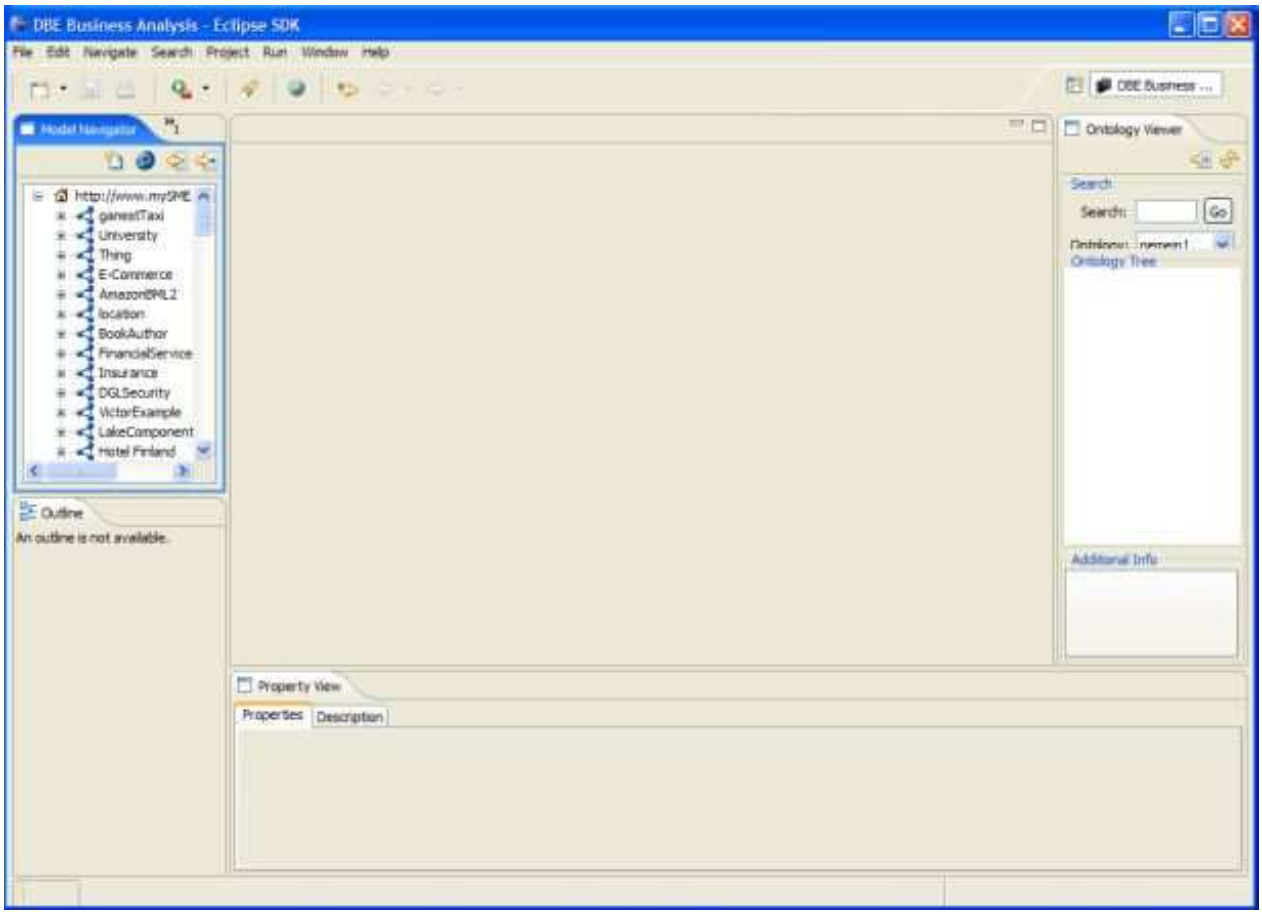


Figure 15 DBE Studio Business Analysis Perspective – Initial Presentation

With the DBE Studio business analysis perspective visually introduced, an evaluation of it is now presented:

- There are no shortcuts available within the "File->New" menu as illustrated below (Figure 16). An entry "BML Model" should be inserted.

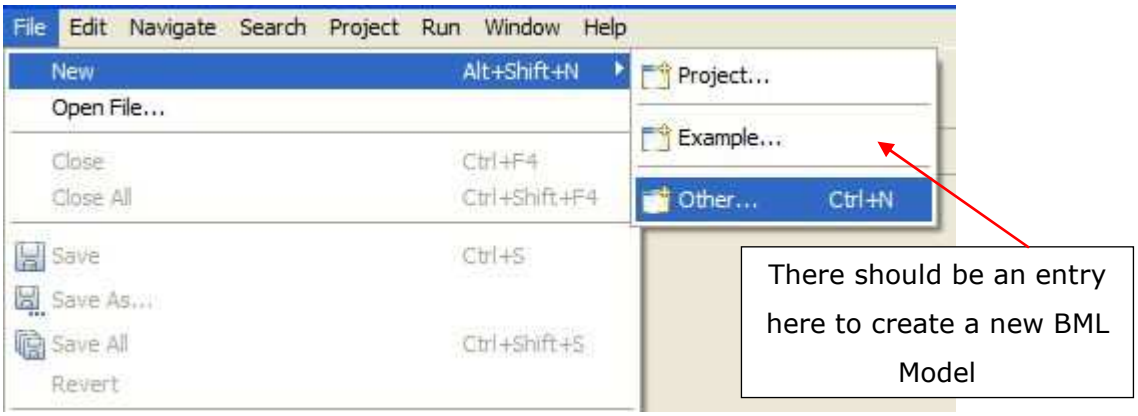


Figure 16 DBE Studio Business Analysis Perspective – New Model Menu Entry

There are issues of connection handling. These are dealt with in the following section, 3.3.1.

3.3.1 Connection Management

Further to section 2.3, there are particular instances of connectivity issues in this perspective. Before the warning (Figure 17) is shown to the user, the entire Eclipse UI is disabled by a modal dialog (Figure 18) until the connection attempt fails. Even when the Business Analysis perspective is successfully connecting to the KB, it disables the UI until such time that the connection is made and populated with the list of models in the Model Navigator view. This should be corrected by providing feedback to the user so the user knows that the Studio is in the process of connecting to a KB.

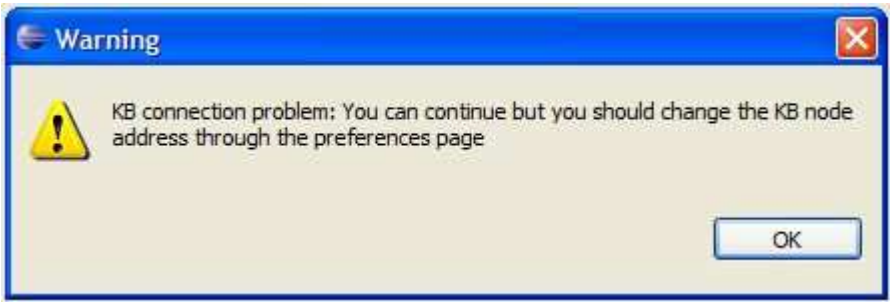


Figure 17 DBE Studio Business Analysis Perspective – Connection Handling

After dismissing the warning dialog box, even though an attempt was already initiated and failed, a second attempt is started to connect to the KB

Figure 18 DBE Studio Business Analysis Perspective – Second connection attempt

One can assume that if the initial connection has failed so too will the next attempt. As such this connection dialog box should not be presented to the user after the initial connection attempt. After dismissing the second connection attempt dialog box the business analysis is presented to the user, as shown below:

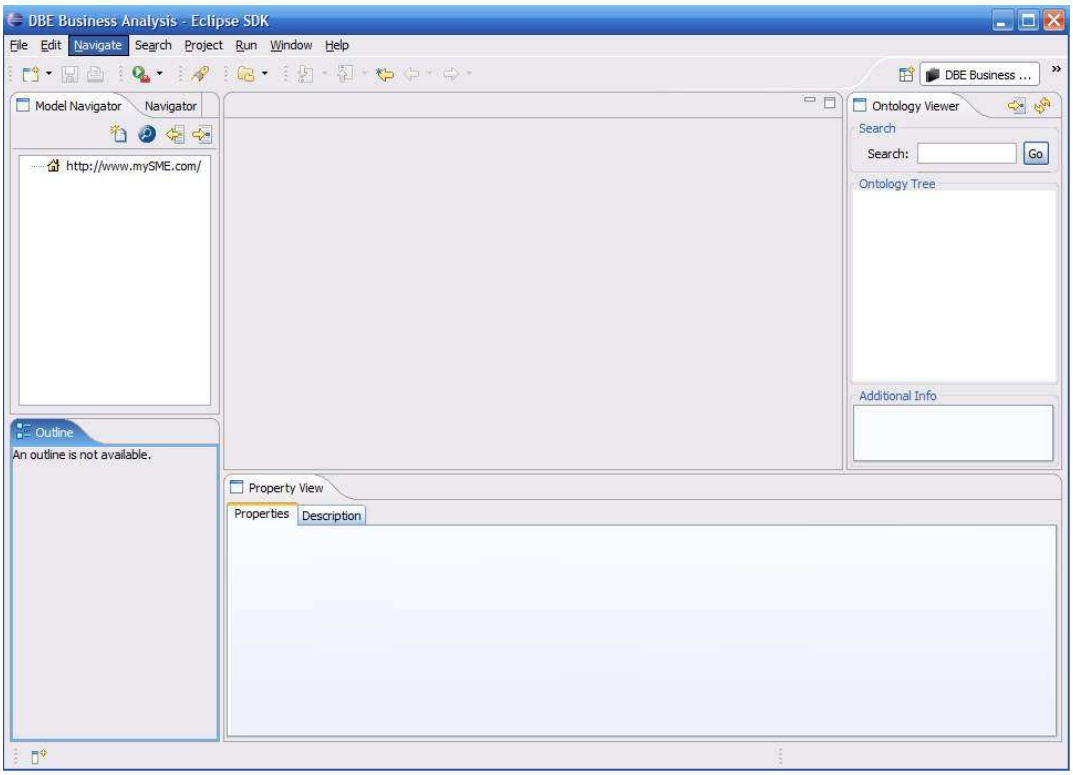


Figure 19 DBE Studio Business Analysis Perspective – No connection present.

One recommendation that is made is to provide an obvious means to allow the user to initiate the connection to the KB.

It is expected that these issues will be resolved by adopting approaches as discussed in section 2.3 and section 2.6.

3.4 Semantic Discovery

The DBE Studio Semantic Discovery perspective groups views and editors that are related to creating and editing of queries and query templates that are used to search for either BML Models or DBE services. Below is a screenshot of how the DBE Studio Semantic Discovery perspective appears once it has been opened by the user.

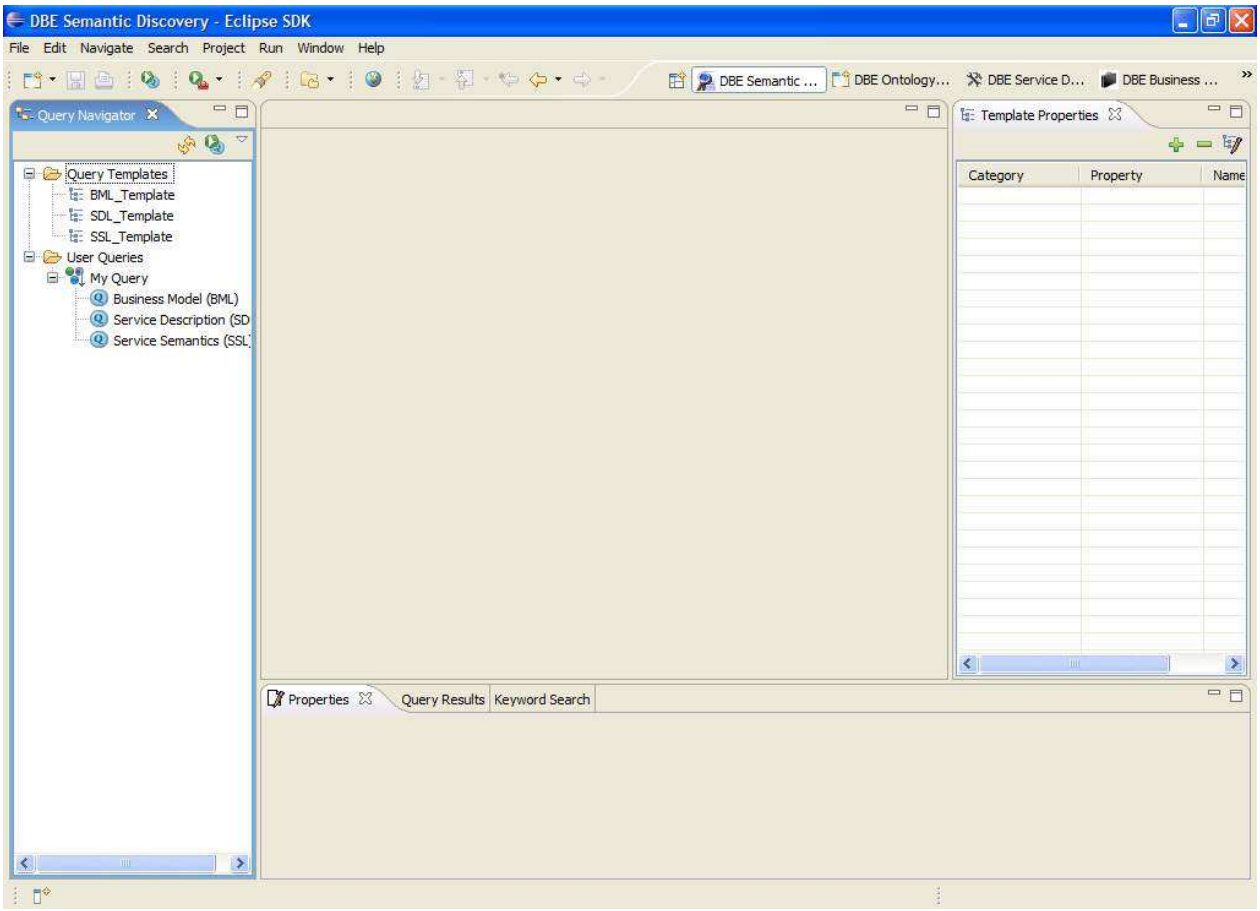


Figure 20 DBE Studio Semantic Discovery Perspective – Initial Presentation

With the DBE Studio Semantic Discovery perspective visually introduced, an evaluation of it is now presented:

- The semantic perspective is perhaps the most over looked perspective in the DBE Studio. One of the reasons for this is that it lacks the means to "bootstrap" itself. What is suggested so that the semantic discovery tools are not overlooked is that a number of example specific queries are supplied. Currently there are only very broad and generic examples supplied with the tools. Perhaps if the DBE Studio example plugin [23] was to include relevant queries for each example project this could supplement and add relevance to the generic examples.

- Another recommendation of this evaluation would be to position the query template view above the area where the query editor would appear. Currently there is a weak cognitive and visual connection between the query editor and query template editor. By placing the query template view above the query editor this would reinforce the fact that the query editor is dependent on the attributes defined in the query template and strengthen the forming of a correct mental model in the users mind.

3.5 Service Composition

The DBE Studio Service Composition perspective groups views and editors that are related to creating and editing of service workflows using BPEL [25] that define a service composition. Below is a screenshot of how the DBE Studio Service Composition perspective appears once it has been opened by the user.

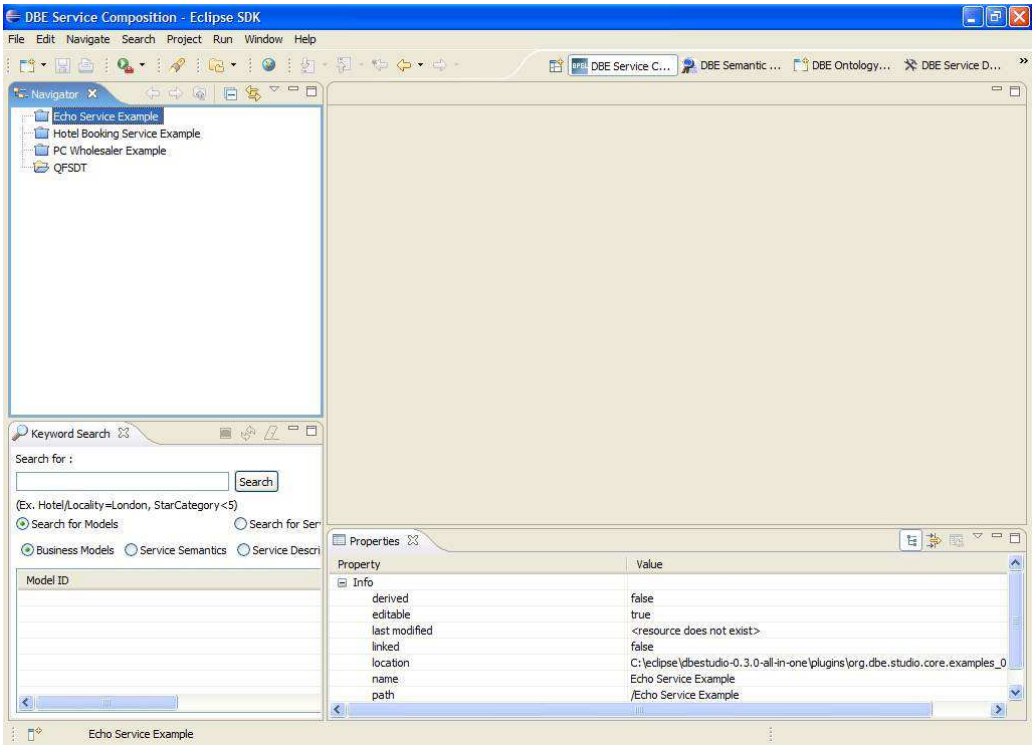


Figure 21 DBE Studio Service Composition Perspective – Initial Presentation

With the DBE Studio Service Composition perspective visually introduced, an evaluation of it is now presented:

- There are no view shortcuts available within the "Window -> Show View" menu as illustrated below (Figure 22). This should be corrected to show the views "Navigator", "Keyword Search" and "Properties".

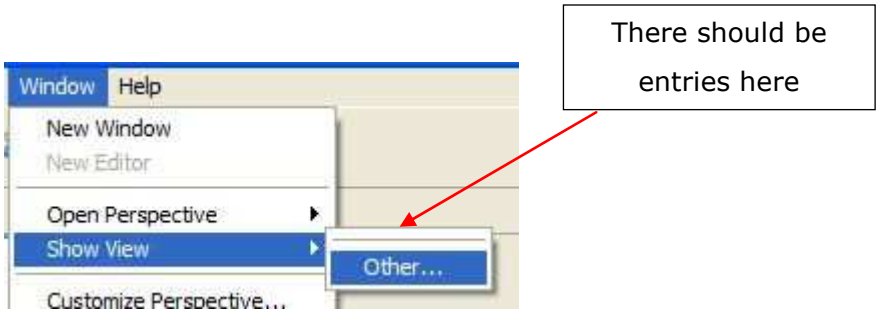


Figure 22 DBE Studio Service Composition Perspective – View Menu

3.6 Service Development

The DBE Studio Service Development perspective groups views and editors that are related to creating and editing of service implementation using the Java language. Below is a screenshot of how the DBE Studio Service Development perspective appears once it has been opened by the user.

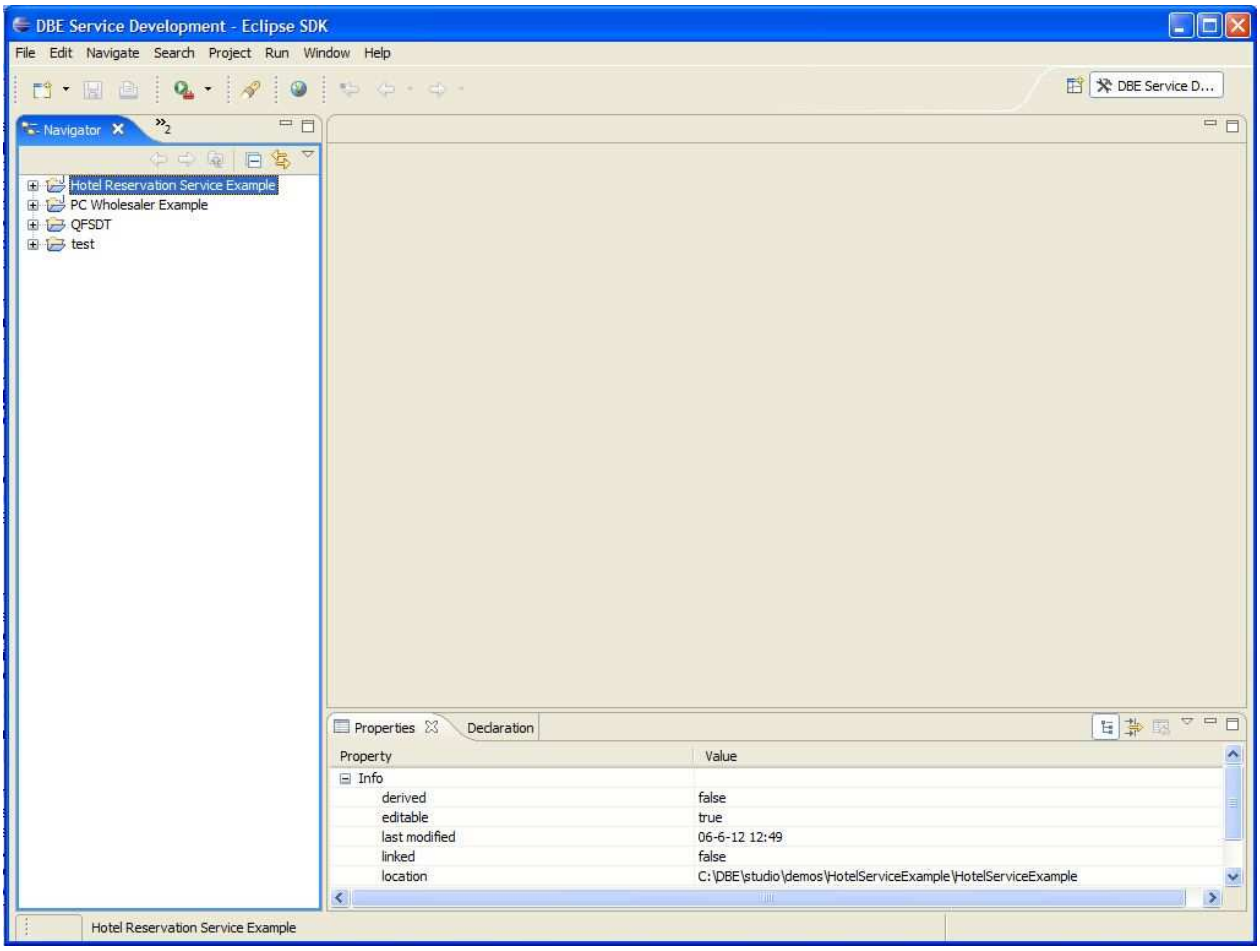


Figure 23 DBE Studio Service Development Perspective – Initial Presentation

With the DBE Studio Service Development perspective visually introduced, an evaluation of it is now presented:

- There are no view shortcuts available within the "Window -> Show View" menu as illustrated below (Figure 24). This should be corrected to show the views "Package Explorer", "Hierarchy", "Problems", "Javadoc" and "Declaration".

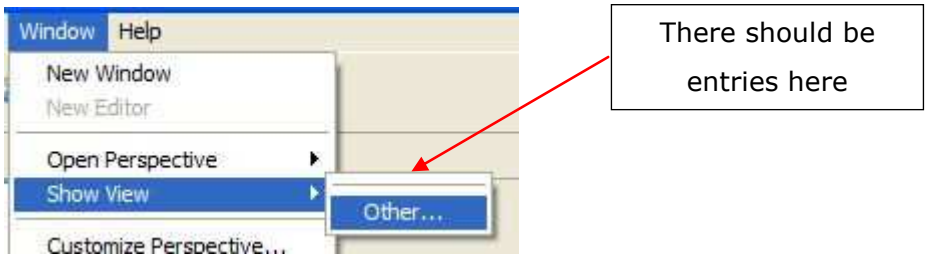


Figure 24 DBE Studio Service Development Perspective – View Menu

3.7 Service Publishing

The DBE Studio Service Publishing perspective groups views and editors that are related to creating and editing of service manifests and allows the population of BML Models with BML data. Below is a screenshot of how the DBE Studio Service Publishing perspective appears once it has been opened by the user.

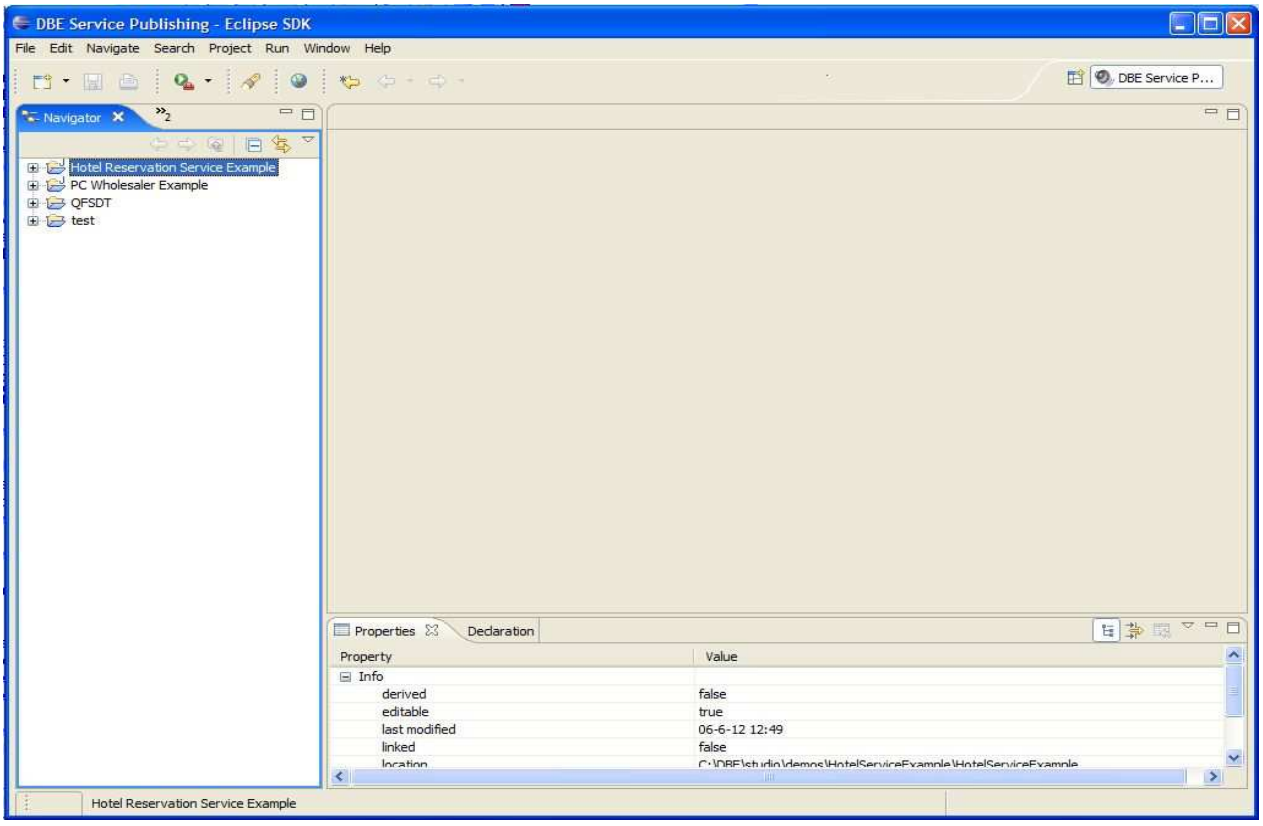


Figure 25 DBE Studio Service Publishing Perspective – Initial Presentation

With the DBE Studio Service Publishing perspective visually introduced, an evaluation of it is now presented:

- There are no view shortcuts available within the "Show View" menu as illustrated below (Figure 26). This should be corrected to show the views "Navigator", and "Properties".

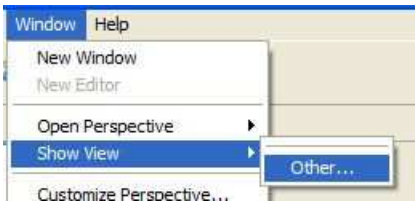


Figure 26 DBE Studio Service Publishing Perspective – View Menu

3.8 General Comments

There are two perspectives that open without any obvious functionality to the user. In this case the user will usually find him/herself searching through menus and contextual menus to find the functionality of a perspective. This can be frustrating and time-consuming especially for users new to the DBE Studio.

A practical solution to resolve this problem would be for a cheat sheet describing the functionality of each perspective to be opened when each is opened. The user then could have the functionality of the perspective presented through the interactive fashion that cheat-sheets offer. In order that users who are already familiar with the DBE Studio's perspectives are not distracted by this feature enabling and disabling this feature should be user selectable from within the cheat sheets. The user's choice would then be stored in the DBE Studio's preferences.

4 Editors

In this chapter the editors contained within the DBE Studio are analyzed and recommendations are made upon that analysis. Within the DBE Studio there are seven editors currently defined that edit resources. Below is a table of all the editors contained in the DBE Studio along with the partner responsible for implementation.

Name	Partner
Ontology Editor	Technical University of Crete
BML Editor	Technical University of Crete
BML Data Editor	University of Central England
Query Editor	Technical University of Crete
SDL Editor	Soluta
BPEL Editor	Trinity College Dublin
PDD Editor	Trinity College Dublin
Service Manifest Editor	Soluta

Table 3 DBE Studio Editors

4.1 UI Recommendations

When designing editors, it is prudent to consult the Eclipse human interface guidelines (HIG) [1]. These guidelines are found in [1] and expanded upon there and also form the basis for the evaluation of DBE editors. Hence, it is useful to summarize them for reference:

Ref.	Description
EG1	Use an editor to edit or browse a file, document, or other primary content.
EG2	Modifications made in an editor must follow an open-save-close lifecycle model.
EG3	Only one instance of an editor may exist, for each editor input, within a perspective.

EG4	It must be possible to open a separate instance of an editor for each different input.
EG5	The editor should be labeled with the name of the file, document, or input being edited.
EG6	In multi-page editors, use a tab control for page activation. Tab labels should be kept to one word, and two words at most.
EG7	All of the commands, except for the obvious commands, available in the editor should be added to the window menu bar.
EG8	Use the standard format for editor contributions in the window menu bar.
EG9	If an editor has support for Cut, Copy, Paste, or any of the global commands, these commands must be executable from the same commands in the window menu bar and toolbar.
EG10	Fill the editor toolbar with the most commonly used items in the view menu.
EG11	Fill the context menu with selection oriented commands.
EG12	Use the standard format for editor context menus.
EG13	Fill the context menu with a fixed set of commands for each selection type, and then enable or disable each to reflect the selection state.
EG14	Register all context menus in the editor with the platform.
EG15	Implement a Command Filter for each object type in the editor.
EG16	If the input to an editor is deleted, and the editor contains no changes, the editor should be closed.
EG17	If the input to an editor is deleted, and the editor contains changes, the editor should give the user a chance to save their changes to another location, and then close.
EG18	If the resource is dirty, prefix the resource name presented in the editor tab with an asterisk.
EG19	Treat read-only editor input as you would any other input. Enable the Save As if possible. Display "Read-only" in the status bar area.
EG20	If the data within an editor is too extensive to see on a single screen, and will yield a structured outline, the editor should provide an outline model to the Outline view.
EG21	Notification about location between an editor and the Outline view should be two-way. A context menu should be available in the Outline view as appropriate.

EG22	An error or warning image should be added to items with the error or warning respectively. A container should have a red X if it there are errors on the container itself, a gray X if any of its descendents have errors (but not the container itself), and no X if neither the container nor any of its descendents have errors.
EG23	If appropriate, implement the "Add Task" feature in your editor.
EG24	If appropriate, implement the "Add Bookmark" feature in your editor.
EG25	Editors with source lines of text should show the current line and optionally column numbers on the status line. It's optional for the editor to show line numbers for each line in the editor itself.
EG26	Table cell editors should support the single-click activation model, and in edit mode, they should render complex controls upon single-click.
EG27	Changes made in a table cell editor should be committed when a user clicks off the cell or hits the "Enter" key. Selection should be cancelled when user hits the "Esc" key. First letter navigation should be supported as a cursoring mechanism within a cell.
EG28	When performing fine-grain error validation in an editor, use red squiggles to underline the invalid content. When users move the mouse over the red squiggles, display the error text in a fly-over pop up box.
EG29	Use the Task view to show errors found when the Save command is invoked.
EG30	If modifications to a resource are made outside of the workbench, users should be prompted to either override the changes made outside of the workbench, or back out of the Save operation when the Save command is invoked in the editor.

4.2 Ontology Editor

The Ontology editor allows users to create, edit and save ontologies. This editor displays the modeled ontology in a graph presentation. The modeling is supported by a palette that contains tools to interact with the model and drawing components that make up a model. What follows is an evaluation of the editor as shown in Figure 27.

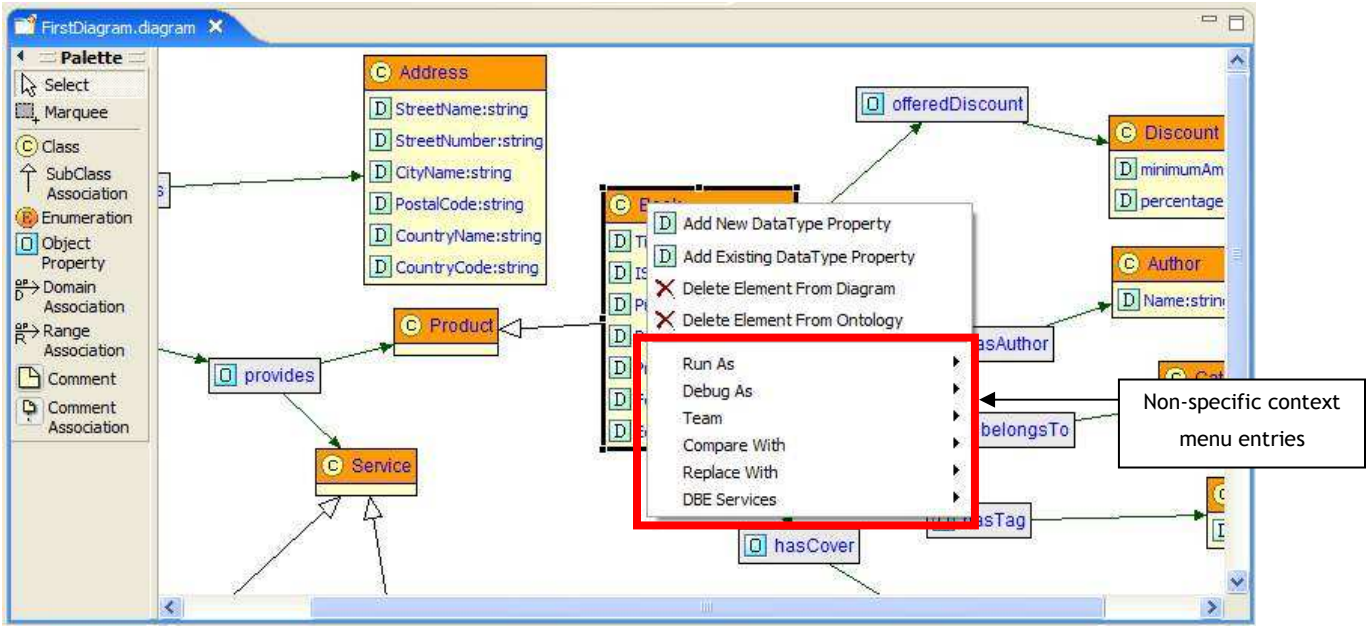


Figure 27 DBE Studio Editors – Ontology Editor

With the DBE Studio BML data editor visually introduced, an evaluation of it is now presented:

- The palette in left of the editor does not support customization, even though the facilities are available when right-clicking on the palette. Most noticeably the palette disobeys the hiding preferences such as "always displayed" and "automatically hide".
- When an existing ontology is opened and the user decides to open another, instead of opening the new model in a new editor window, the existing model is closed and the new model is opened in its place. This is unexpected behavior. It is recommended that the second model is opened in a new editor window to conform with EG4.
- The context menu shown in Figure 27 contains functionality that is not specific to the content contained in the editor. These should be removed as it does not conform to EG11.
- There are also a number of issues associated with how the ontology editor persists its data. These issues are discussed in section 2.4 and section 2.5 and the suggestions made in these sections should be considered.

Accompanying the ontology editor is a supporting view (Figure 28) that gives the overview of the entire model should it not fit in the viewport of the editor. This is a welcome addition as it provides

the user with a spatial awareness of his/her model. However the manner in which it is currently implemented does not allow it to be moved and as a result, this evaluation recommends that it be modified so it can be moved at the discretion of the user.

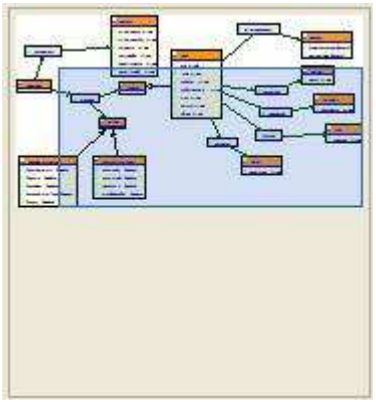


Figure 28 DBE Studio Editors – Ontology Editor Overview

4.3 BML Editor

The BML editor allows users to create, edit and save BML models. This editor displays the modeled BML in a graph presentation. The modeling is supported by a palette that contains tools to interact with the model and drawing components that make up a model. This editor is, due to the complexity of BML, comprised of a number of sub-editors. For the purpose of this evaluation it was decided that only the first two sub-editors be evaluated as the remainder so not seem to be part of the current workflow within the DBE Studio. As the other tabs (Process, Event, Location and Motivation) do not currently form part of the workflow within the DBE Studio, it is wise to hide them in order to reduce the visual interference they may cause to a user, even though their titles hint at possibly useful additions to the DBE Studio. Another reason to hide these sub-editors is that there is no documentation supporting these sub-editors. A preference could be set in the BML Editor preference page dictating whether or not these tabs are displayed to the user.

What follows is an evaluation of the two sub-editors the Semantic Description Editor (Figure 29) and the Business Organization Editor (Figure 30).

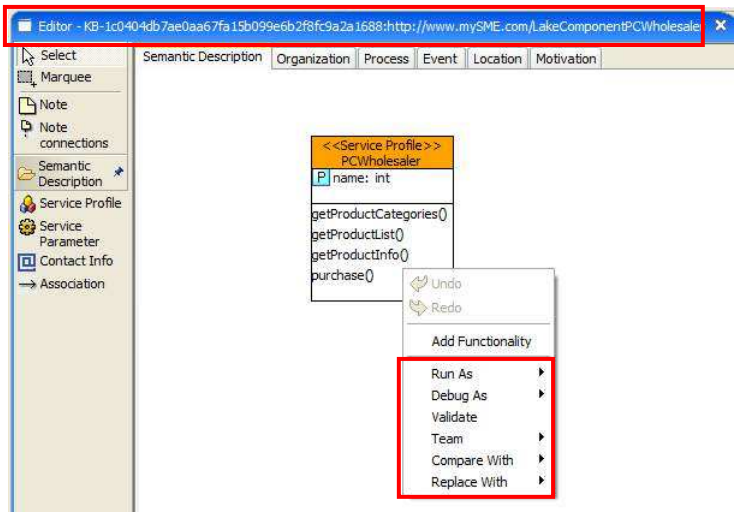


Figure 29 DBE Studio Editors- Semantic Description Editor

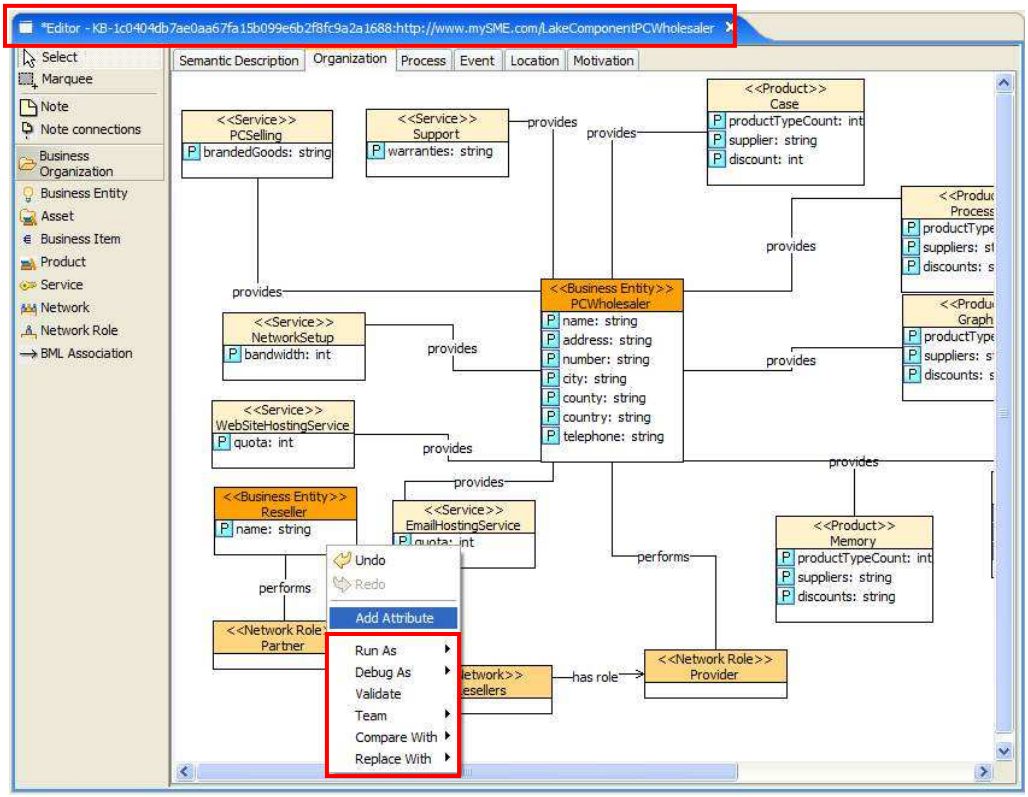


Figure 30 DBE Studio Editors- Business Organization Editor

With the two BML sub-editors visually introduced, the findings of the evaluation on both are presented together.

- Unlike the ontology editor, there is no way to zoom in and out on the model shown in the editor's viewport. This functionality should be exposed so as to compliment the BML editor overview view.
- When a new BML model is opened while an existing model is currently opened the existing model is closed and replaced with the newly requested model. This does not conform with EG4.
- When switching from the Business Analysis perspective to another perspective, the BML editor is closed without any warning to the user. This is unexpected behavior and measures should be taken so this does not occur in future versions of the BML editor.
- The title of the editor is often too long to be read. This should be shortened with only the most relevant information shown in the title.
- The first part of the editor's title should always contain the name of the editor. In this case the name is "Editor". This should be renamed to "BML Editor".
- The editor's panel cannot be hidden by the user. This should be allowed as it will increase the maximum workspace a user has to create BML models in. This is very much a consideration for users that may not have devices capable of displaying a high resolution.
- The orientation of editor sub-tabs should be re-oriented so their location is at the bottom of the editor, which is the normal and expected appearance with eclipse multi-tab editors.
- If a user is to right-click on a diagram object title (e.g. business entity) the user should be presented with a list of all possible actions that can be performed on that object (e.g. "add functionality" and "add attribute").
- The content in both context menus shown above contain entries that are not specific to BML modeling and should be removed. This does not conform with EG11.
- There are also a number of issues associated with how the ontology editor persists its data. These issues are discussed in section 2.4 and section 2.5 and the suggestions made in these sections should be considered.
- When a BML model is saved to the local file system, 2 files are generated. The first .bml contains serialized information relating how diagramming objects are laid out on screen. The

second is a .xmi file which contains the model definition. It is recommended that the .bml file be hidden in some way, out of the view of the user. Having two files present will add confusion when a user decides that they want to perform a subsequent action upon a BML model.

- There is no way to open a model persisted on the local file system even though when double clicking on a .bml file the BML editor is opened. Ideally this should be considered.

4.4 BML Data Editor

The BML data editor allows users to create, edit and save BML data files. This editor displays the BML data in a hierarchical tree presentation, with the data represented by nodes editable in the lower half of the editor. What follows is an evaluation of the editor as shown in Figure 31.

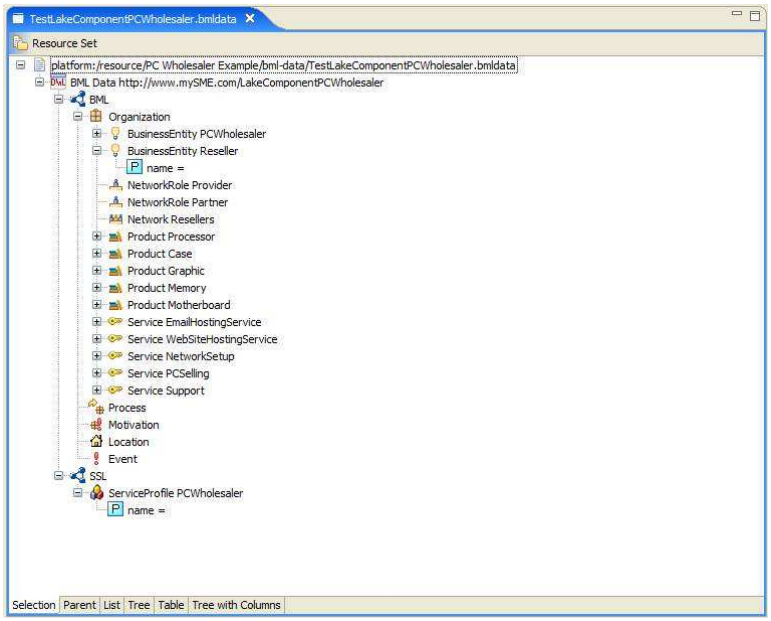


Figure 31 DBE Studio Editors – BML Data Editor

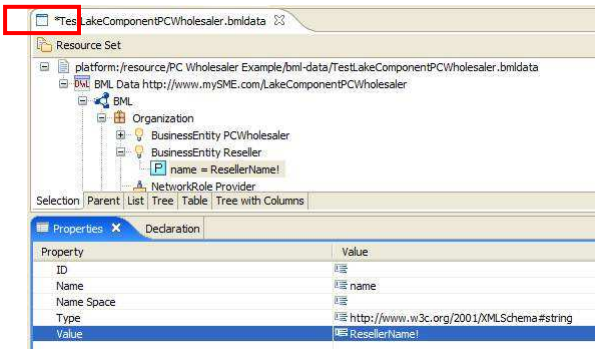


Figure 32 DBE Studio Editors – BML Data Editor, Editing a value.

With the DBE Studio BML data editor visually introduced, an evaluation of it is now presented:

- The tabs at the bottom of the BML Data editor should be removed as these are not necessary for the operation of the editor.
- The editor is set as the default editor for files ending with .bmldata, however the entry in the open menu is not named accordingly with a suitable icon as shown in Figure 33.

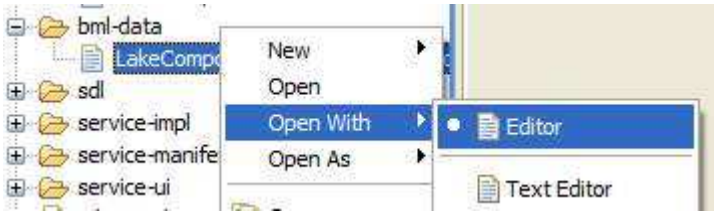


Figure 33 DBE Studio Editors – BML Data Editor Open Menu

- The top tab (highlighted in Figure 32) should have a representative icon and not a generic icon.
- The BML Data editor creates two files when it persists its data. Just like the BML Editor this can lead to confusion. Either only the correct file is outputted or if that is not possible then a filtering mechanism should be investigated and implemented.
- The BML Data editor only allows the creation of BML Data files that are solely based upon BML models persisted in the KB. This goes against recommendations made in section 2.5.

This editor should be able to create BML Data files from BML models that currently reside inside the user's current workspace.

4.5 Query Editor

The query editor allows users to edit and save semantic queries. This editor displays the query parameters of the query in a grid editor that allows inline editing of the parameter fields. What follows is an evaluation of the editor as shown in Figure 34.

Figure 34 DBE Studio Editors – Query Editor

With the DBE Studio Query editor visually introduced, an evaluation of it is now presented:

- What may not be immediately obvious to a user of this editor is that it is in fact a multi-dimensional editor. Values that run from left to right are composed in the query with the AND operator. Values that run from top to bottom are composed in the query with the OR operator. It is recommended that this powerful feature be highlighted if possible, perhaps by providing a cheat sheet that brings the user through the process of creating a query using a template and that explains the multi-dimensionality of the query editor.
- The query editor operates as expected. However some direction should be given to the user on how cells containing parameters can be edited. These directions could be supplied in help documentation.

4.6 SDL Editor

The SDL editor allows users to create, edit and save SDL files. This editor displays the SDL model in a hierarchical tree presentation and data is edited using the standard Eclipse property view. What follows is an evaluation of the editor as shown in Figure 35.

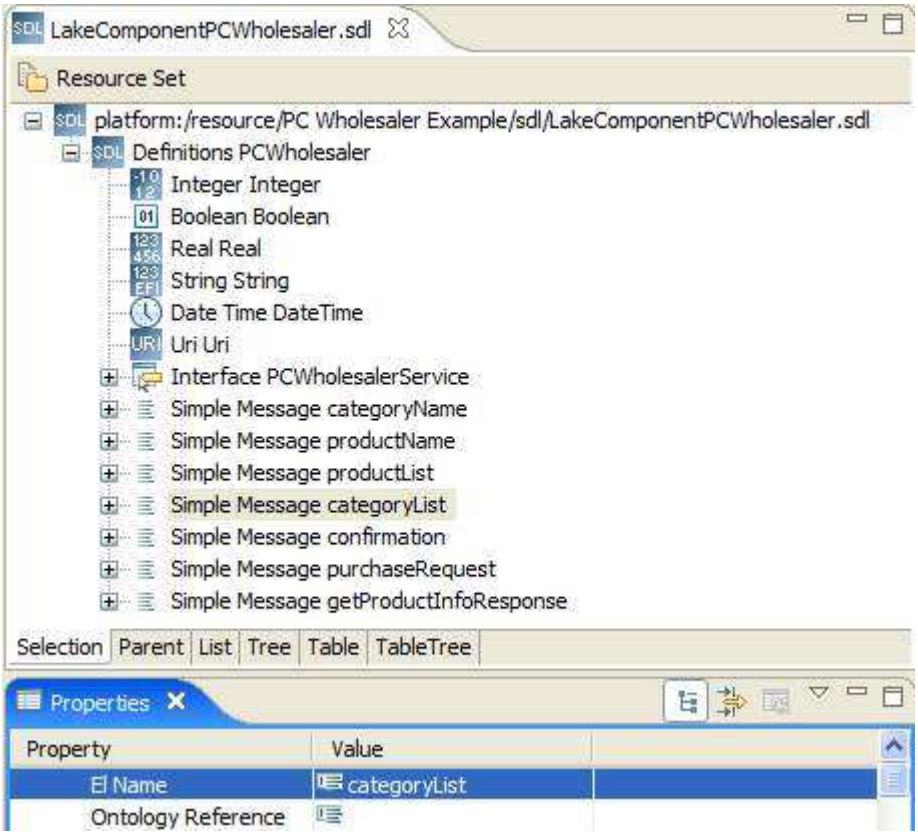


Figure 35 DBE Studio Editors – SDL Editor

With the DBE Studio SDL editor visually introduced, an evaluation of it is now presented:

- The tabs located at the bottom of the editor should be removed as they offer no extra functionality and only add confusion to a user.
- A tab at the bottom of the editor should be added so a user can view the raw contents (XML) of the SDL file.

- On opening the SDL file, the SDL editor should also automatically open the properties view
- It is not necessary for the SDL Editor to make contributions to Eclipse's main menus. The functionality in the menu (Figure 36) can already be accessed through the contextual menu invoked by right-clicking within the editor.

Figure 36 DBE Studio Editors – SDL Editor Menu

4.7 BPEL Editor

The BPEL editor allows users to create, edit and save BPEL files. This editor displays the BPEL model in a number of presentations – a graphical, a hierarchical tree and source representation. Data in the BPEL editor is edited using the standard Eclipse property view within the hierarchical editing mode. What follows is an evaluation of the editor as shown in Figure 37, Figure 38 and Figure 39.

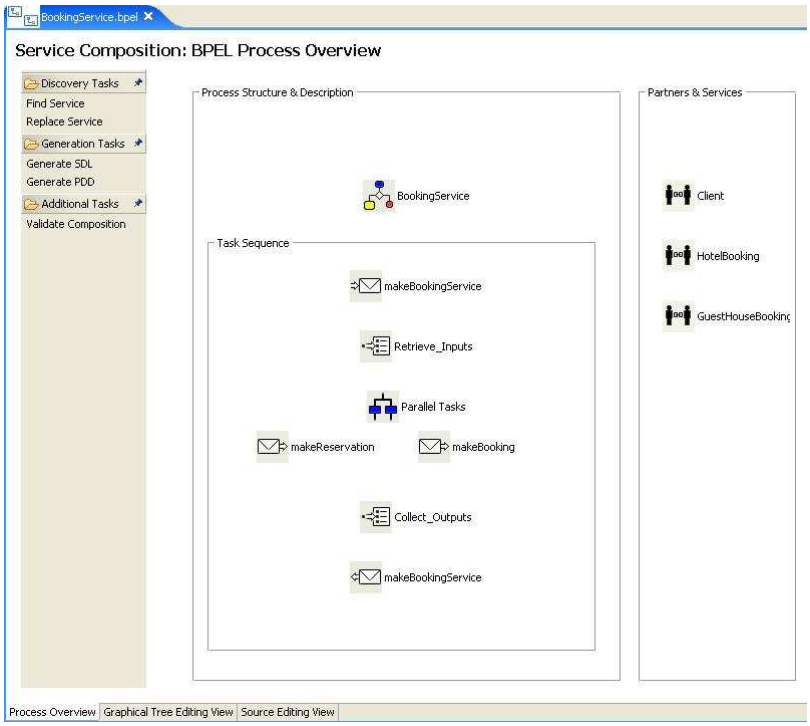


Figure 37 DBE Studio Editors – BPEL Editor

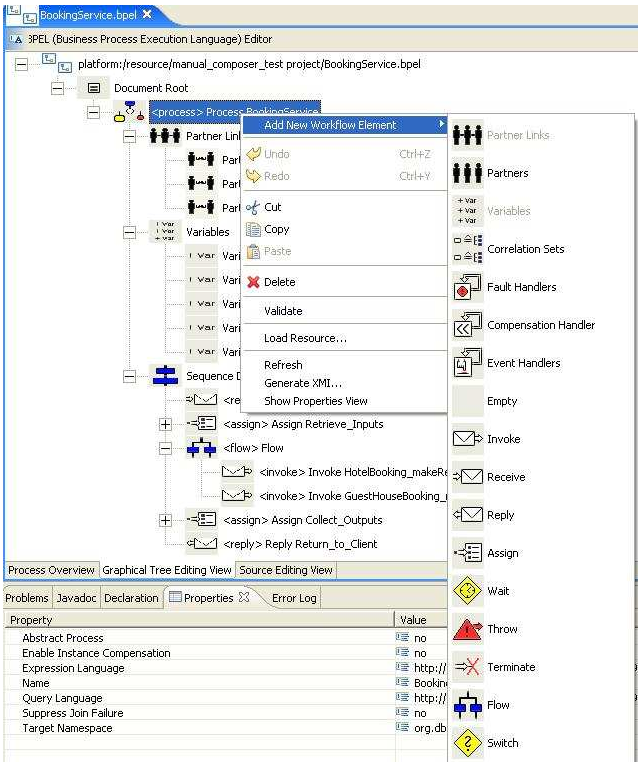


Figure 38 DBE Studio Editors – BPEL Editor

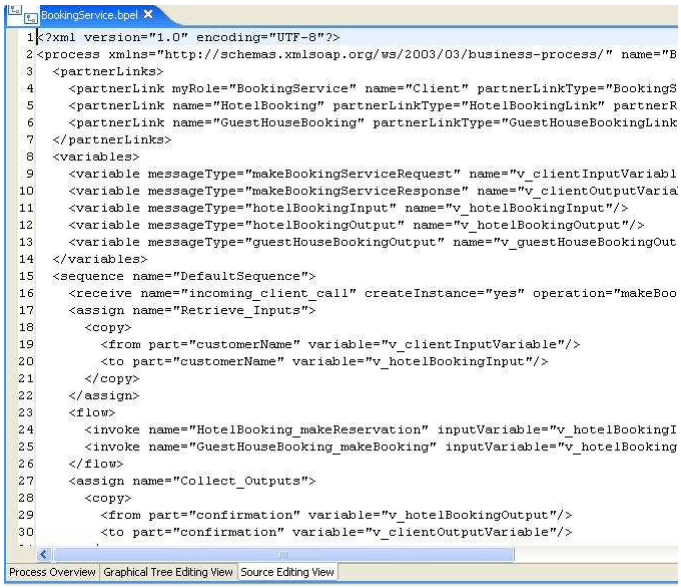


Figure 39 DBE Studio Editors - BPEL Editor

- Syntax highlighting could be added to the source view for easier readability and navigation.
- Ideally, the graphical representation of the work flow should allow editing of workflows.

As the BPEL editor is at a very early stage of development and its sole purpose is to create BPEL compliant work flows, it would be suggested that rather than "re-invent the wheel" that the DBE Studio reuse an already available open source BPEL editor [24] from eclipse with an active community that the DBE Studio developers can contribute to.

4.8 PDD Editor

The PDD editor allows users to create, edit and save PDD files. PDD files are required for deploying workflows to the ActiveBPEL [REF] workflow engine. Data in the PDD editor is edited using the standard Eclipse property view. This editor displays the PDD model in a number of presentations – a graphical, a hierarchical tree and source representation. What follows is an evaluation of the editor as shown in Figure 40.

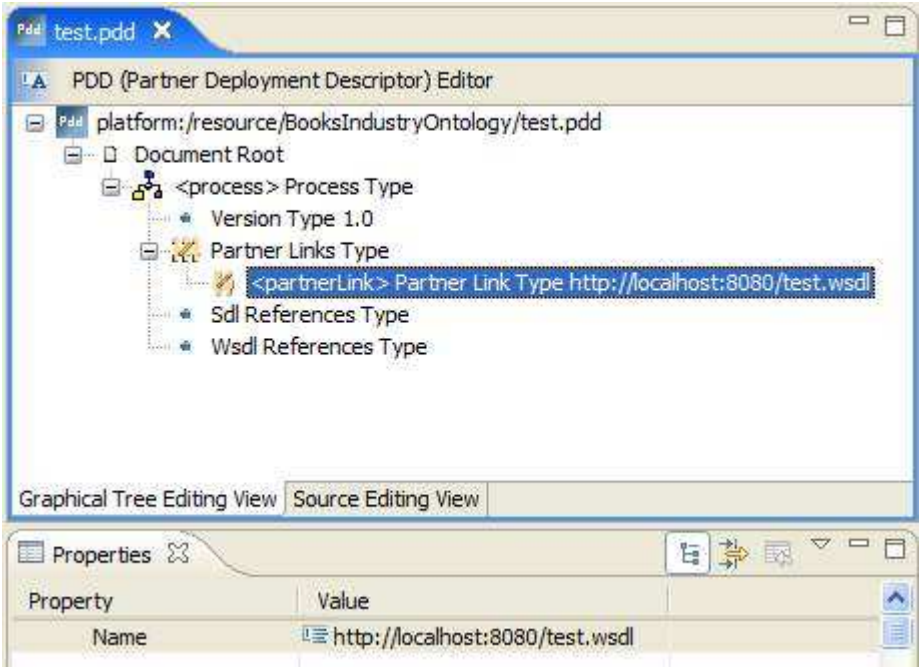


Figure 40 DBE Studio Editors – PDD Editor

With the DBE Studio PDD editor visually introduced, an evaluation of it is now presented:

- The PDD editor's bottom tabs should be renamed as follows:
    - "Graphical Tree Editing View" – "Graphical Tree"
    - "Source Editing View" – "Source"
- Syntax highlighting could be added to the source view for easier readability and navigation.

4.9 Service Manifest Editor

The service manifest editor allows users to create, edit and save service manifest files. It is the service manifest which represents a service offering once published within the DBE network. Typically a service manifest will contain the BML model, BML data and the technical interface of the published service. The service manifest editor presents itself in the style of a form editor and consists of 3 service manifest related tabs. What follows is an evaluation of the editor as shown in Figure 41.

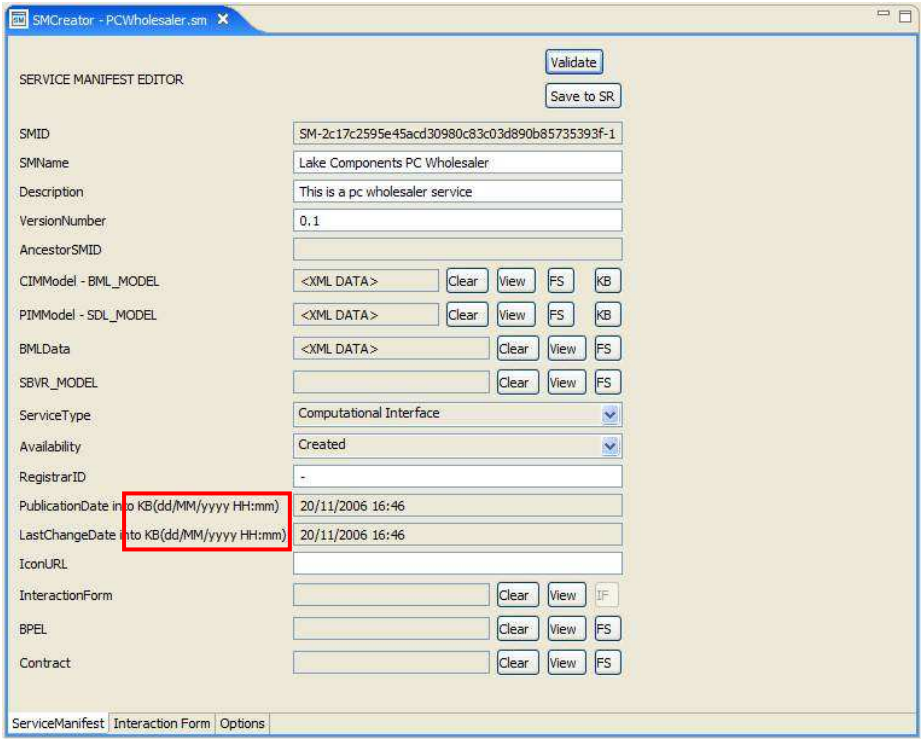


Figure 41 DBE Studio Editors – Service Manifest Editor

With the main tab of the editor, "ServiceManifest", introduced its evaluation is presented here:

- The two buttons at the top right hand corner of the editor are non-standard. If an editor wants to expose functionality through the use of such a button, then that button should be presented via the button toolbar. In this particular case the "Validate" button may not required as one would imagine that as a SM file is created it is created according to the internal model the SM Editor has of the SM schema.
- The "Save to SR" button is also non-standard. This does not conform to the recommendations set out in section 2.5. If a resource has to be saved to a remote system, the semantic registry in this case then the use of an export-type wizard is called for. It's recommended that this approach is taken so as to be compliant with the approach taken by all eclipse plugins.
- The inclusion of the text "CIM_Model" and "PIM_Model" is unnecessary, even if the DBE Studio architecture is one that follows the OMG's MDA approach to system modeling.

These are terms that are rarely used throughout the DBE Studio and it is recommended that it be removed to avoid confusion.

- The distance between text labels (e.g. "SMID") and text input boxes should be reduced. From reviewing this particular editor it can be seen that the date labels (Figure 41, highlighted in red) are the cause of the wide gap between the components. As such, it is recommended that the label "PublicationDate into KB (dd/MM/yyyy HH:mm)" be changed to "KB Publication Date:" and the label "LastChangeDate into KB (dd/MM/yyyy HH:mm)" be changed to "KB Last Change Date:"
- There are a number of read-only fields in the service manifest editor. These should all be grouped and bounded by a box that is titled "SM Info". This will allow a user easily separate information fields from input fields. For greater separation these information fields could be moved so to be displayed in the eclipse standard properties view.
- There are a number of necessary and required fields to be filled when creating a SM file. It is only these that should be present on the first tab that is presented to the user. The others ("SBVR_Model", "Registrar_ID", "IconURL", "InteractionForm", "BPEL", "Contract") should be placed in an ancillary tab titled "Advanced". This reduces visual clutter and makes the task of supplying information to and creating a SM file less daunting.
- The labeling of each field description needs to be improved so that they are consistent with eclipse HIG guidelines. Capitalization, as detailed in section 2.7 should be adhered to. Also this should be applied to the naming of editor tabs where a space should separate the word "ServiceManifest".
- Input validation is required on all text fields that a user is required to enter. A perfect example of this is with the "VersionNumber" field. As the label text describes it is a number that the user is required to enter yet it is possible to enter any arbitrary string in the field and not be alerted to this fact.

With the first editor tab of the service manifest editor discussed, the second editor tab is now presented (Figure 42).

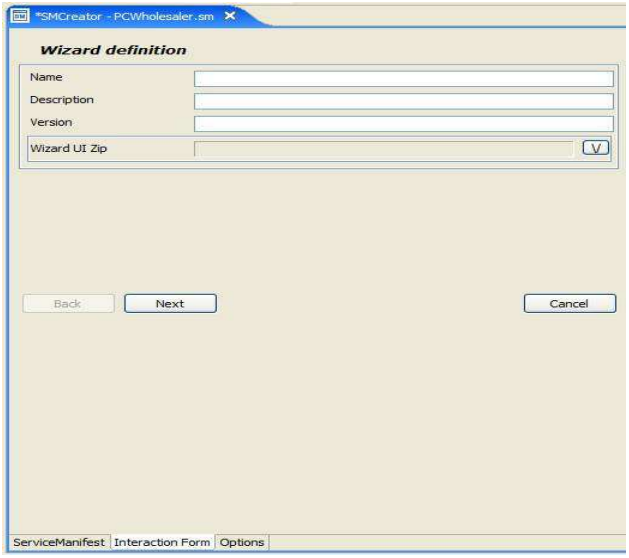


Figure 42 DBE Studio Editors – Service Manifest Editor

With the second editor tab the evaluation of it is as follows:

- One observation of this tab is that its function is orthogonal to the Service Manifest's goal. That goal is the creation of service manifests and not Interaction Forms. Including this tab within the service manifest editor mixes contexts and can induce confusion to a new user of the DBE Studio. It is recommended that this tabs functionality be refactored into a completely separate editor or wizard that is integrated with the SM Editor.

The last tab in the Service Manifest Editor is now presented in Figure 43.

Figure 43 DBE Studio Editors – Service Manifest Editor

With the third tab shown above its evaluation is as follows:

- This tab seems to try to perform 2 distinct tasks within the one UI (persisting SMs and managing SMs), which within the context of eclipse is non-standard. This UI should in fact be removed and it's functionality integrated into other places within eclipse as following:
    - Management (Listing, Deleting) of SMs should be done via a supporting view. SM management is also a contentious issue due to the limited support for security in the DBE Studio.
    - Loading a SM should be done in the style of eclipse "Import" wizards and following a successful import having the SM Editor open the imported SM.
    - Saving service manifests to the semantic registry should be accomplished using an export wizard. If a service manifest is to be saved to the local file system then a mechanism similar to saving any file in eclipse should be implemented.
- The modification button should be removed as this is functionality that can be mirrored using "import and update" semantics, where the update is essentially a save-type operation.

5 Views

In this chapter the views contained within the DBE Studio are analyzed and recommendations are made upon that analysis. A view is a visual component within a workbench and is normally grouped within perspectives along with an editor. Views support the role of an editor and provide information about the model that represents what is displayed in the editor. Views normally represent information of an editor in a one dimensional fashion.

Within the DBE Studio there are nine views currently defined. Below is a table of all the views contained in the DBE Studio along with a brief description of each.

Name	Function	Partner
Keyword Search	Search for services or models based on a keyword	Technical University of Crete
Model Navigator	Lists all models contained in the KB ordered by SME ID.	Technical University of Crete
Outline View	Displays the object contained within an ontology	Technical University of Crete
Ontology Viewer	Allows users to search for ontology classes and use in the BML editor	Technical University of Crete
Property View	Shows data of selected objects in an editor and allows them to be edited	Technical University of Crete
Query Navigator	Shows all queries and query templates used by the user.	Technical University of Crete
Query Properties	Shows the properties of a query	Technical University of Crete
Query Results	Lists the results of a query to the KB	Technical University of Crete
Query Template	Shows the properties of a query template	Technical University of Crete

Table 4 DBE Studio Views

What are listed in the above table (Table 4) are the DBE Studio views that are to be considered for evaluation in this document. These very same views can be accessed within the DBE Studio through the menu "Windows->Show View ->Other…" and are illustrated below:

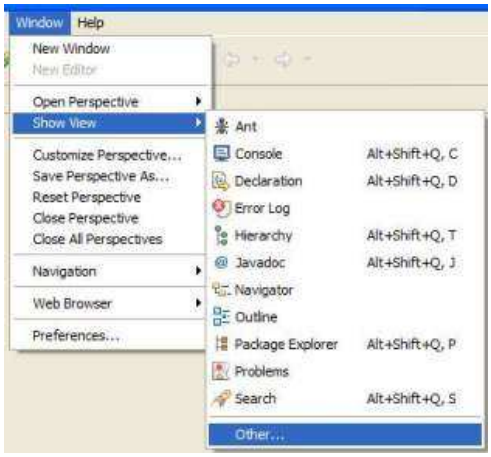


Figure 44 DBE Studio Views – Access

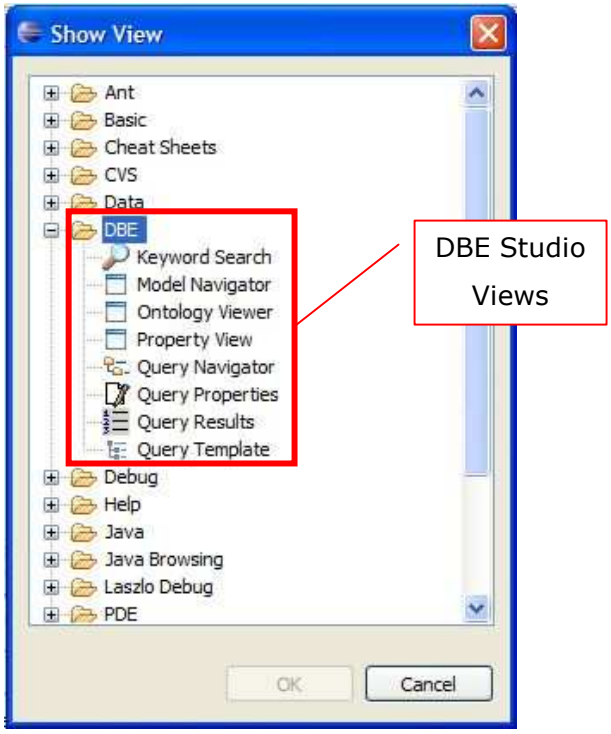


Figure 45 DBE Studio Views - Selection

Before considering each individual view, the general Eclipse recommendations are detailed which should always be considered when designing and implementing Eclipse views.

5.1 UI Recommendations

When designing views, it is prudent to consult the Eclipse HIG [1] for guidelines. These guidelines are found in [1] and expanded upon there and also form the basis for the evaluation of DBE views. Hence, it is useful to summarize them for reference:

Ref.	Guideline Description
VG1	Use a view to navigate a hierarchy of information, open an editor, or display the properties of an object.
VG2	Modifications made within a view must be saved immediately.
VG3	Only one instance of a view may exist in a perspective.
VG4	A view must be able to be opened in more than one perspective.
VG5	A view can be opened from the Window -> Show View menu.
VG6	The view label in the title bar must be prefixed with the label of the view in the Perspective > Show View menu.
VG7	If a view contains more than one control, it may be advisable to split it up into two or more views.
VG8	When a view first opens, derive the view input from the state of the perspective.
VG9	If a view displays a resource tree, consider using the window input as the root of visible information in the view.
VG10	Use the view pull-down menu for presentation commands, not selection-oriented commands.
VG11	Use the standard order of commands for view pull-down menus.
VG12	Put only the most commonly used commands on the toolbar. Any command on a toolbar must also appear in a menu, either the context menu or the view menu.
VG13	Fill the context menu with selection oriented actions, not presentation actions.
VG14	Use the standard order of commands for view context menus.
VG15	Fill the context menu with a fixed set of commands for each selection type, and then enable or disable each to reflect the selection state.
VG16	If an object appears in more than one view, it should have the same context menu in each.
VG17	Register all context menus in the view with the platform.
VG18	Implement a Command Filter for each object type in the view.
VG19	If a view has support for Cut, Copy, Paste, or any of the global commands, these

	commands must be executable from the same commands in the window menu bar and toolbar.
VG20	Persist the state of each view between sessions.
VG21	Navigation views should support "Link with Editor" on the view menu

Table 5 Recommended View Guidelines

These guidelines are used through out this section on the evaluation of DBE Studio views and the above table will serve as a reference point for this evaluation and also developers creating or refactoring DBE Studio plugins.

5.2 Resource Browsing Views

Resource browsing views are ones that allow users browse a collection of resources, such as model and files, and perform related operations upon them.

5.2.1 Outline View

The DBE Studio outline view allows users of the DBE Studio to view and interact with the classes and diagrams of a particular ontology. The outline view supports the ontology editor. The outline view is found in one perspective – the "DBE Ontology Analysis". Below is a screenshot of how the DBE Studio outline view generally appears once it has been opened by the user.

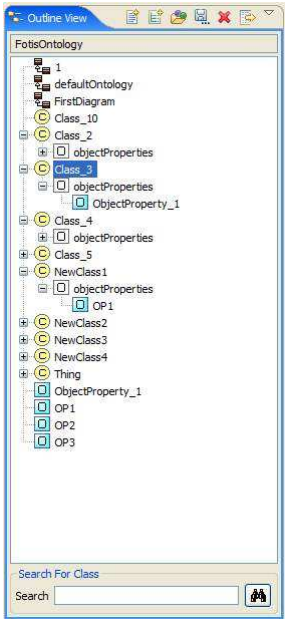
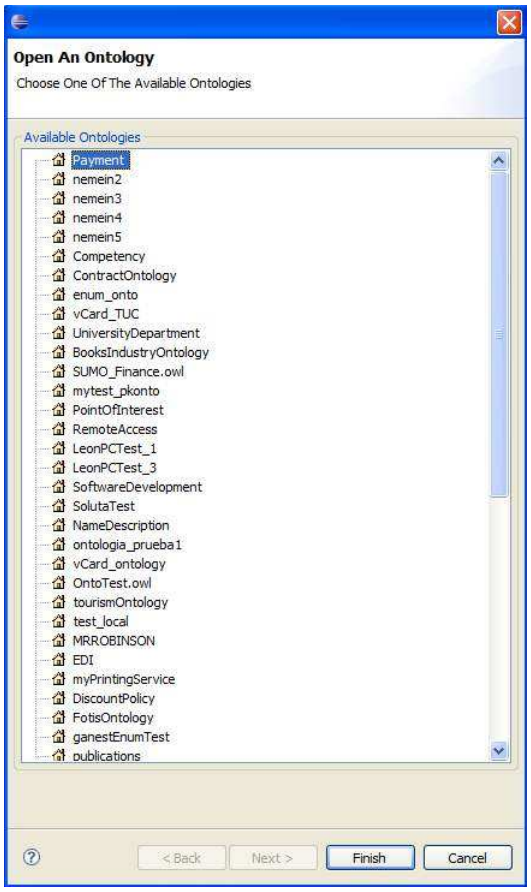


Figure 46 DBE Studio Views – Outline View

With the DBE Studio outline view visually introduced, an evaluation of it is now presented:

- There is the capability to delete ontologies from the KB. As there is no security implemented guarding ontologies this functionality should be considered.
- It is not possible for a user want to move the outline view within the workspace.
- To reflect better its function, it is recommended that it be renamed to "ontology outline".
- A full menu of actions that can be accomplished with the view is available in the view's menu (accessed through the view's toolbar). All these functionalities should be made accessible in the contextual menu when a user right-clicks within the view.
- There is the functionality to save an ontology to the KB and this is given to the user through a view button at the top of the view. However, to be compliant with the recommendations in section 2.5, an "Export" wizard should be implemented to correctly reflect this functionality.
- In order to open an ontology the user can open an ontology from the KB. Again this is non-compliant to the recommendations in section 2.5. When the user does select this "open ontology" operation the following dialog is displayed:



There are a number of issues associated with this dialog.

- When the dialog opens, it height is always the maximum height of the screen displaying the dialog. Resizing the dialog and reopening it subsequentially does not solve the problem either.
- When this dialog opens it attempts to retrieve all ontologies stored in the KB. This operation could potentially be a long one and retrieving all ontologies from a KB could be impractical especially if there are thousands. Measures should be put in place within this dialog so that this does not become a future issue.
- It would make the selection of an ontology from the presented list easier if there was some means to filter or sort the list.

5.2.2 Model Navigator

The DBE Studio model navigator view allows users of the DBE Studio to view and interact with BML models that have been published within the DBE on the KB. The model navigator view supports the BML model editor. The model navigator view is found in one perspective – the "DBE Business Analysis". Below is a screenshot of how the DBE Studio model navigator view generally appears once it has been opened by the user.

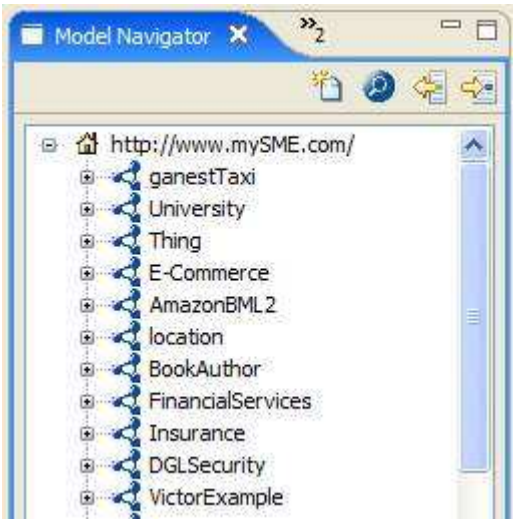


Figure 47 DBE Studio Model Navigator View

With the DBE Studio model navigator view visually introduced, an evaluation of it is now presented:

- A representative icon in the view's title bar is recommended.
- There is the capability to delete models from the KB. As there is no security implemented guarding ontologies, this functionality should be removed.
- There is the capability to delete BML models from the KB. As there is no security implemented guarding BML models this functionality should be considered.
- The menu of actions that can be accomplished with the view should be made available in the in the contextual menu when a user right clicks within the view.

- Unlike many eclipse views, the model navigator cannot be moved within the workspace. It is recommended that this functionality be implemented.
- As soon as the model navigator loads it attempts to load all models in the KB. If there are thousands of models in the KB this could become an expensive and lengthy operation. Measures should be taken so this does not become an issue.
- With the addition of more models, the listed displayed by the model navigator becomes long and difficult to navigate. A solution to make this list manageable and easier to use would be to adopt the eclipse view-filtering mechanism. This would allow the user to filter the view based on various possible parameters, most useful being "Date Modified", "Creator" and a free text entry (e.g. to list all models beginning with "an", the free text would be "an*").

5.2.2.1 Connection Management Issues

Further to section 2.3, there is no way for the user to specify when he/she wants to connect to retrieve data. Also there is no way for the user to initiate/terminate a connection. It is recommended that, more control should be given to the user with the introduction of the connection manager plugin.

5.2.3 Query Navigator

The DBE Studio query navigator view allows users of the DBE Studio to view and interact with semantic queries and query templates that the user authored. The query navigator view supports both the query and query template editors. The query navigator view is found in one perspective – the "DBE Semantic Discovery". Below is a screenshot (Figure 48) of how the DBE Studio query navigator view generally appears once it has been opened by the user.

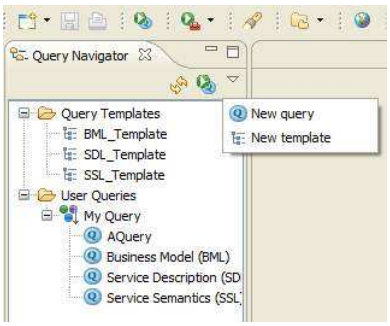


Figure 48 DBE Studio Query Navigator

With the DBE Studio query navigator view visually introduced, an evaluation of it is now presented:

- Saved queries are in a separate project. It is recommended that these queries be saved in the project that was originally created with the DBE project wizard (6.2.3). Doing this is inline with the recommendations set out in section 2.4.

5.3 Properties Views

When designing property views, it is prudent to consult the Eclipse HIG [1] for guidelines. These guidelines are found in [1] and expanded upon there and also form the basis for the evaluation of DBE property views. Hence, it is useful to summarize them for reference:

Ref.	Description
PVG1	Use the Properties view to edit the properties of an object when quick access is important, and you will switch quickly from object to object.
PVG2	Use a Properties Dialog to edit the properties of an object which are expensive to calculate.
PVG3	Use a Properties Dialog to edit the properties of an object which contain complex relationships between one another.
PVG4	Properties Dialog should contain the superset of items shown in the Properties view.

Table 6 Recommended Properties View Guidelines

5.3.1 Property View

The DBE Studio property view allows users to view and edit data contained in a model loaded in an editor. It is used in the ontology analysis and business analysis perspective. This property view loads different content editors depending on what object is selected within the ontology or BML editor. As such the evaluation of this view is on these content editors and not the property view itself, which is essentially a container (Figure 49).

Figure 49 DBE Studio Property View

5.3.1.1 Ontology Content View Editors

There is essentially one property view for ontology diagram objects that dynamically adapts to what ever the user has focus on within the ontology analysis perspective. This view contains a number of sub-views that allow the editing of particular properties diagramming object. If the user selects, for example, a class-type diagramming object the properties view adapts to this object and presents a number of diagramming object specific sub-views contained within the properties view as shown in Figure 50, Figure 51 and Figure 52. The examples shown in these three screenshots are properties that are associated with an ontology class diagramming object. There are similar views for the other ontology diagramming objects.

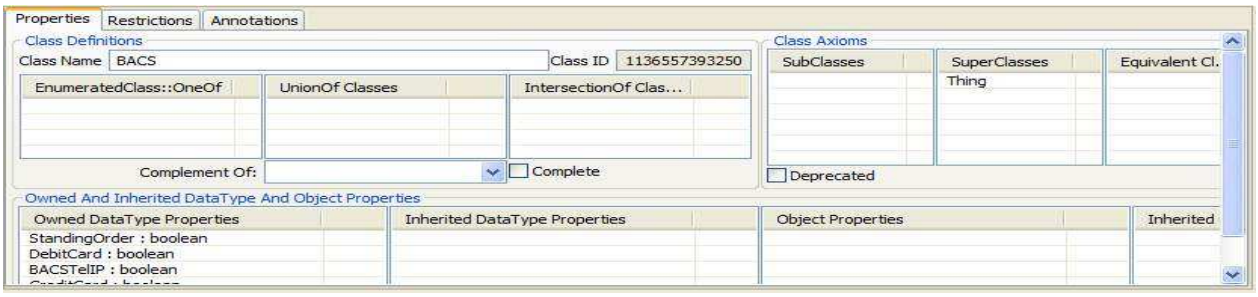


Figure 50 DBE Studio Views – Main Ontology Content View

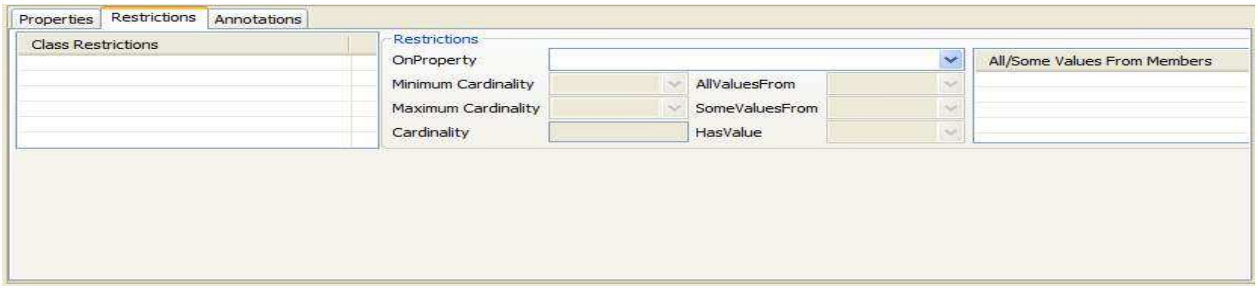


Figure 51 DBE Studio Views – Ontology Restrictions View

Figure 52 DBE Studio Views – Ontology Annotations View

The main recommendation for the above property views (Figure 50, Figure 51 and Figure 52) is that the amount of data that is editable at one time be reduced. Due to the flexibility of the BML language, there is a large amount of properties that can be set and the current properties view does not suit this as there just is not enough "real estate" to display editable properties to the user. It is the suggestion of this evaluation that more property tabs be created so that visual organization of information presented is optimized. A specific example can be given using Figure 50. In this properties view sub tab, there are three groupings of editable properties ("Class Definition", "Class Axioms" and "Owned and Inherited DataType and Object Properties"). Each of these three could

be separated out into their own appropriately titled tab. In addition help documentation and/or cheat sheets could be provided in order to explain the concepts and properties displayed to the user.

In the next two screen shots (Figure 53 and Figure 54), the properties views are shown for data types contained within an ontology.

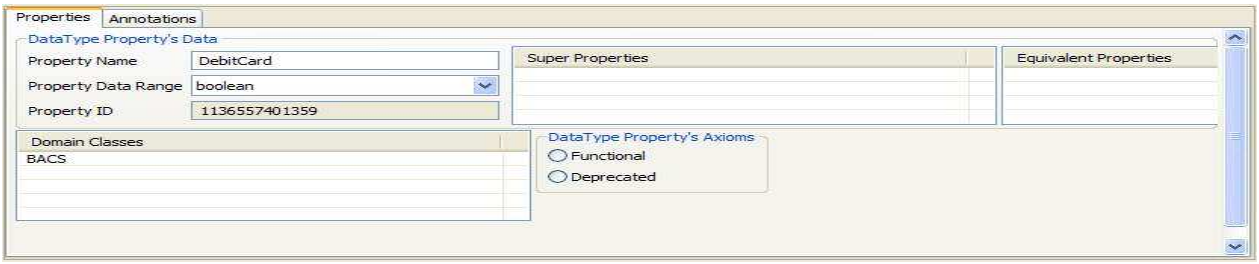


Figure 53 DBE Studio Views – Ontology Data type Properties View

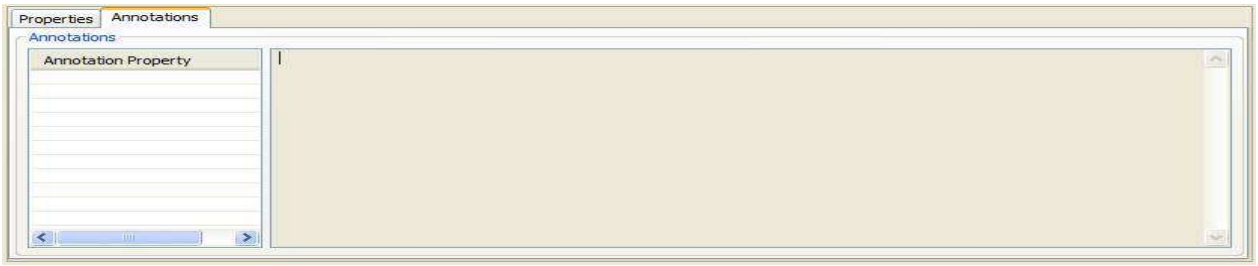


Figure 54 DBE Studio Views – Ontology Data type Properties View

5.3.1.2 BML Content View Editors

There is essentially one property view for BML diagramming objects that dynamically adapts to what ever the user has focus on within the ontology analysis perspective. This properties view is very much similar to the one present in the ontology editor. This view contains a number of sub-views that allow the editing of particular properties diagramming object. If the user selects, for example, a class-type diagramming object the properties view adapts to this object and presents a number of diagramming object specific sub-views contained within the properties view. The examples shown in these three screenshots are properties that are associated with a BML Business Entity diagramming object. There are similar views for the other ontology diagramming objects.

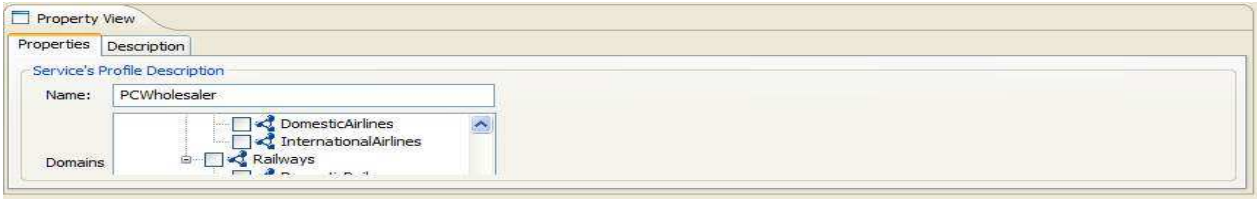


Figure 55 DBE Studio Views – Service Profile Properties View

This view (Figure 55) allows the service profile being modeled be assigned to a particular ontology and allows the editing of the service profile name. An issue associated with this view is that there is not enough space to accommodate the list of ontologies. Perhaps a solution to this issue would be to provide a "Browse…" button that would open a dialog from where a user can select the required ontology.

Figure 56 DBE Studio Views – Service Profile Description View

This particular view (Figure 56) is present in a number places within the Business Analysis perspective. It allows the user attach a description to a diagramming object. However, there is no way that this allows the user to do so. As a recommendation it would be advisable to either enable this functionality or remove the view.

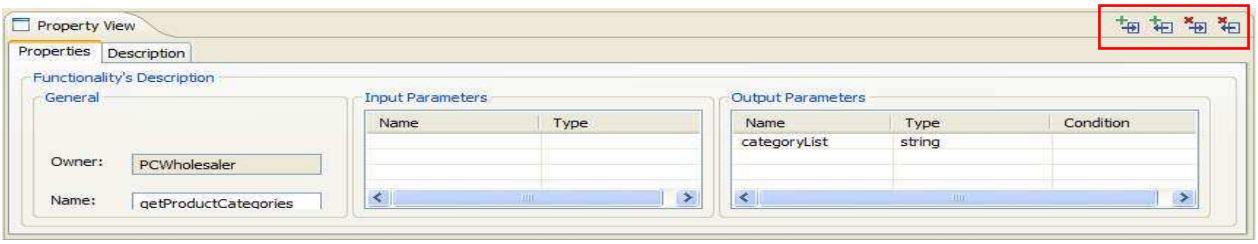


Figure 57 DBE Studio Views – Service Profile Functionality Property View

In the view above (Figure 57), a user can modify a service's functionality. It is here where the user can add parameters to the functionality of a service profile. These parameters are logically separated into input and output parameters. Those parameters can be added to and removed from the service profile's functionality by clicking the relevant button (highlighted in red). However, it would add great value to this view if a right click contextual menu was added to both input and output parameter list box.

Figure 58 DBE Studio Views – Service Profile Attribute Property View

Above (Figure 58), displays the view that allows a user edit an attribute of a service profile. From the "Type" dropdown menu the user can select basic types and also types defined in an ontology for the selected attribute.

Figure 59 DBE Studio Views – Business Element Property View

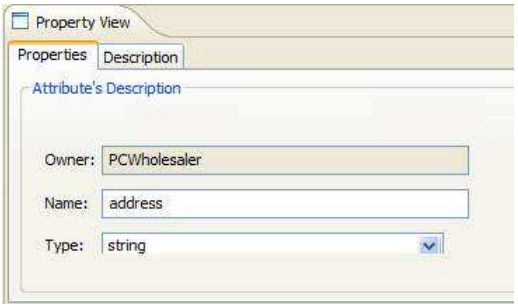


Figure 60 DBE Studio Views – Business Attribute Property View

In the above two screenshots (Figure 59 and Figure 60), the capabilities of editing the properties of a business element are displayed. Both are logically laid out. In the case of Figure 59, this allows the editing of a business element's name. In Figure 60, it allows the editing of an attribute contained

within a business element. From the "Type" dropdown menu the user can select basic types and also types defined in an ontology for the selected attribute.

5.3.2 Query Properties

There is essentially one property view for query properties that dynamically adapts to what ever the user has focus on within the semantic discovery perspective. If the query editor itself, rather than its content, has the user's focus then the properties shows general information about the query, as shown in Figure 61. If the user is actively editing a field in the query editor then the properties view adapts to this and adds extra functionality associated with the field such as "Operation" (how restrictions should be placed on a field value) and "Weight" (how important the field is within the context of the query, based on a percentage). These are highlighted and shown in Figure 62 and can be adjusted to suit the users needs.

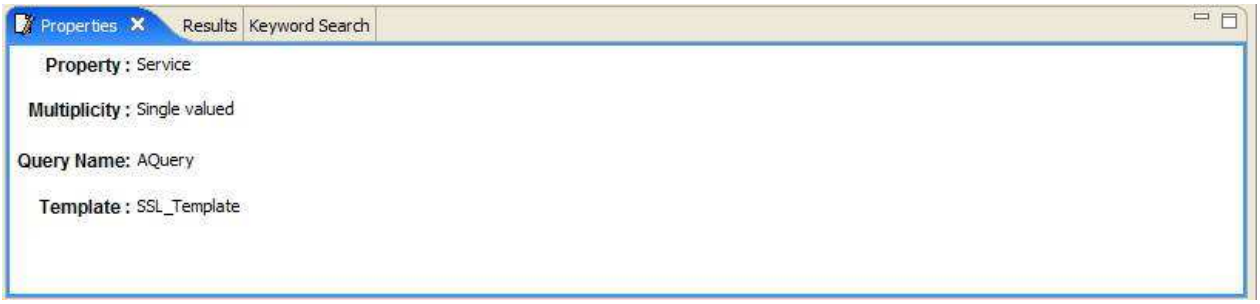


Figure 61 DBE Studio Views – General Query Information

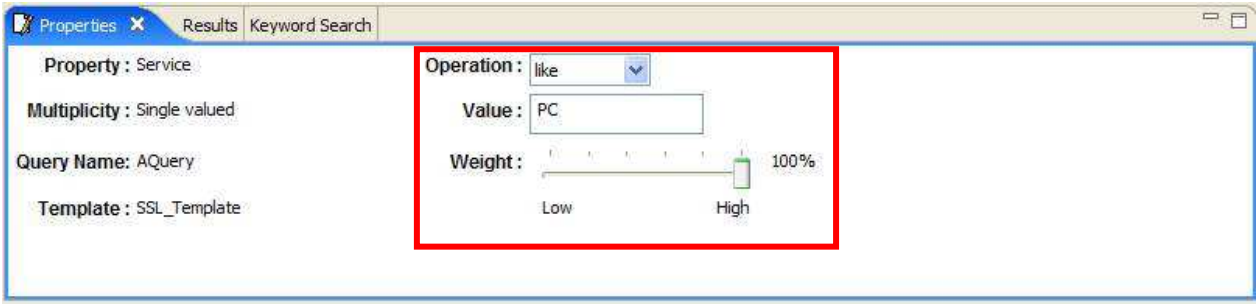


Figure 62 DBE Studio View – Field Specific Parameters

On evaluating these two views there was nothing found that was deemed in need of improvement.

5.4 Ontology Viewer

The DBE Studio ontology viewer allows users to search and view classes contained in an ontology. It is used in the business analysis perspective. This view is used to help the user when creating BML models and is shown below.

Figure 63 DBE Studio Ontology Viewer View

With the DBE Studio ontology viewer view visually introduced, an evaluation of it is now presented:

- The ontology tree component could be positioned so as not to clip the drop down menu (highlighted in green). This is due to the fact that the components in the view do not resize proportionately.
- There is functionality in this view to import a pre-existing ontology. It is exposed by providing a button (highlighted in red) in the toolbar of the ontology view. However, if a resource is to be imported, this is not the standard way to do so in Eclipse. The import functionality should be exposed using an import wizard (see 6.4).
- The additional info field should not be used here. Instead the additional information displayed in this field should be loaded into the properties view.
- A representative icon should be used in the view's title bar.
- It would be very useful, once a class is found, if a class could be dragged and dropped from the "Ontology Tree" into the BML editor
- The ontology viewer cannot be maximized, minimized or closed like standard Eclipse views. It is recommended that these functions be supported.

5.4.1 Connection Management Issues

Further to section 2.3, this component automatically connects to the KB in order to display available ontologies and classes. This has the disadvantage that there is no way for the user to specify when he/she wants to connect to retrieve data and so disables the user and removes control from the user.

Also, when searching for a particular class within an ontology the result is not returned asynchronously. This has the result of locking up the whole DBE Studio UI.

5.5 Query Template

The DBE Studio query template property view allows users of the DBE Studio to view the properties and add new BML elements to the template. The query template property view supports the query and query template editors. The keyword search view is found in one perspective – the "DBE Semantic Discovery" perspective. Below is a screenshot of how the DBE Studio query template property view generally appears once it has been opened by the user.

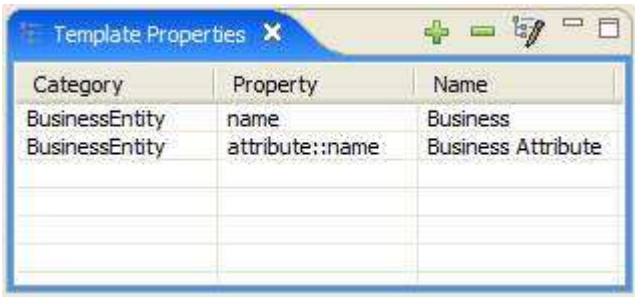


Figure 64 DBE Studio Views – Query Template View

With the DBE Studio query template property view visually introduced, an evaluation of it is now presented:

- It should be possible to right-click on an entry in the view and perform the activities of "Add" and "Remove" BML element. On clicking "Add" the following dialog (Figure 65) is displayed, again with the condition that a connection exists to the KB

5.5.1 Connection Management

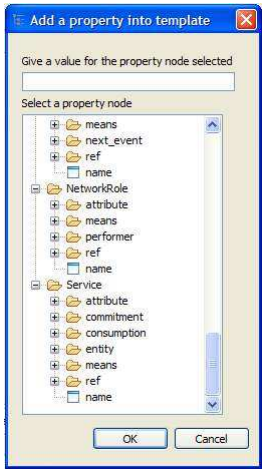


Figure 65 DBE Studio Views – Query Template View, Adding a property

Above in Figure 65 is the dialog that allows a user to add new properties to a query template. It is easy to use. However, it is also subject to the KB connection behaviour previously described in 6.2.5.1.

5.6 Query Results

The DBE Studio query result view allows users of the DBE Studio to view the properties and add new BML elements to the template. The query result view supports the query and query template editors. The query result view is found in one perspective – the "DBE Semantic Discovery". Below is a screenshot of how the DBE Studio query result view generally appears once it has been opened by the user. In this results view, the user can right click on a model of interest and open it in the business analysis perspective for inspection and editing. The query results view is also used in the service composition perspective and in this perspective its functionality is extended so suitable discovered services can be added to a service composition using the right-click contextual menu.

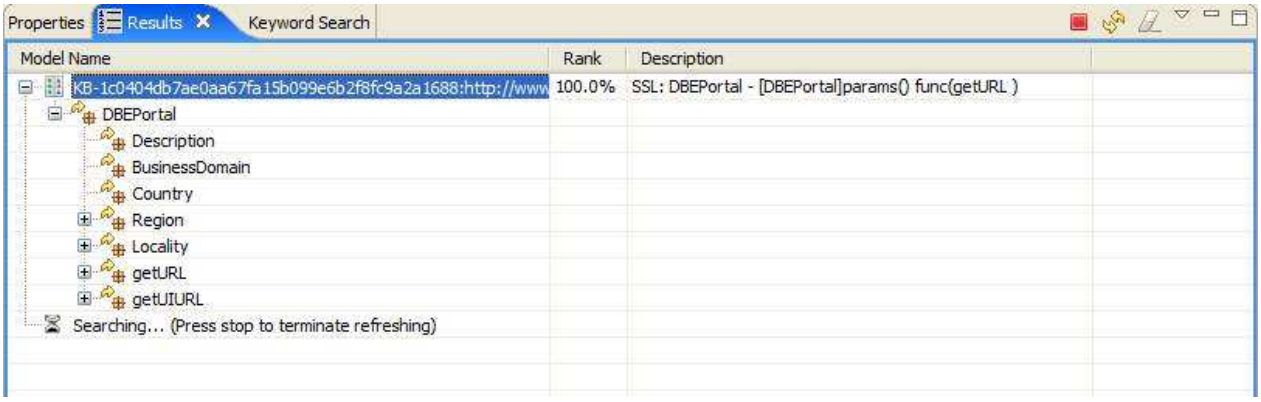


Figure 66 DBE Studio Views – Semantic Query Tool Results

5.7 Keyword Search

The DBE Studio keyword search view allows users of it to search for and retrieve information about both published models, including BML and SDL, and DBE services represented by Service Manifests. The keyword search view supports the query and query template editors. Not only is it used as a support to editors but it can be used in a "stand-alone" mode for simply searching for models/services. The keyword search view is found in two perspectives – the "DBE Semantic

Discovery" and the "DBE Service Composition". In this keyword results view, the user can right click on a model of interest and open it in the business analysis perspective for inspection and editing. Below is a screenshot of how the keyword search view generally appears once it has been opened by the user.

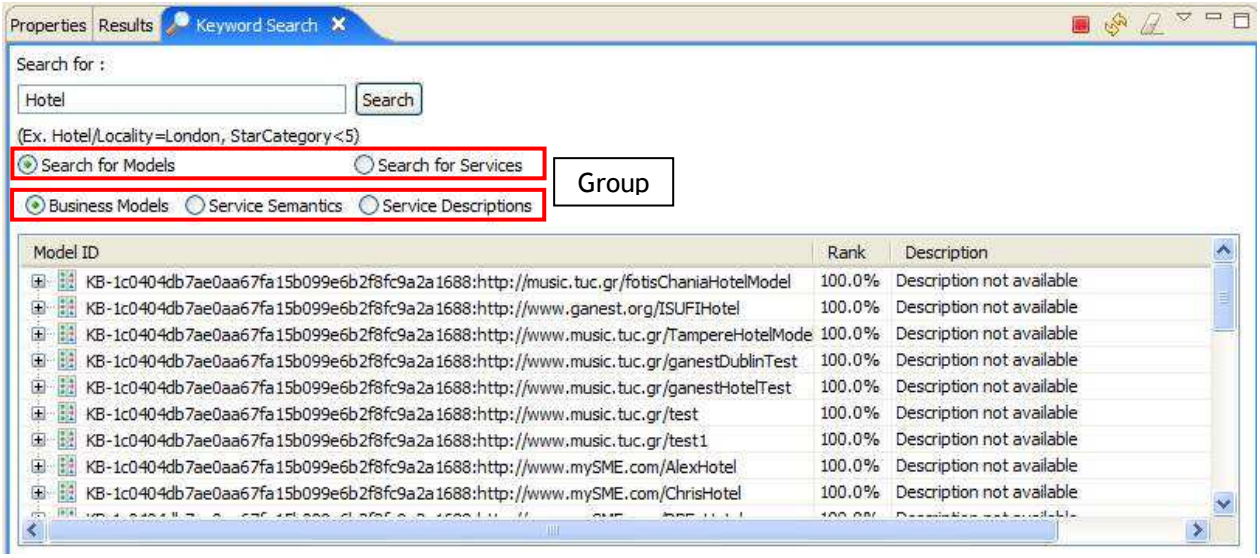


Figure 67 DBE Studio View – Keyword Search

With the DBE Studio keyword search view visually introduced, an evaluation of it is now presented. It is recommended that:

- The three radio buttons should be grouped.
- When the view is activated the keyword search box should receive focus.

5.7.1 Connection Management Issues

On issuing a search request for models, for example using the "Hotel" keyword, the keyword search view froze the whole UI when populating the results list box. It was unclear if it was because data was being retrieved was too large or an error had occurred. If the data set returned was too large it is recommended that the keyword search view should put in place some sort of limiting mechanism. This should also avoid freezing the UI when there is interaction with a long list box.

6 Wizards

In this chapter the wizards contained within the DBE Studio are analyzed and recommendations are made upon that analysis. Wizards provide a major contribution of functionality in the DBE Studio. In any case where the creation, import and export of a file or model are required, these particular types of modal dialogs should be used. In general, a wizard is a modal dialog that assists the user to automate a task that comprises of a number of sequential steps in a specific order. Within Eclipse, wizards are usually used for the creation of new resources, the import of resources and the export of resources.

The DBE Studio currently has 16 wizards that perform a number of tasks. Table 7 lists all the wizards contained in the DBE Studio along with a brief description of each.

Name	Description	Wizard -Type	Partner
BML Data Wizard	Associates BML Data with a BML Model	Creation	University of Central England
BPEL Wizard	Creates an initial skeleton BPEL model	Creation	Trinity College Dublin
DBE Project Creation Wizard	Creates a DBE Project Structure	Creation	Intel Ireland Ltd.
Query Wizard	Creates a semantic query based on a query template	Creation	Technical University Crete
Query Template Wizard	Creates a semantic query template to based queries on	Creation	Technical University Crete
PDD Wizard	Creates an empty PDD model	Creation	Trinity College Dublin
SDL Wizard	Creates an empty SDL model	Creation	Soluta
Service Manifest Wizard	Creates an empty service manifest	Creation	Soluta

Test Scenario Creator	Creates a test scenario to run many test cases	Creation	University of Surrey
Test Case Generator	Creates a test case for one interface	Creation	University of Surrey
SSL2SDL Wizard	Transforms SSL (input) to SDL (output)	Contextual	Soluta
SDL2Java Wizard	Transforms SDL (input) to Java skeletons (output)	Contextual	Soluta
SDL Import Wizard	Imports a requested SDL file from the KB	Import	Soluta
SDL Export Wizard	Exports a SDL file to the KB	Export	Soluta
Service Exporter Wizard	Exports/Deploys a service implementation to the servent	Export	Intel Ireland Ltd.
Metering Wizard	Adds the ability to meter on methods of a particular interface	Export	Waterford Institute of Technology

Table 7 DBE Studio Wizards

The DBE Studio wizards that are to be considered for evaluation in this document are listed in Table 7. These very same wizards can be accessed within the DBE Studio through four main fashions:

- Access through the "File->New" menu.
- Access through a right-click invoked contextual menu
- Access through the "File->Import" menu
- Access through the "File->Export" menu

It is under these access-type categorizations that the wizards of the DBE Studio will be evaluated. Before considering each individual wizard, the general Eclipse recommendations are detailed which should always be considered when designing and implementing Eclipse wizards.

6.1 UI Recommendations

When implementing an Eclipse plugin that requires the features and presentation interface of a wizard, there are a number of guidelines that should be adhered to. These guidelines are found in [1] and expanded upon there and also form the basis for the evaluation of DBE wizards. Hence, it is useful to summarize them for reference, as shown in Table 8.

Ref.	Description
WG1	Use a wizard for any task consisting of many steps, which must be completed in a specific order.
WG2	Wizards must contain a header with a banner graphic and a text area for user feedback. It must also contain Back, Next, Finish, and Cancel buttons in the footer.
WG3	Start the wizard with a prompt, not an error message.
WG4	Seed the fields within the wizard using the current workbench state.
WG5	Validate the wizard data in tab order. Display a prompt in the text area for user feedback when information is absent and an error when information is invalid.
WG6	Only enable the Next / Finish buttons if all required information in the dialog is present and valid.
WG7	Remove all programming message IDs from wizard text.
WG8	Use a Browse Button whenever an existing object is referenced in a wizard.
WG9	If a new file is created, open the file in an editor. If a group of files are created, open the most important, or central file in an editor. Open a file explaining the example project on creation.
WG10	If a new project is created, prompt users and change the active perspective to suit the project type.
WG11	If a new object is created, select and reveal the new object in the appropriate view.
WG12	Create folder objects in a wizard if reasonable defaults can be defined.

WG13	Use the term "Project name" for the input field label when the item must be a Project; otherwise, use the term "Folder name". Do not qualify the term.

Table 8 Recommended Wizard Guidelines

These guidelines are used through out this section on the evaluation of DBE Studio wizards and the above table will serve as a reference point for this evaluation and also developers creating or refactoring DBE Studio plugins.

6.2 New Resource Creation Wizards

These wizards are displayed to the user as a result of requesting the creation of a file or model through Eclipse's "New" dialog (see Figure 68, accessed through "File->New…") or by right clicking on a recognized file type that is implicitly associated with a creation-type wizard that uses that file as input.

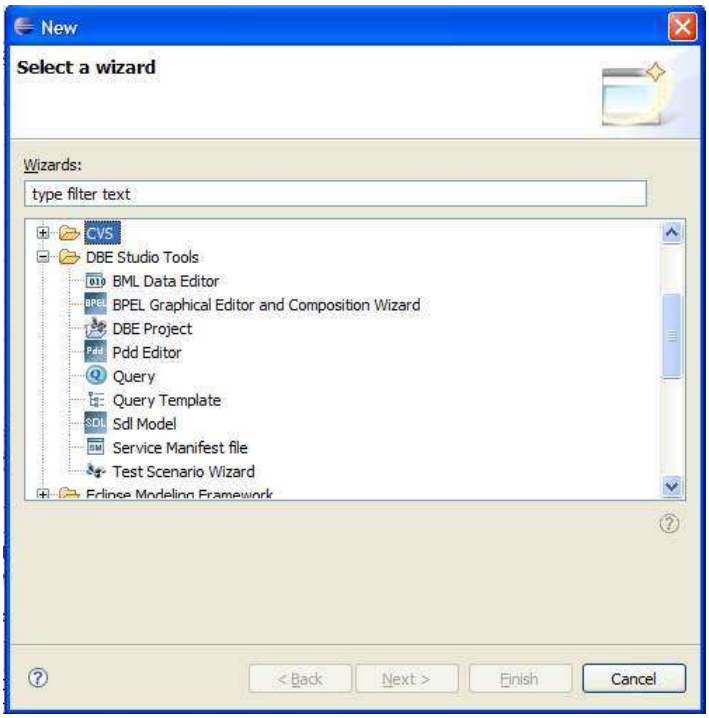


Figure 68 DBE Studio Creation Wizards

The following is an evaluation of the DBE Studio category for new wizards:

- The text that describes the category ("DBE Studio Tools") should be renamed to "DBE Studio". There are a number of entries under this category that require renaming. Noting capitalization changes, those are:
    - "BML Data Editor" to "BML Data"
    - "BPEL Graphical Editor and Composition Wizard" to "BPEL Composition Model"
    - "Pdd Editor" to "PDD File"
    - "Service Manifest file" to "Service Manifest"
    - "Test Scenario Wizard" to "Test Scenario"
- The icons to which all entries in the category correspond to provide a good hint to the user as to what the function of each entry performs. However, this is not the case for the following entries where the icon is too similar and as such maybe be visually misleading due to their similarity:
    - "BPEL Graphical Editor and Composition Wizard"
    - "PDD Editor"
    - "SDL Model"
- Also the icon of the entry "Test Scenario Wizard" is too generic to give a reasonable visual representation of the named entry.
- Descriptive text under "Select a wizard" at the top of the dialog, should be displayed as each of the wizard in the category "DBE Studio Tools" title text is selected. Entries under this category that require this are:
    - "BML Data Editor"
    - "DBE Project"
    - "Sdl Model"
    - "Service Manifest"
    - "Test Scenario Wizard"

- The help button in the lower left of the dialog (see Figure 68) should also be investigated as a means for revealing help to a user.

6.2.1 BML Data Wizard

The BML Data wizard is a wizard to assist the user in creating a BML Data file based on a particular BML model. This wizard consists of three wizard pages and as soon as the user clicks on the "Finish" button of the wizard, the BML Data Editor is loaded up.

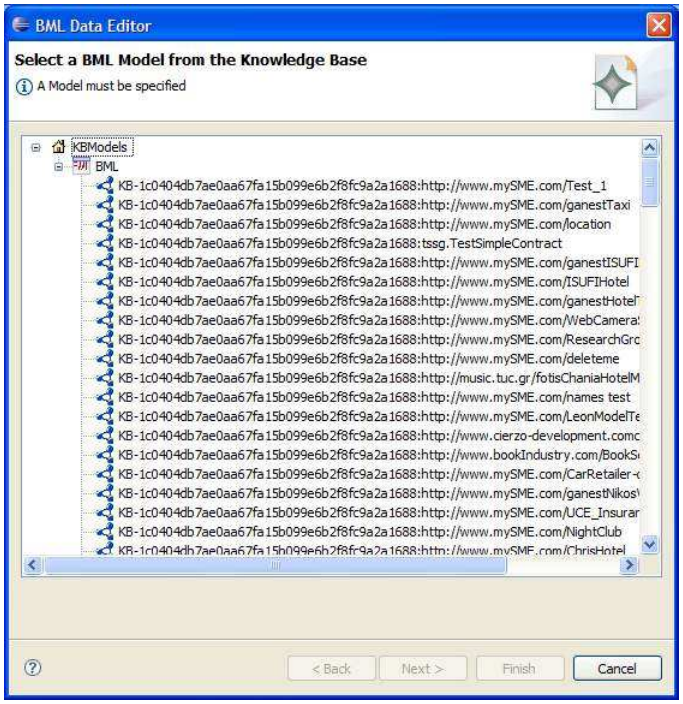


Figure 69 DBE Studio Wizards - BML Data Wizard

The following is an evaluation of the BML Data Wizard:

- When this wizards starts up it attempts to retrieve all models from the KB. This could be problematical if there are a large number of models. Measures should be taken so this potential problem is negated.

- When searching through the list of BML models, it is difficult to find the appropriate model. A filtering mechanism would be useful for this list of models.
- If a local BML model exists in the current workspace, this should be used in favour of contacting the KB and incurring the expense of a network call.
- There is a third page in the BML Data wizard which is unnecessary and should be removed.
- The title of the wizard currently reads "BML Data Editor". It should read "BML Data Wizard".
- A top right hand corner icon representative of the wizard is required.

6.2.1.1 Connection Management Issues

Further to section 2.3, within the BML Data Wizard the connection management could be improved. There is a noticeable time from invoking the wizard from the "New" dialog. The assumed reason for the delay in the start up is network access. If this is the case then a progress dialog should be displayed to the user.

Along with this, the BML Data wizard assumes that there is an ever present connection to the KB. Where there is no connection present, the user cannot continue with the process of creating a DBE service, even if that user has all the required resources in his/her project. It should be made possible so a user can supply a locally stored BML model as a basis for the users BML data population efforts. This request is echoed in the points above (see section 6.2.1).

When the BML Data wizard cannot create a connection to the KB, the container showing what KBs are available should have a description that is not described in a pseudo-Hungarian-notion fashion. Suggestion is to rename this to "KB not reachable".

6.2.2 BPEL Wizard

The BPEL wizard is a wizard to assist the user in creating a workflow definition of cooperating DBE services and outputs a skeleton workflow using BPEL to represent the definition. This wizard consists of four wizard pages. The resulting output of this wizard is loaded up in the BPEL Editor

(see section 4.7). What follows is an evaluation of all the pages contained within this wizard, starting with the initial wizard page that is introduced to the user (Figure 70).

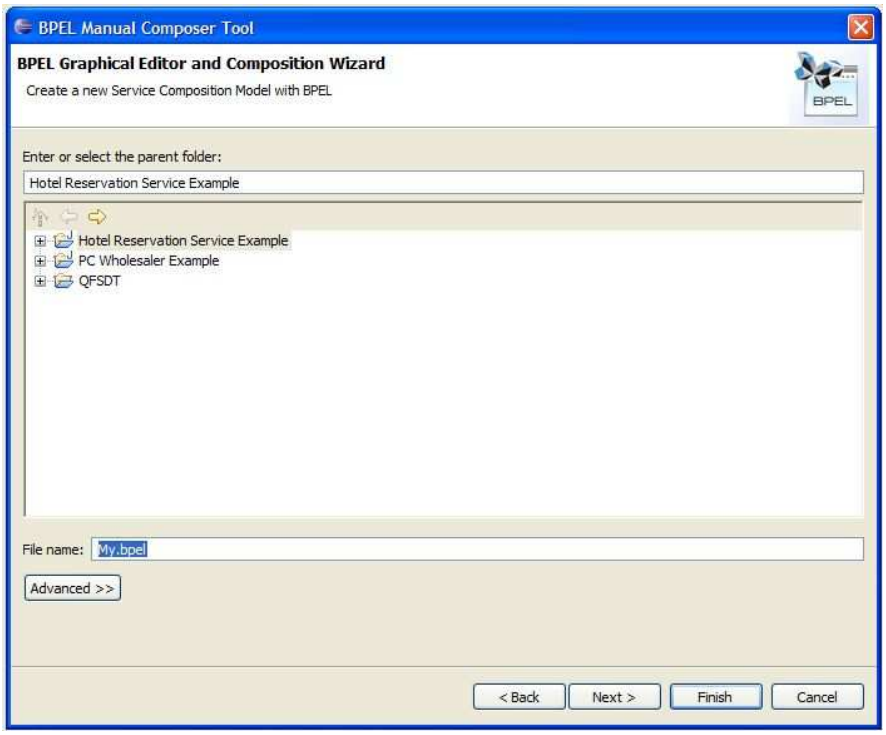


Figure 70 DBE Studio Wizards – BPEL Wizard, Page 1

With the first page of the DBE Studio BPEL Wizard visually introduced, an evaluation of it is now presented:

- The title of the wizard window should be renamed to "BPEL Wizard"

On clicking "Next" the second page (Figure 71) of the DBE BPEL Wizard is displayed to the user.

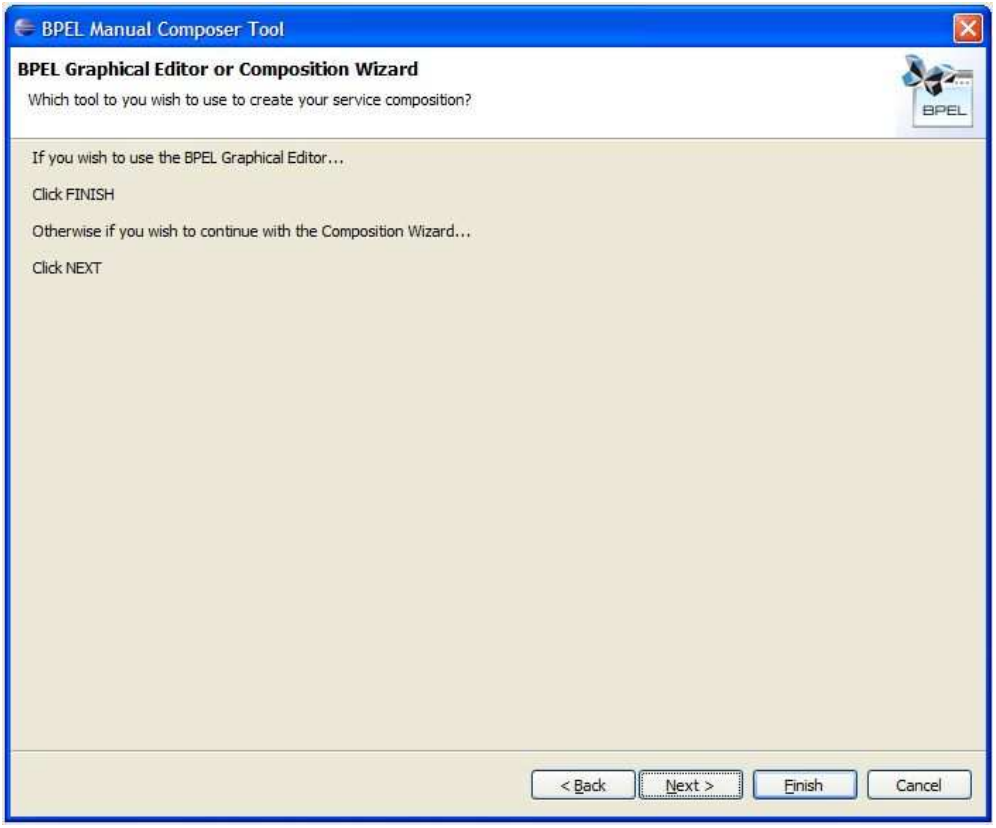


Figure 71 DBE Studio Wizards – BPEL Wizard, Page 2

With the second page of the DBE Studio BPEL Wizard visually introduced, an evaluation of it is now presented:

- This page is unnecessary as it adds no value or functionality to the wizard. The recommendation is to remove this page from the wizard completely.

On clicking "Next" the third page (Figure 72) of the DBE BPEL Wizard is displayed to the user.

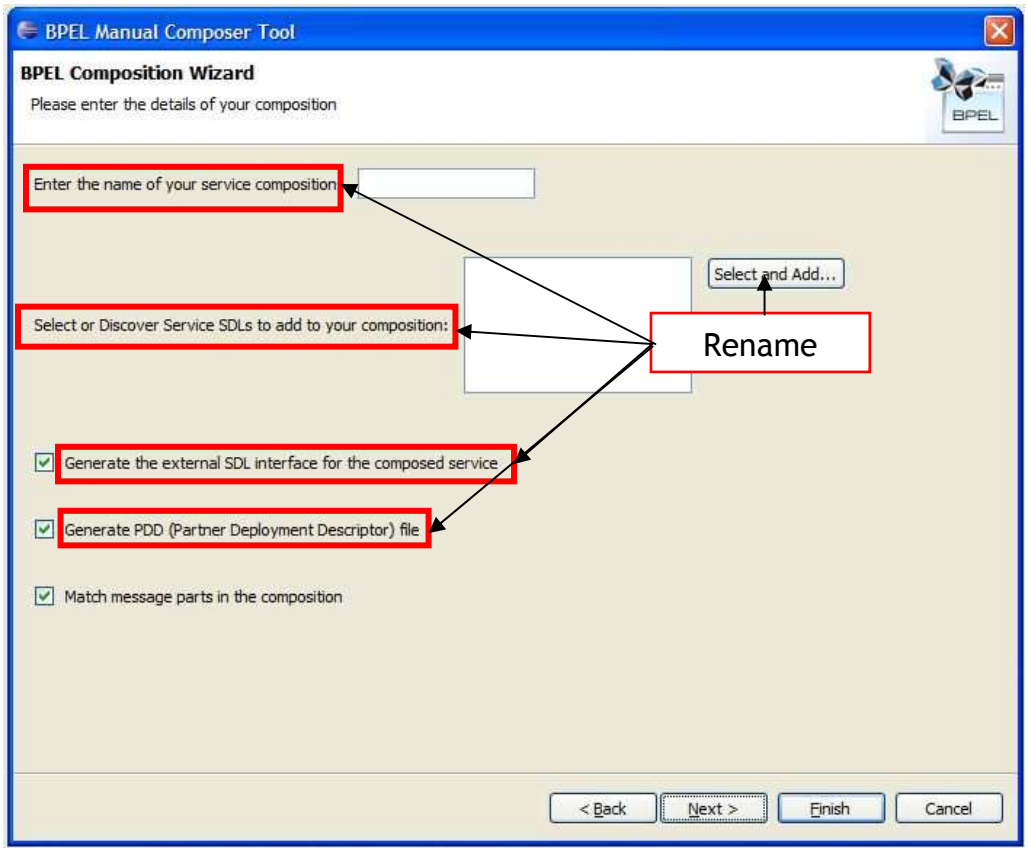


Figure 72 DBE Studio Wizards – BPEL Wizard, Page 3

With the third page of the DBE Studio BPEL Wizard visually introduced, an evaluation of it is now presented:

- No user input validation is performed on this wizard page in order to assure that data entered is correct
- The space and layout of components on this page should be reorganised.
- The text fields highlighted in red need to be terser in their description as they occupy too much space. Suggested text are:
    - "Enter the name of your service composition" – "Service Composition Name"

  - "Select or Discover Service SDLs to add to your composition" – "SDL Interfaces"
  - "Generate the external SDL interface for the composition" – "Generate Composition Interface"
  - "Generate PDD (Partner Deployment Descriptor) file" – "Generate PDD"
- The position of the text description field ("Select or Discover Service SDLs to add to your composition") needs to be placed above the input box and its text be justified to the left.
- The position of the button named "Select and Add…" needs to be moved to bottom of the list box, on the list box's east side. This button should also be renamed to "Add…"
- A remove button, to allow entries in the input box to be removed, needs to be added below the "Select and Add…" button.
- The list box, if possible, should have a descriptive column header describing the contents of it.
- The three checkboxes should be grouped as list box with checkable entries and that list box titled appropriately. A suggestion is "Composition Options". These check boxes should not be checked by default when appropriate.
- Until all required information is supplied the "next" buttons should not be enabled

On clicking "Next" the fourth page (Figure 73) of the DBE BPEL Wizard is displayed to the user.

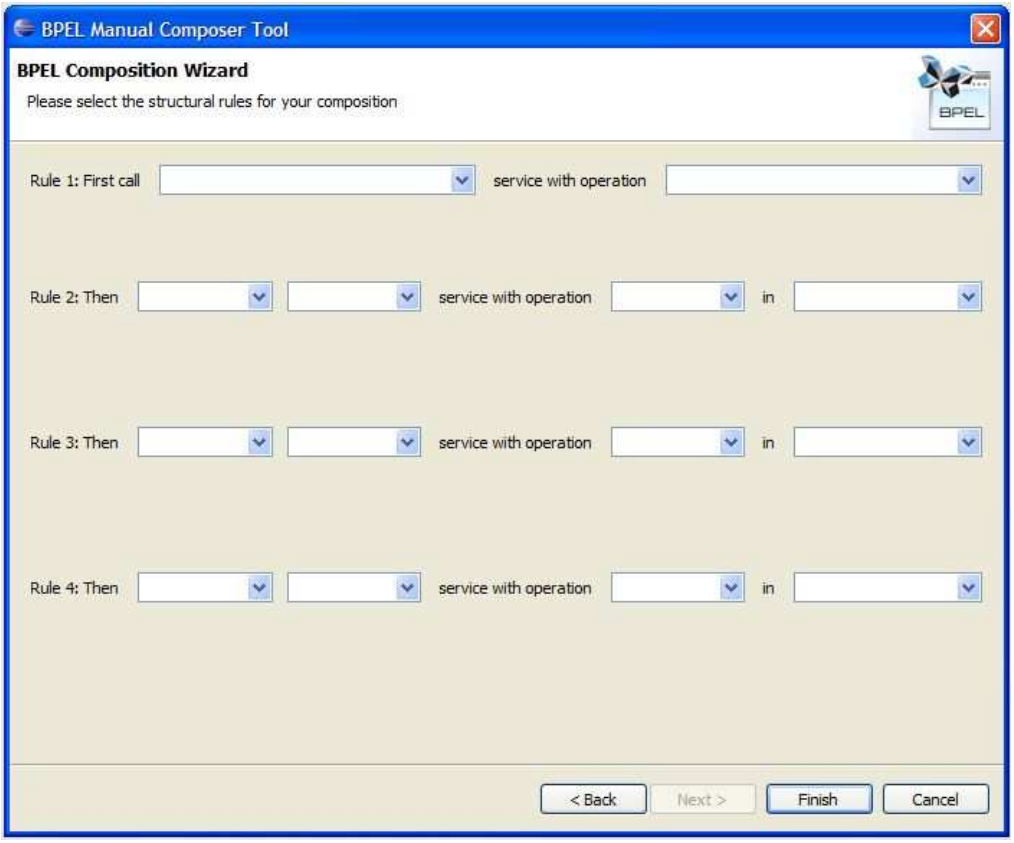


Figure 73 DBE Studio Wizards – BPEL Wizard, Page 4

With the fourth page of the DBE Studio BPEL Wizard visually introduced, an evaluation of it is now presented:

- By default, everything within this wizard page is enabled. This should not be allowed as this may only encourage user data input errors. As each rule set is used then and only then should the preceding rule set be enabled.
- A major limitation with this page is that there is only a finite amount of rules that can be created. A better scheme for presenting multiple rules would be to allow the developer to add new rules in a dynamic fashion and not be limited. An example of how such an interface could be implemented is the rule set editor in the Thunderbird Mozilla mail client as shown below:

Figure 74 DBE Studio Wizards – Rule Set Editor From Mozilla Thunderbird

- Until all required information is supplied the "next" and "finish" buttons should not be enabled.

6.2.3 DBE Project Wizard

The DBE Project wizard is a wizard to assist the user in the creation of a new DBE project and layout structure. This wizard consists of two wizard pages and when complete, an appropriate perspective is opened. What follows is an evaluation of all the pages contained within this wizard, starting with the initial wizard page that is introduced to the user (Figure 75).

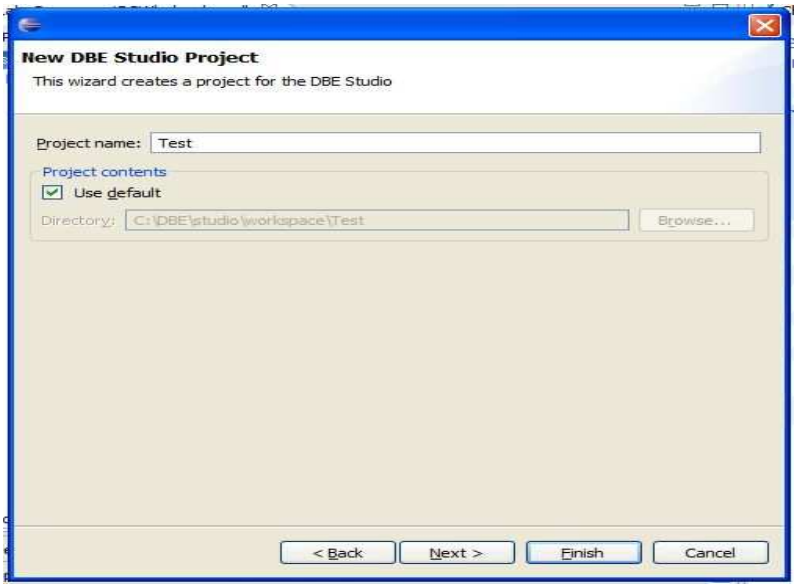


Figure 75 DBE Studio Wizards – DBE Project Wizard, Page 1

With the first page of the DBE Project Wizard visually introduced, an evaluation of it is now presented:

- The wizard title should be ideally populated and not left blank.
- If a user is to press "Finish" at this page then all directories for all DBE Studio tasks listed in the second wizard page (Figure 76) should be created.
- There is space wasted within this page and the second page's (Figure 76) content could be inserted here to reduce the number of pages the user comes in contact with.
- The wizard page needs a title graphic on the top right hand side.

On clicking "Next" the second page (Figure 76) of the DBE Project Wizard is displayed to the user.

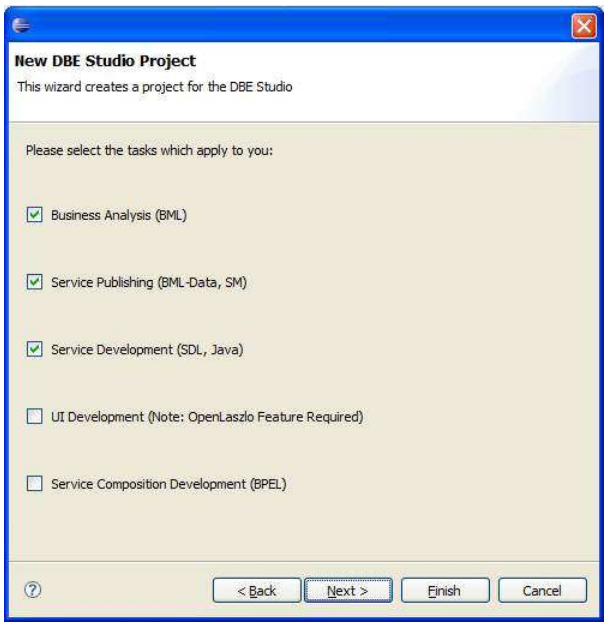


Figure 76 DBE Studio Wizards – DBE Project Wizard, Page 2

With the second page of the DBE Project Wizard visually introduced, an evaluation of it is now presented:

- If a user presses "Finish" on this page then only the directories related to the tasks as listed will be generated.
- The wizard title should be ideally populated and not left blank.
- The choices presented to the user in this page should be presented in a list box with elements selectable by check boxes.
- The contents of this wizard page could be merged with those of the first page in order to reduce the number of wizard pages a user needs to step through.
- Also a default choice of perspective should be given that allows the user change it
- The wizard page needs a title graphic on the top right hand side.

6.2.4 Query Wizard

The Query wizard is a wizard to assist the user in creating semantic queries based on query templates. This wizard consists of one wizard page and as soon as the user clicks on the "Finish" button of the wizard, the query is displayed in its editor. What follows is an evaluation of all the pages contained within this wizard, starting with the initial wizard page that is introduced to the user (Figure 77).

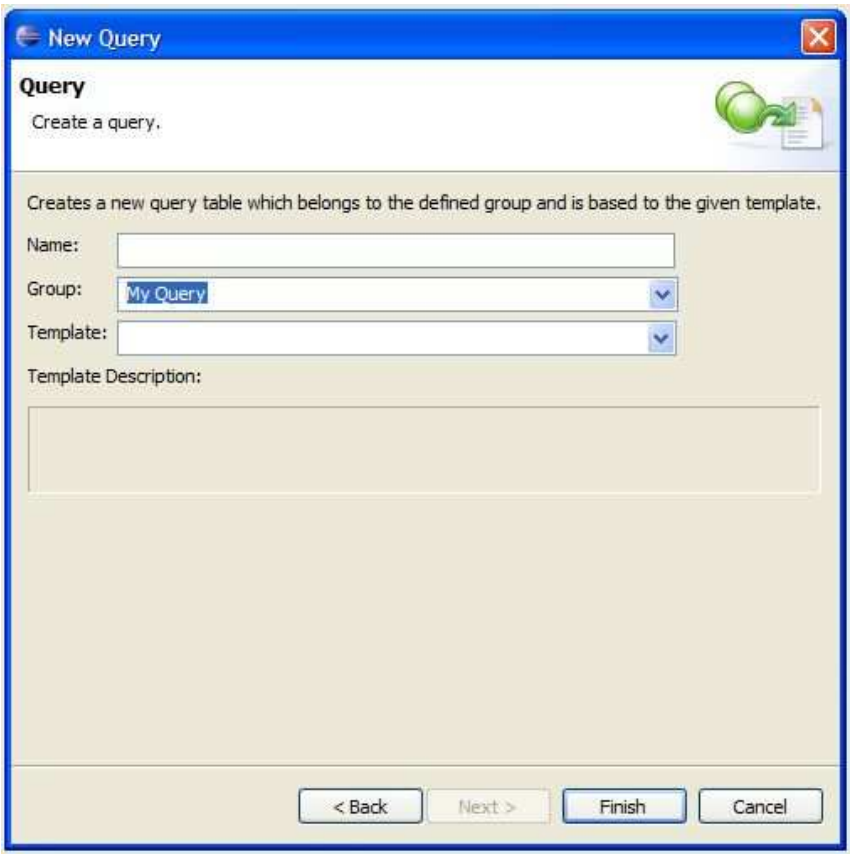


Figure 77 DBE Studio Wizards – Query Wizard

With the first page of the Query wizard visually introduced, an evaluation of it is now presented:

- Ideally, the "Finish" button should not be enabled by default until a name and "Template" is supplied
- With reference to 2.4, ideally queries should be saved on a per-project basis and not in a separate project - the DBE project is the core concept that binds all data related to a project in the DBE Studio

6.2.5 Query Template Wizard

The Query Template wizard is a wizard to assist the user in creating semantic query templates. This wizard consists of two wizard pages and as soon as the user clicks on the "Finish" button of the wizard, the template is saved and control returns back to the semantic discovery perspective. What follows is an evaluation of all the pages contained within this wizard, starting with the initial wizard page that is introduced to the user (Figure 78).

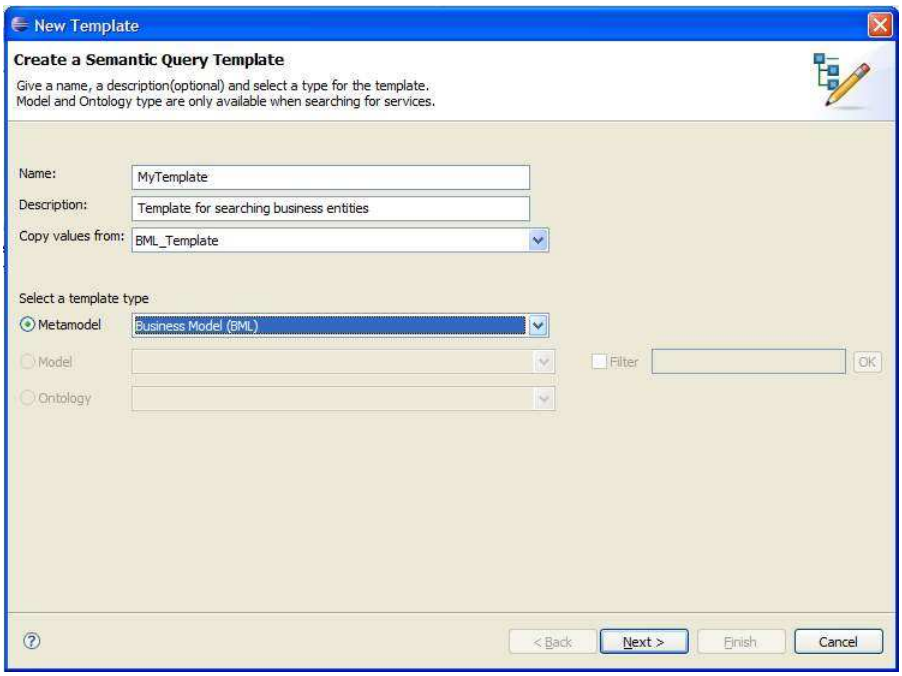


Figure 78 DBE Studio Wizards – Query Template, Page 1

With the first page of the Query template wizard visually introduced, an evaluation of it is now presented:

- There are no issues associated with the presented dialog.

On clicking "Next" the second page (Figure 79) of the Query Template Wizard is displayed to the user.

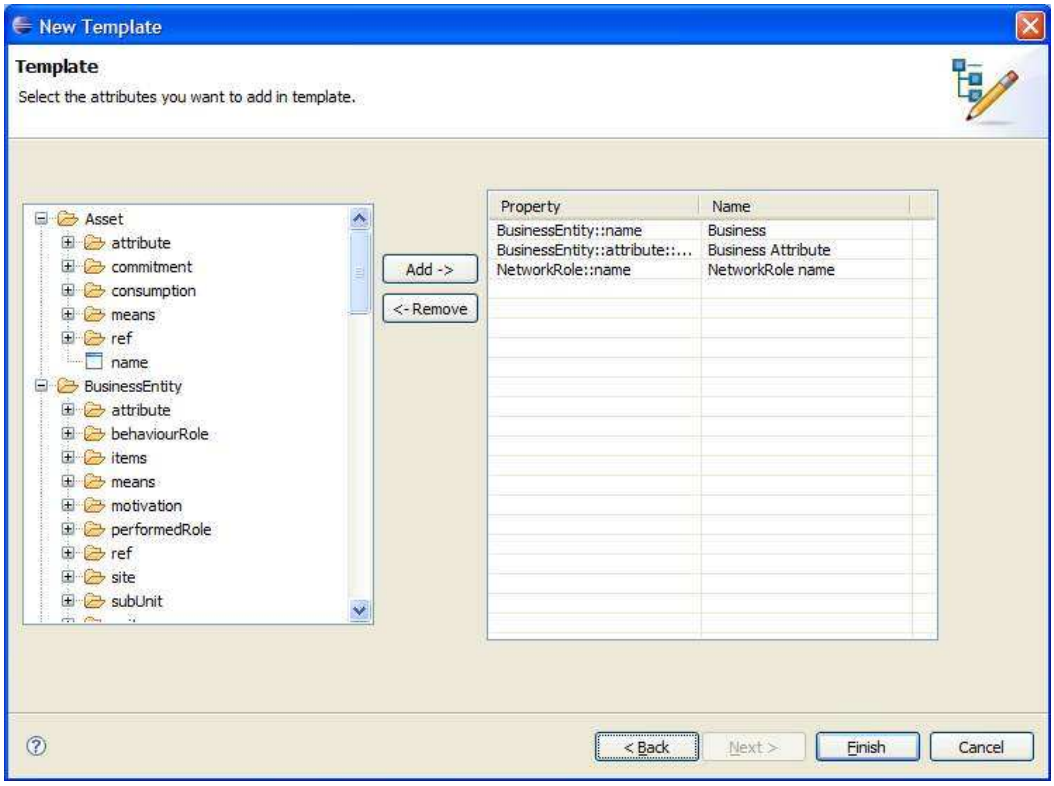


Figure 79 DBE Studio Wizards – Query Template, Page 2

With the second page of the Query Template Wizard visually introduced, an evaluation of it is now presented:

- The wizard page satisfies all thirteen perspective guidelines found in Table 8. It also satisfies this guideline from [1] "Slush Bucket widget (or Twin Box) should flow from left to right

with the source objects on the left hand side. It should have the >, <, >>, << control buttons in this order." Although the guideline does specify that all control buttons should be included it is not valid in the case of the query template wizard.

6.2.5.1 Connection Management Issues

Further to 2.3, within the query template wizard further consideration needs to be given to how connecting to network resources is handled. There is currently a requirement to be connected to a KB in order to populate the second page with BML attributes. This issue is also evident in section 5.5.1. A mechanism that allows the working with query templates without a network connection should be investigated. Perhaps it is possible to have a local catalog of BML attributes, much the way XML editors implement validation of XML documents using local XML schemas that are normally an external URI.

6.2.6 PDD Wizard

The PDD wizard is a wizard to assist the user in creating a partner deployment descriptor [REF] file. This wizard consists of two wizard pages and as soon as the user clicks on the "Finish" button of the wizard, a basic skeleton PDD file is created and loaded in the PDD editor (4.8). What follows is an evaluation of all the pages contained within this wizard, starting with the initial wizard page that is introduced to the user (Figure 80).

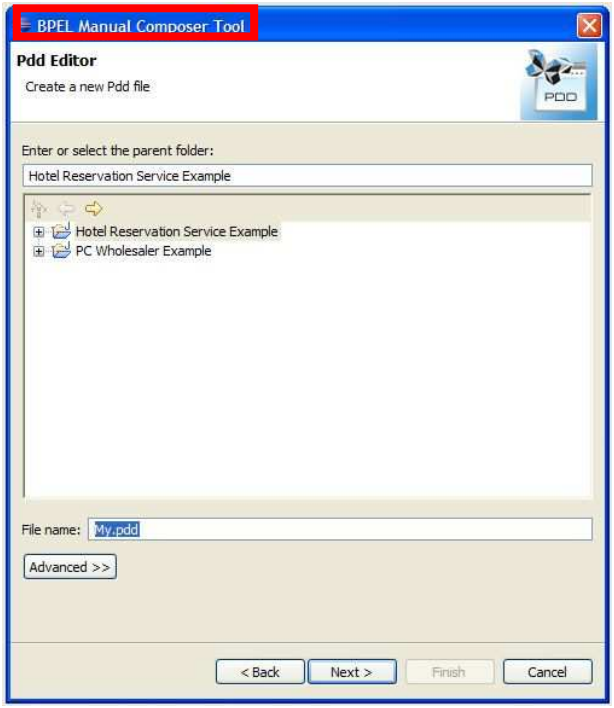


Figure 80 DBE Studio Wizard – PDD Wizard, Page 1

With the first page of the Query wizard visually introduced, an evaluation of it is now presented:

- The title, highlighted in red, should read "PDD Wizard"
- The Wizard description of "PDD Editor" is inaccurate and should be renamed to reflect what the wizard's purpose is.

On clicking "Next" the second page (Figure 81) of the PDD Wizard is displayed to the user.

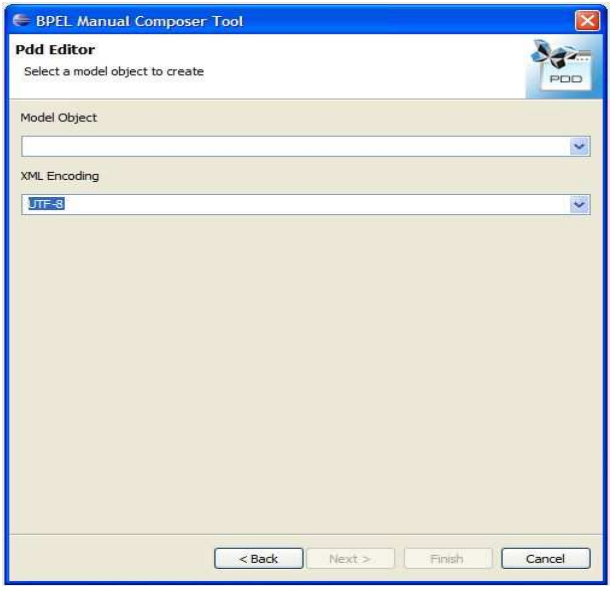


Figure 81 DBE Studio Wizard – PDD Wizard, Page 2

With the second page of the PDD wizard visually introduced, an evaluation of it is now presented:

- This page is confusing to the user and a result of the PDD editor being automatically generated using the Eclipse Modeling Framework. This page should be removed and the choices asked of the user by it coded into the implementation of the wizard.

6.2.7 SDL Wizard

The SDL wizard is a wizard to assist the user in creating service definition language (SDL) interfaces. This wizard consists of two wizard pages and as soon as the user clicks on the "Finish" button of the wizard, the generated SDL is opened in the SDL editor (4.6). What follows is an evaluation of all the pages contained within this wizard, starting with the initial wizard page that is introduced to the user (Figure 82).

Figure 82 DBE Studio Wizard – SDL Wizard, Page 1

With the first page of the Query wizard visually introduced, an evaluation of it is now presented:

- The title of the SDL wizard should be named "SDL Wizard".

On clicking "Next" the second page (Figure 83) of the PDD Wizard is displayed to the user.

Figure 83 DBE Studio Wizard – SDL Wizard, Page 2

With the second page of the SDL wizard visually introduced, an evaluation of it is now presented:

- This page is confusing to the user and a result of the SDL editor being automatically generated using EMF [REF]. This page should be removed and the choices asked of the user by it coded into the implementation of the wizard.

6.2.8 Service Manifest Wizard

The Service Manifest wizard is a wizard to assist the user in the creation of service manifests. This wizard consists of one wizard page and as soon as the user clicks on the "Finish" button of the wizard, the service manifest editor is opened to edit the newly created SM file. What follows is an evaluation of all the pages contained within this wizard, starting with the initial wizard page that is introduced to the user (Figure 84).

Figure 84 DBE Studio Wizard – Service Manifest Wizard, Page 1

With the only page of the Service Manifest wizard visually introduced, an evaluation of it is now presented:

- The wizard title should be ideally populated and not left blank.
- The initial name of the service manifest could be taken from the enclosing project name.
- As the minimum requirements of a service manifest is a BML model, BML data and optionally SDL, it should be possible to add these files from the workspace automatically so the user does not have to do it. This is possible only where all DBE Studio plugins adhere to the recommendations set out in section 2.4 and section 2.5. It is recommended that the service manifest wizard implement this suggestion.

6.2.9 Test Scenario Wizard

The Test Scenario wizard is a wizard to assist the user in creating a test scenario to run a set of test cases, generated by the test case generator (section 6.3.4). This wizard consists of two wizard pages and as soon as the user clicks on the "Finish" button of the wizard, the generated test scenario is opened in the Java editor. What follows is an evaluation of all the pages contained within this wizard, starting with the initial wizard page that is introduced to the user (Figure 85).

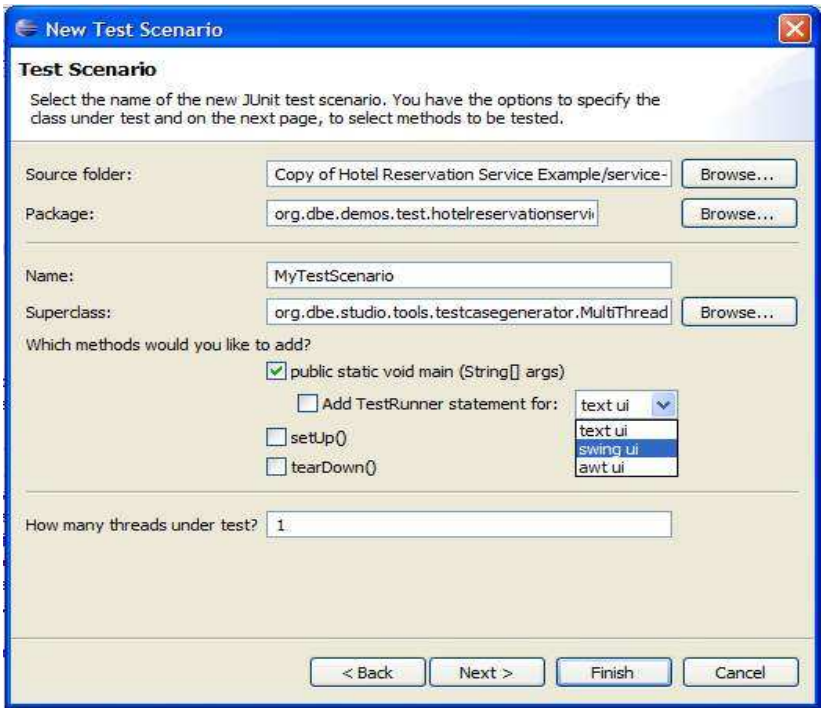


Figure 85 DBE Studio Wizards – Test Scenario Wizard, Page 1

With the first page of the test scenario wizard visually introduced, an evaluation of it is now presented:

- Before any dialog is shown, the user is prompted to add 2 libraries to the projects class path. This is done with 2 consecutive popup dialog boxes. This would be better presented as an extra wizard page before the current initial wizard page (Figure 85).
- An explanatory page at the beginning of this wizard explaining that first test cases are need to be generated using the service test case generator before this test scenario is created. If this could be automated then this again would improve on the user's experience.

On clicking "Next", the second page (Figure 83) of the test scenario wizard is displayed to the user.

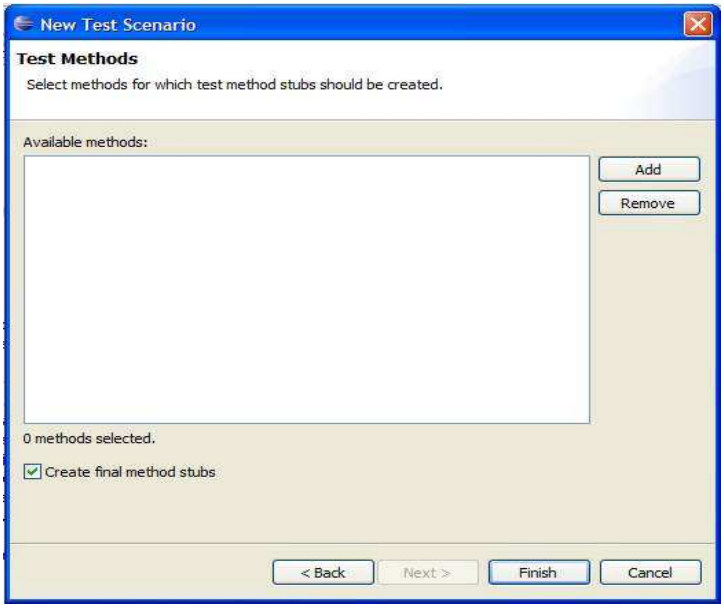


Figure 86 DBE Studio Wizard – Test Scenario Wizard page 2

With the second page of the test scenario wizard visually introduced, an evaluation of it is now presented:

- The add button should be titled as "Add…"
- The "Finish" button should not be enabled until the user has supplied something to the list box.

6.3 Contextual Right-Click Wizards

These wizards are displayed to the user when the user right-clicks on a target file. A contextual menu is then presented and the user selects and applies the appropriate action from the contextual menu. Within the DBE Studio these should all be organized under a common contextual menu and as a sub-menu of the root context menu so users are familiarized with where these types of wizards are located within the DBE Studio.

There are contextual right-click wizards that do not organize themselves under this DBE Studio sub-menu. Those wizards are only the test case generator wizard (Figure 95). It should be placed under this sub-menu.

6.3.1 SSL2SDL

The SSL2SDL wizard is a wizard to assist the user in generating a SDL interface from the SSL contained in a BML model. This wizard does not present itself to the user as a conventional wizard but rather a sequence of dialog boxes that pop up to request information from the user. This is not the recommended approach of creating resources using Eclipse wizards and not the approach taken in the DBE Studio. As such, the recommendation for this wizard to be presented as an Eclipse-style wizard (and hence DBE Studio-style) in order to present a uniform and consistent experience to the user. A suggested wizard for the provisioning of SSL to SDL services in the DBE Studio would be as follows:

- 1st page – Project container selection.
- 2nd page – Generation options.

What is available to the user is accessed by right clicking on an appropriate BML model file and navigating to "SSL2SDL Compiler" in the contextual menu as shown in the below diagram Figure 87.

Figure 87 DBE Studio Wizards – SSL2SDL Wizard, Menu Entry

A potential issue was noticed when using this contextual tool. Currently BML models are stored as both .bml and .xmi files with the same name in the same folder. A user without much experience with the tool could easily select a .bml file and attempt SDL generation with it. Unfortunately for the

user he will be faced with an error. In order to negate such user errors, the SSL2SDL tool should only be available on .xmi file types.

After clicking on "Generate SDL File" the following modal dialog box appears (Figure 88).

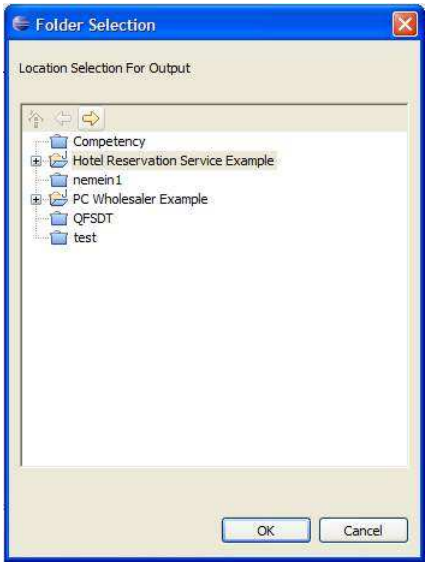


Figure 88 DBE Studio Wizards – SSL2SDL

From a usability point of view with regards to dialogs, this particular dialog has no particular issues, however this dialog should be merged with a wizard that implements the same functionality of the SSL2SDL tool.

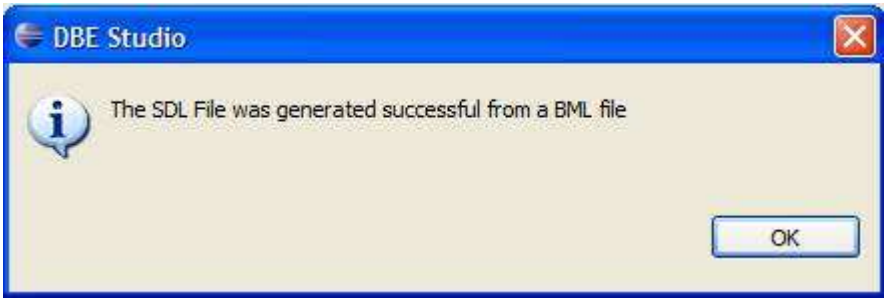


Figure 89 DBE Studio Wizards – SSL2Java

When the SSL2SDL tool has successfully transformed the SSL to SDL the above modal dialog is presented to the user.

- The reference to BML, although correct, could be misleading as the user knows this tool to generate SDL from SSL and not BML.
- In addition, the title of the dialog box is incorrect and should read like "SSL2SDL Tool".
- Ideally, the status of SSL generation should be shown to the user using progress bars.

6.3.2 SDL2Java

The SDL2Java wizard is a wizard to assist the user in generating a Java skeleton and interfaces with which the user can implement the business logic of a DBE service. This wizard does not present itself to the user as a conventional wizard but rather a sequence of dialog boxes that pop up to request information from the user. This is not the recommended approach of creating resources using Eclipse wizards and not the approach taken in the DBE Studio. As such, the recommendation for this wizard to be presented as an Eclipse-style wizard (and hence DBE Studio-style) in order to present a uniform and consistent experience to the user. A suggested wizard for the provisioning of SDL to Java services in the DBE Studio would be as follows:

- 1st page – options and generation

Adopting this approach also has the advantage that the options for namespace (see 8.7) would then no longer be hidden from the user of the tool.

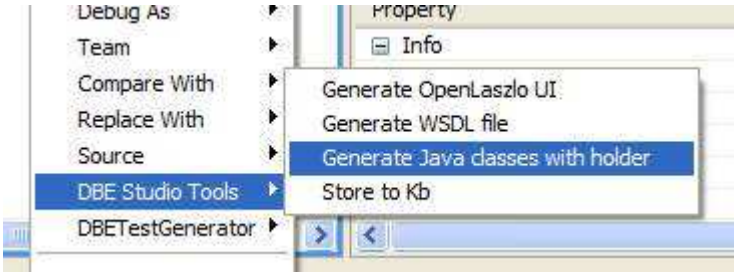


Figure 90 DBE Studio Wizards – SDL2Java

The contextual menu entry shown above should be renamed to "Generate Java Classes…". After clicking on "Generate SDL File" the following modal dialog box appears (Figure 91).

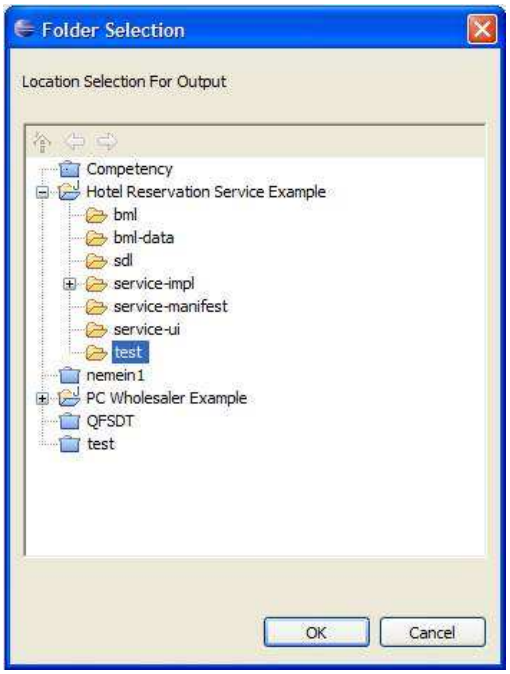


Figure 91 DBE Studio Wizards - SDL2Java

The above modal dialog asks the user to select a project into which the generated Java source files will be placed. It gives the option to navigate to sub-folders. However, if a user selects a sub-folder into where the user wants the SDL file placed, the SDL2Java tool generates no source with a message stating that the generation was successful.

From a usability point of view with regards to dialogs, this particular dialog has no particular issues, however this dialog should be merged with a wizard that implements the same functionality of the SDL2Java tool.

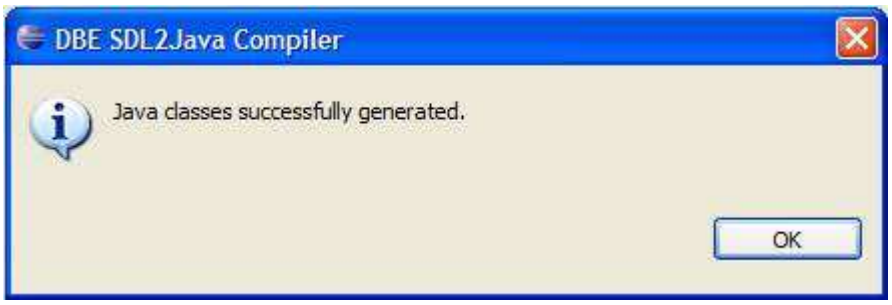


Figure 92 DBE Studio Wizards – SDL2Java

When the SDL2Java tool has successfully transformed the SDL to Java source files the above modal dialog is presented to the user. Rather than stating the Java classes have been generated it should say "Java source files successfully generated". The status of Java files generation should be shown to the user through the use of progress bars.

6.3.3 SDL2KB

The SDL2KB contextual right-click wizard allows a user to save a SDL model to the knowledge base by right-clicking on a SDL file and selecting the "Store to Kb" entry in the contextual menu (as shown in Figure 93.

Figure 93 DBE Studio Wizards – SDL2KB

On selecting the "Store to Kb" entry, the user is then presented with the following dialog (Figure 94)

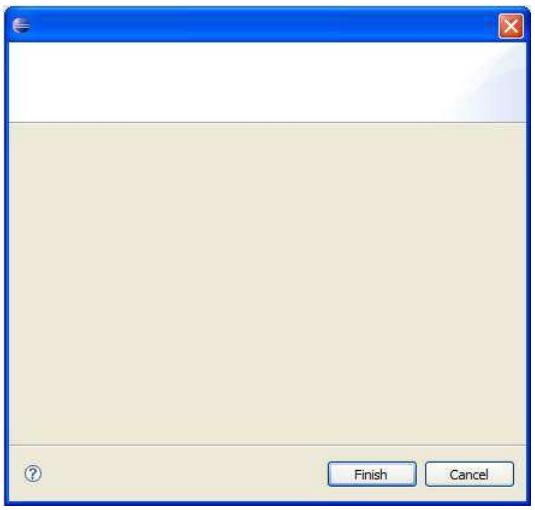


Figure 94DBE Studio Wizards – SDL2KB wizard

With the SDL2KB wizard presented its evaluation is now given:

- This wizard's main task is to export a resource from the user's workspace to some form of remote storage, in this case the KB. As such, how the wizard is presented to the user through a contextual right click is incorrect. This wizard should be exposed to the user as an export-type wizard such as the service exporter wizard (section 6.4.1). It is strongly recommended that this wizard follows suit. A consequential recommendation is to remove the contextual menu entry.
- The dialog that is revealed to the user upon selecting the SDL for export provides no details as to what the dialog will do as it is blank. This should have information informing what action will take place as soon as the user clicks on the "Finish" button. There is also no title to this wizard.

6.3.4 Test Case Generator

The test case generator wizard is a wizard to assist the user in generating Java test cases based on a SDL interface. This wizard does not present itself to the user as a conventional wizard but rather a sequence of dialog boxes that pop up to request information from the user. This is not the recommended approach of creating resources using Eclipse wizards and not the approach taken in the DBE Studio. As such, the recommendation is for this wizard to be presented as an Eclipse-style wizard (and hence DBE Studio-style) in order to present a uniform and consistent experience to the

user. A suggested wizard for the provisioning of test case generation services in the DBE Studio would be as follows:

Figure 95 DBE Studio Wizards – Test Case Generator

The contextual menu entry shown above should be placed as a sub-entry, named "Test Generator", into the menu group "DBE Services" and be renamed to "Generate Test Case…". After clicking on "Generate SDL File" the following modal dialog box appears (Figure 96).

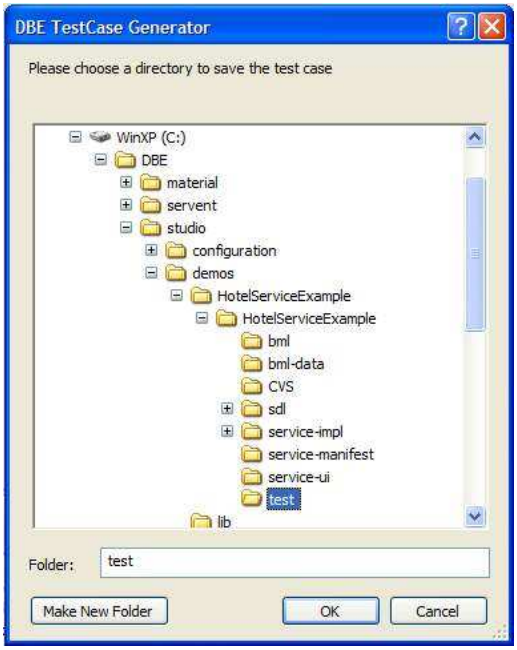


Figure 96 DBE Studio Wizards – Test Case Generator

The above modal dialog asks the user to select a project into which the generated Java source files will be placed. It gives the option to navigate to sub-folders. However, this dialog box is not Eclipse specific. It is recommended that a dialog box like the one in Figure 91 should be used instead. Following this dialog the user is asked by a number of dialog boxes (up to seven), an example of which is shown in Figure 97, to add required libraries so that the generated tests compile

successfully. This is quite repetitive and could be easily solved by asking the user to allow the addition of required dependencies with just one dialog box.

Figure 97 DBE Studio Wizards – Test Case Generator

From a usability point of view with regards to dialogs, this particular dialog has no particular issues, however this dialog should be merged with a wizard that implements the same functionality of the test case generator tool.

Figure 98 DBE Studio Wizards – Test Case Generator

When the test case generator tool has successfully created the test case, the above modal dialog is presented to the user. Ideally, the status of test case generation should be shown to the user using progress bars.

6.3.5 User Interface Generator

The user interface generator wizard is a wizard to assist the user in generating a skeleton OpenLaszlo [26] user interface using a SDL file as input. The user interface generator presents itself to the user when the user right-clicks on an SDL file (Figure 99)

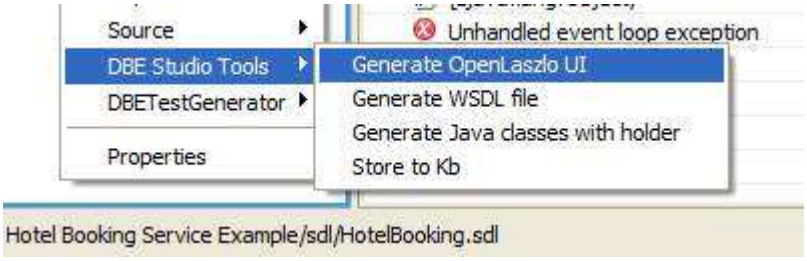


Figure 99 DBE Studio Wizards – User Interface Generator

Once the user clicks on "Generate OpenLaszlo UI" the following dialog is displayed:

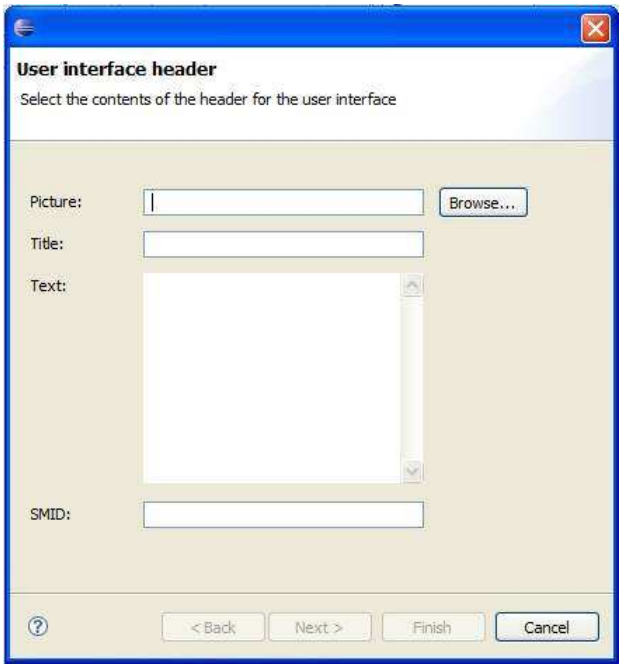


Figure 100 DBE Studio Wizards – User Interface Generator

From what is presented visually in the above diagram the evaluation of the UI is as follows:

- The wizard title should be ideally populated and not left blank.
- The information to populate the service manifest identifier (SMID) text input field is already present in the users workspace. This should be exploited and the field automatically populated.

After the user has supplied the requested the following UI is then displayed:

Figure 101 DBE Studio Wizards – User Interface Generator

From evaluating the above user interface it was evident to be a very simple interface. However, the wizard title should be ideally populated and not left blank .

6.4 Export/Import Wizards

These wizards are displayed to the user as a result of requesting the import or export of a resource through Eclipse's "File->Import" or "File->Export" dialogs or by right clicking within Eclipse's "Navigator" view and selecting the same named entries in the contextual menu. Before delving straight into particular instances of import/export wizards, there are a few comments that need to be made about their integration within the import and export dialogs (Figure 102).

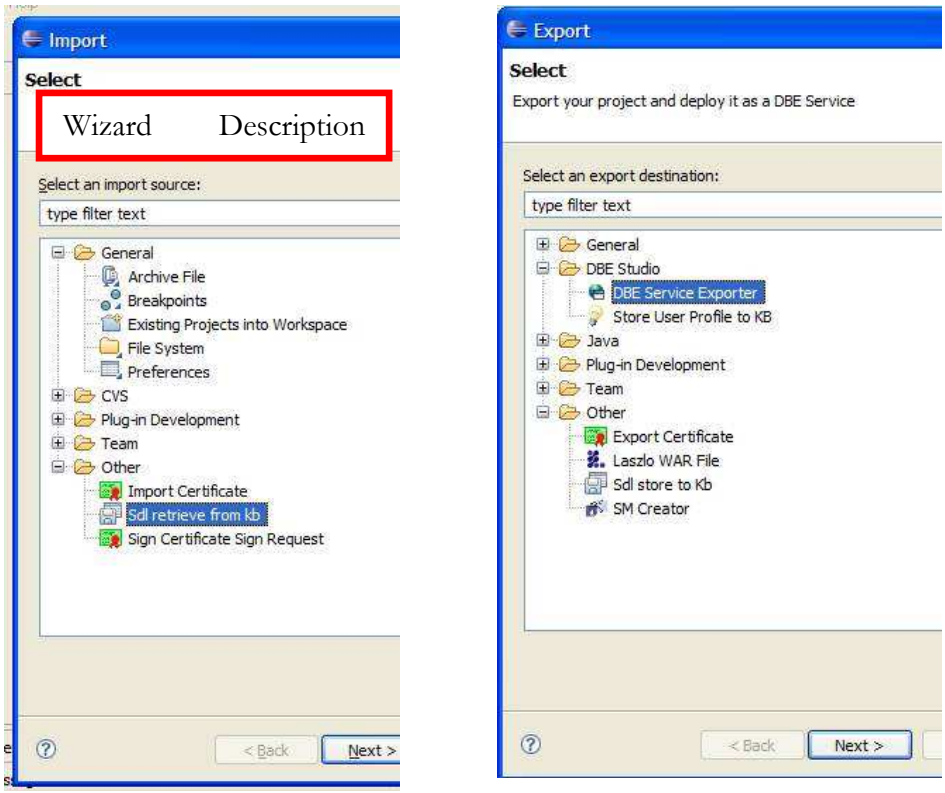


Figure 102 DBE Studio Wizards – Import and Export Wizards

With the import and export dialogs visually introduced, an evaluation of it is now presented:

- The "Sdl retrieve from kb" import plugin should be moved into the DBE Studio category.
- The "Sdl store to Kb" and "SM Creator" export plugins should be moved into the DBE Studio category.
- The text "Sdl retrieve from kb" should be renamed to "SDL Interface"
- The text "Sdl store to Kb" should be renamed to "SDL Interface"
- The text "Store User Profile to KB" should be renamed to "DBE User Profile"
- When each DBE import/export wizard is selected a description of the wizard should appear in the area as shown in the above diagram. The wizards that do not currently provide this are:
    - "Sdl retrieve from kb"
    - "Sdl store to Kb"
    - "Store User Profile to KB"

6.4.1 Service Exporter

The service exporter wizard is a wizard to assist the user in deploying the implementation of a DBE service to the servent. This wizard consists of two wizard pages. The result of this wizard is the successful deployment of a service implementation to the servent. This wizard also has the metering wizard (section 6.4.1.1) integrated. With this, users can attach metering at deployment time. What follows is an evaluation of all the pages contained within this wizard, starting with the initial wizard page that is introduced to the user (Figure 103).

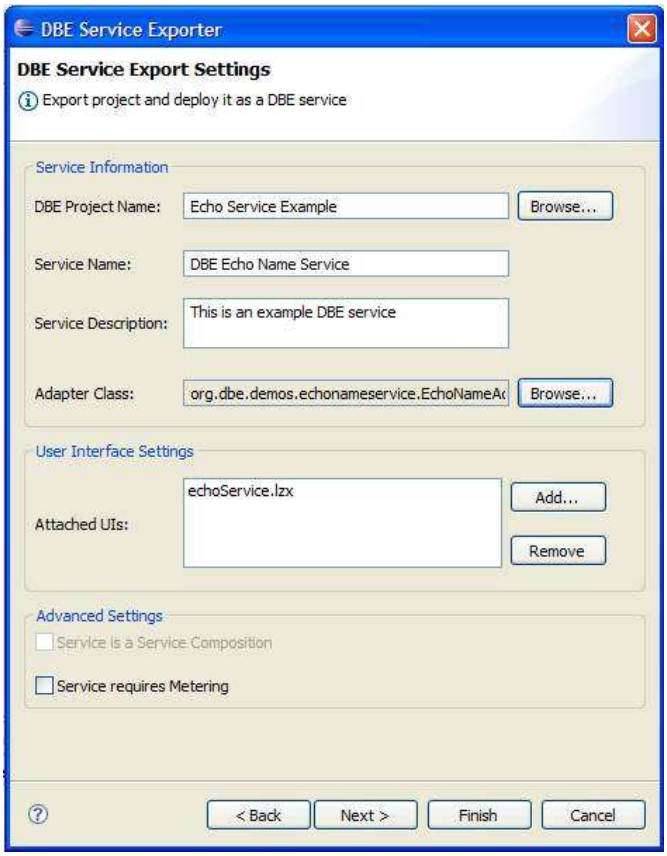


Figure 103 DBE Studio Wizard – Service Exporter, Page 1

With the first page of the DBE Studio service exporter wizard visually introduced, an evaluation of it is now presented:

- A graphic is required for the top right hand corner of this page.

On clicking "Next" the second page (Figure 104) of the DBE service exporter Wizard is displayed to the user.

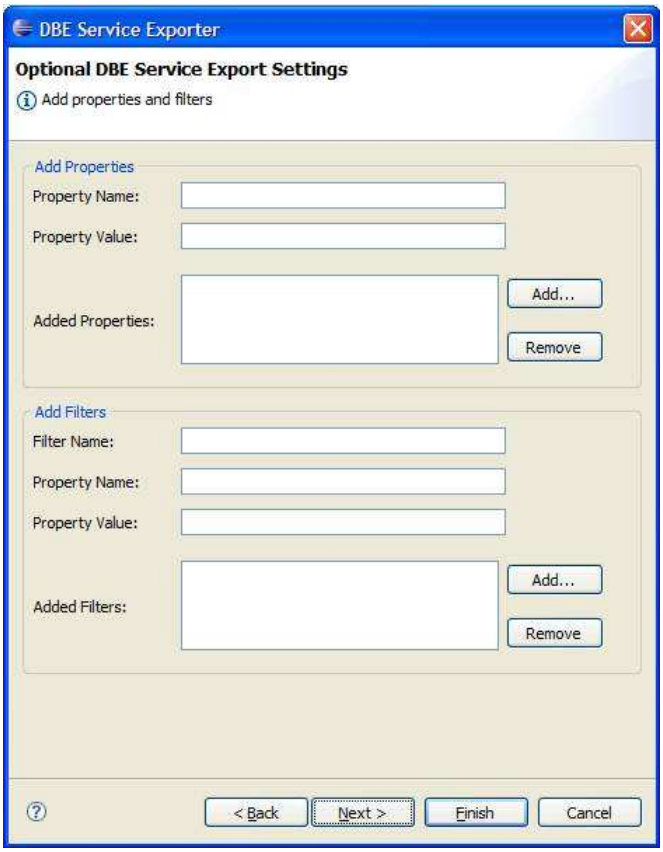


Figure 104 DBE Studio Wizard – Service Exporter, Page 2

With the second page of the DBE Studio service exporter wizard visually introduced, an evaluation of it is now presented:

- A graphic is required for the top right hand corner of this page.

6.4.1.1 Service Metering Wizard

The DBE service metering wizard is a wizard to assist the user in attaching metering filters to methods within a particular SDL interface. This wizard consists of one wizard page and is part of the

DBE Studio Service Exporter. The result of this wizard is the addition of directives in the deployment descriptor file telling the servent to attach metering filters.

What follows is an evaluation of all the pages contained within this wizard, starting with the initial wizard page that is introduced to the user (Figure 103).

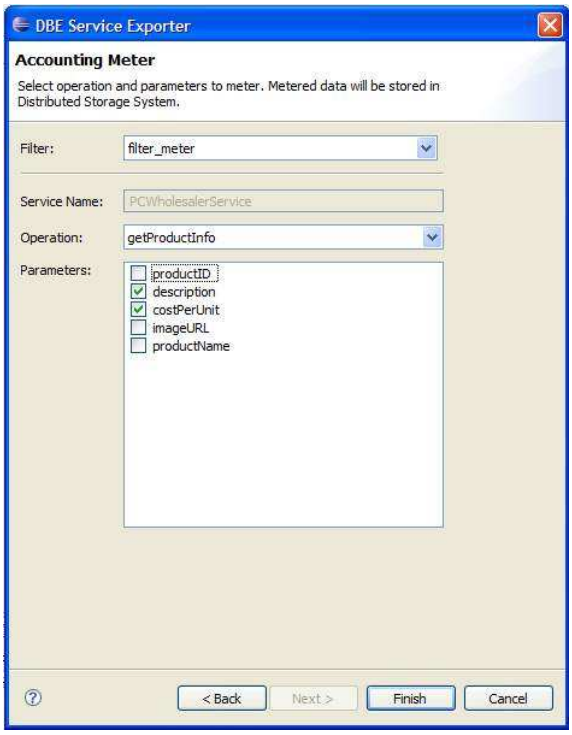


Figure 105 DBE Studio Wizard – Metering Wizard, Page 1

With the first page of the Query wizard visually introduced, an evaluation of it is now presented:

- A graphic is required for the top right hand corner of this page.

6.4.2 SDL Import

The SDL import wizard is a wizard to assist the user in importing a specified SDL file from a KB. This wizard consists of three wizard pages. The result of this wizard is the successful import of the

SDL file specified. What follows is an evaluation of all the pages contained within this wizard, starting with the initial wizard page that is introduced to the user (Figure 106).

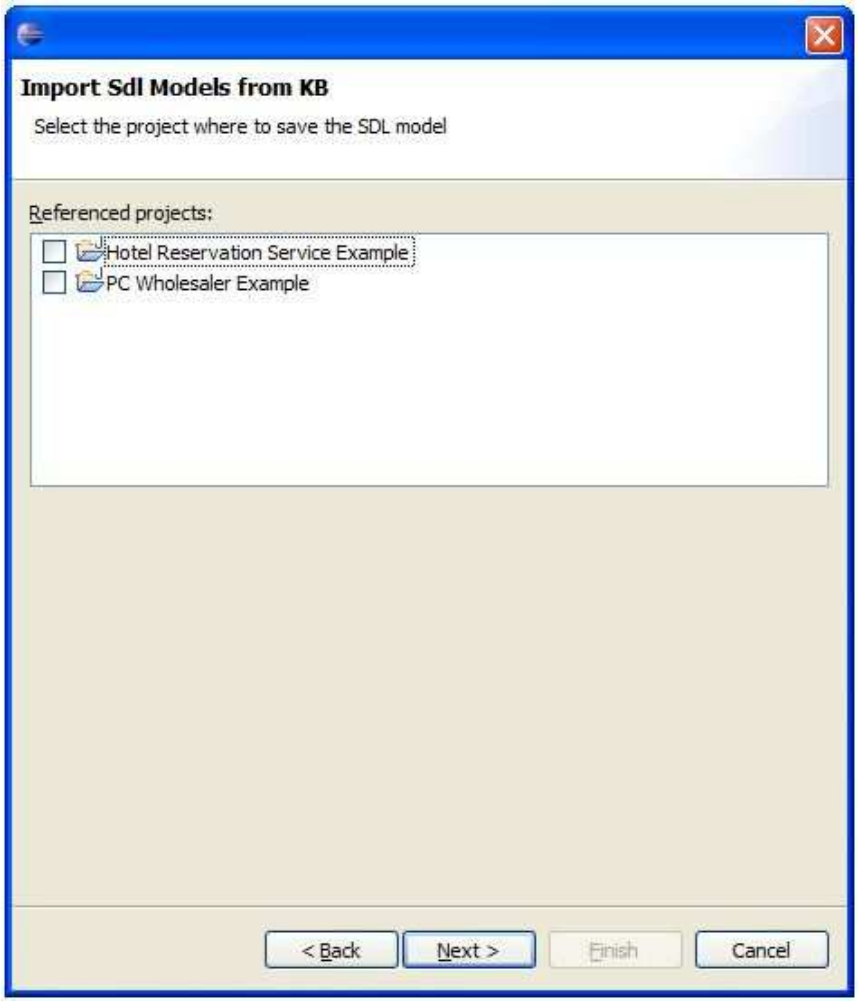


Figure 106 DBE Studio Wizard – SDL Import, Page 1

With the first page of the DBE Studio SDL import wizard visually introduced, an evaluation of it is now presented:

- The wizard title should be ideally populated and not left blank.
- To save a SDL file, the wizard should use a mechanism similar to Figure 70 as there is a lot of unused space that could be optimized.

- Users should be able to browse for a specific location within their project workspace so that they can save the imported SDL file there.

On clicking "Next" the second page (Figure 107) of the DBE SDL import wizard is displayed to the user.

Figure 107 DBE Studio Wizard – SDL Import, Page 2

With the second page of the DBE Studio SDL import wizard visually introduced, an evaluation of it is now presented:

- The wizard title should be ideally populated and not left blank.
- The main wizard description reads as "WSDL Export/ SDL Settings". It is recommended that this be changed to better reflect the purpose of the presented wizard page.

- The hint at the top of the wizard shows an error and this is not in line with WG3. This should be displayed to the user as an informational prompt, not an error.
- This page is potentially confusing to a user as it offers no real clue as to what is required of the user.
- The text "Definition name" should be replaced with "SDL Definition Name" and the same for similar references within the page.
- This page could be merged with the first page and so reducing the number of steps required by a user to complete the wizard.
- The text label "FADA node" is not relevant and should be removed.

6.4.3 SDL Export

At the time of writing it was not possible to review the 0.3.0 version of the SDL to KB plugin [27]. What follows is the analysis is based on the 0.2.x version. The SDL Export wizard is a wizard to assist the user in exporting a specified SDL file to a KB. This wizard consists of two wizard pages. The resulting output of this wizard is successful export of the specified SDL to a KB. What follows is an evaluation of all the pages contained within this wizard, starting with the initial wizard page that is introduced to the user (Figure 108).

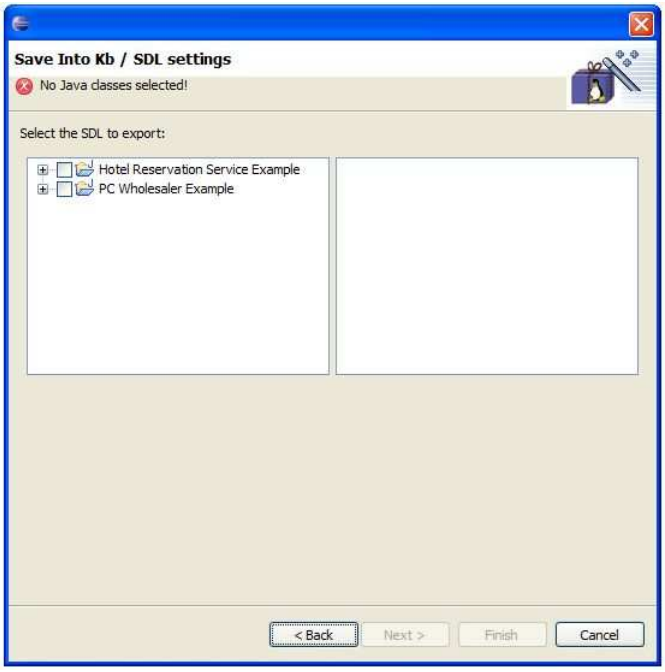


Figure 108 DBE Studio Wizard – SDL Export, Page 1

With the first page of the DBE Studio SDL export Wizard visually introduced, an evaluation of it is now presented:

- The wizard title should be ideally populated and not left blank.
- The hint at the top of the wizard shows an error and this is not in line with WG3. This should be displayed to the user as an informational prompt, not an error
- The wizard description needs to be shortened. A suggestion would be "Export to KB Wizard Settings".
- To browse for a SDL file the wizard should use a mechanism similar to Figure 70 as there is a lot of unused space that could be optimized.

7 Documentation

In this chapter the documentation contained within the DBE Studio are analyzed and recommendations are made upon that analysis. Within the DBE Studio documentation is provided

by two main mechanisms. The first is using the in-built Eclipse help system. The second is using cheat-sheets, which are task specific helping guides. Ideally, all tools (perspectives, editors and views) contained within the DBE Studio should provide help documentation and that documentation be easily accessible.

7.1 Help

The help provided in the DBE Studio is essentially a set of user guides for DBE Studio tools. These can be accessed using the main Eclipse help menu. The help provided is accessed from the main menu entry "Help->Help Contents…". This will open the following window that allows user browse and search help.

Figure 109 DBE Studio Documentation - Help

Help documentation is a basic requirement of any software tool that is used extensively by users. Below is a table of all the discussed perspectives, editors, views and wizards contained within the DBE Studio and whether or not they provide sufficient help documentation. It is **strongly** suggested that all plugin developers contribute to the help plugin with appropriate user documentation.

	Name	Partner	Help Doc. Provided
Perspectives	Ontology analysis	Technical University of Crete	Yes
	Business Analysis	Technical University of Crete	Yes
	Semantic Discovery	Technical University of Crete	Yes
	Service Composition	Trinity College Dublin	Yes
	Service Development	Intel Ireland Ltd.	No
	Service Publishing	Intel Ireland Ltd.	No

Table 9 DBE Studio Documentation – Provided Perspective Help Documentation

	Name	Partner	Help Doc. Provided
Editors	Ontology Editor	Technical University of Crete	Yes
	BML Editor	Technical University of Crete	Yes
	BML Data Editor	University of Central England	No
	Query Editor	Technical University of Crete	Yes

	SDL Editor	Soluta	Yes
	BPEL Editor	Trinity College Dublin	Yes
	PDD Editor	Trinity College Dublin	Yes
	Service Manifest Editor	Soluta	No

Table 10 DBE Studio Documentation – Provided Editor Help Documentation

	Name	**Partner**	**Help Doc. Provided**
Views	Outline View	Technical University of Crete	Yes
	Model Navigator	Technical University of Crete	Yes
	Query Navigator	Technical University of Crete	Yes
	Property View	Technical University of Crete	Yes
	Query Properties	Technical University of Crete	Yes
	Ontology Viewer	Technical University of Crete	Yes
	Query Template	Technical University of Crete	Yes
	Query Results	Technical	Yes

		University of Crete	
	Keyword Search	Technical University of Crete	Yes

Table 11 DBE Studio Documentation – Provided View Help Documentation

	Name	**Partner**	**Help Doc. Provided**
Wizards	BML Data Wizard	University of Central England	No
	BPEL Wizard	Trinity College Dublin	Yes
	DBE Project Wizard	Intel Ireland Ltd.	Yes
	Query Wizard	Technical University of Crete	Yes
	Query Template Wizard	Technical University of Crete	Yes
	PDD Wizard	Trinity College Dublin	No
	SDL Wizard	Soluta	Yes
	Service Manifest Wizard	Soluta	No
	Test Scenario Wizard	University of Surrey	Yes
	SSL2SDL Wizard	Soluta	Yes
	SDL2KB	Soluta	No
	SDL2Java Wizard	Soluta	Yes
	Metering Wizard	Waterford Institute of Technology	Yes
	UI Generator	Intel Ireland Ltd.	No
	Test Case Generator	University of Surrey	Yes

	Service Exporter	Intel Ireland Ltd.	Yes
	SDL Import	Soluta	No
	SDL Export	Soluta	No

Table 12 DBE Studio Documentation – Provided Wizard Help Documentation

7.2 Cheat Sheets

A cheat sheet is an interactive tutorial which opens a view on the right hand side of the DBE Studio screen outlining steps on how to perform tasks within the DBE Studio. Currently there are three cheat sheets implemented in the DBE Studio (one is supplied with the DBE Studio examples). For the DBE Studio, the first is how to get started with the DBE Studio tools and the second one is a walk through on how to create a DBE service from beginning to end. These cheat sheets are accessed through the main menu entry "Help -> Cheat Sheets…".

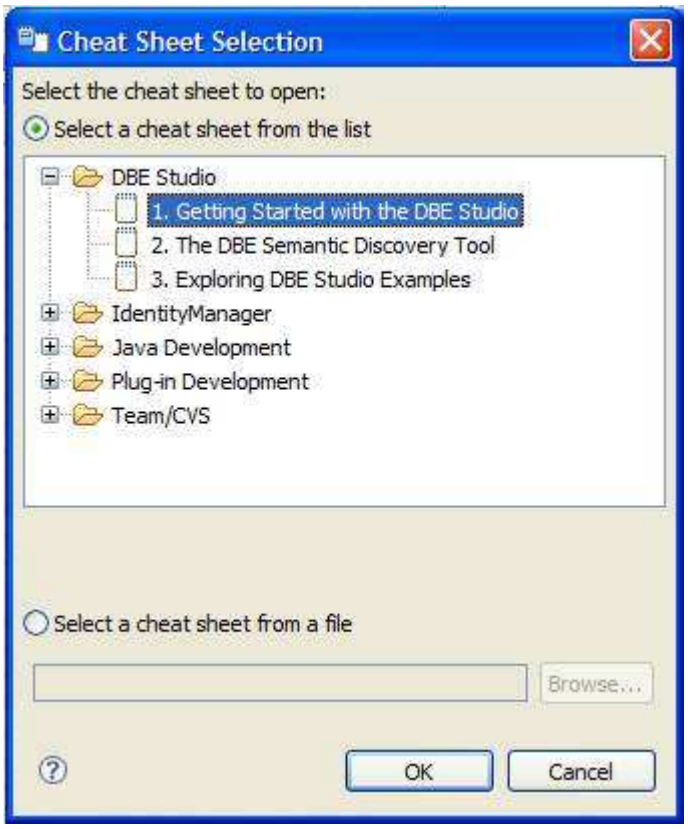


Figure 110 DBE Studio Documentation – Cheat Sheet Access

Figure 111 DBE Studio Documentation – Cheat Sheet

These cheat sheets are invaluable to a new user to the DBE Studio as they provide clear and well defined paths from beginning to end to accomplish a task. It is recommended that more cheat sheets be created for assisting the user in other areas in the Studio.

8 Preferences

In this chapter the perferences contained within the DBE Studio are analyzed and recommendations are made upon that analysis. The DBE Studio preferences are contained and integrated within the preferences of Eclipse. There are general preferences for the DBE Studio and then specific preferences for some plugins. The plugins that are customizable through the preferences are as shown in the following table (Table 13). Those plugins that are not listed in the table read their preferences from the general preferences page.

Plugin Name	Partner
BML Editor	Technical University of Crete
Connection Manager	Intel Ireland Ltd.
DBE Ontology Analysis	Technical University of Crete
Ontology Viewer	Technical University of Crete
Recommender	Technical University of Crete
Sdl2Java Compiler	Soluta
Semantic Discovery Tool	Technical University of Crete
SSL2SDL Compiler	Soluta
UI Generator	Intel Ireland Ltd.

Table 13 DBE Studio Plugin-specific Preference Pages

What are listed in the above table (Table 13) are the DBE Studio plugin-specific and general preference pages that are to be considered for evaluation in this document. These very same preferences can be accessed within the DBE Studio through the menu "Windows->Preferences…" and are illustrated below:

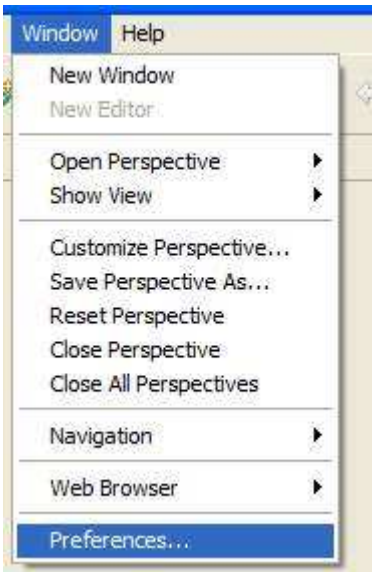


Figure 112 DBE Studio Preferences - Access

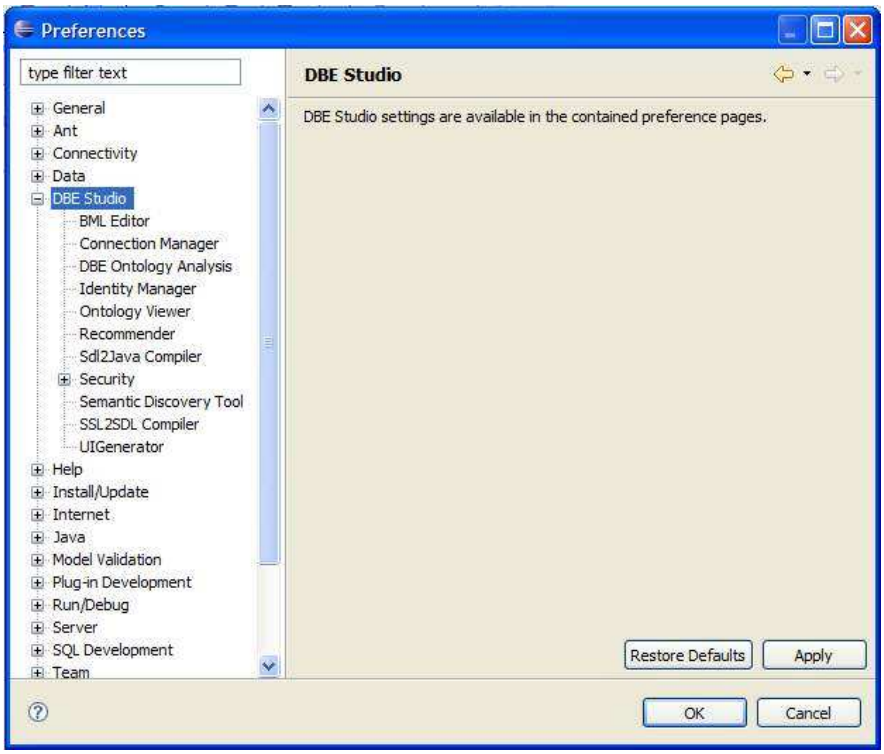


Figure 113 DBE Studio Preferences - General

Before considering each individual plugin preference, the general Eclipse recommendations are detailed which should always be considered when designing and implementing Eclipse views.

8.1 UI Recommendations

When designing preferences, it is prudent to consult the Eclipse HIG [1] for guidelines. These guidelines are found in [1] and expanded upon there and also form the basis for the evaluation of DBE preferences. Hence, it is useful to summarize them for reference:

Ref.	Description
PG1	Global options should be exposed within the preferences dialog.
PG2	Expose the preferences for a particular view, editor or window in the view itself, via a menu or tool item.

PG3	Start out with a single preference page. Then evolve to more if you need to.
PG4	If you create a preference group, use the root page for frequently used preferences, or those preferences which have wide spread effect. Specialize within the sub pages. The root preference page should not be blank.
PG5	Attempt to integrate plug-in preferences, wizards, and views into existing categories for a new plug-in first, before considering the creation of a new category.

Table 14 Recommended Preference Guidelines

These guidelines are used through out this section on the evaluation of DBE Studio preferences and the above table will serve as a reference point for this evaluation and also developers creating or refactoring DBE Studio plugins.

8.2 Connection Manager Preference Page

The DBE Studio connection manger preference page includes options that configure the connection manager through which all DBE Studio plugins connect to the ExE.

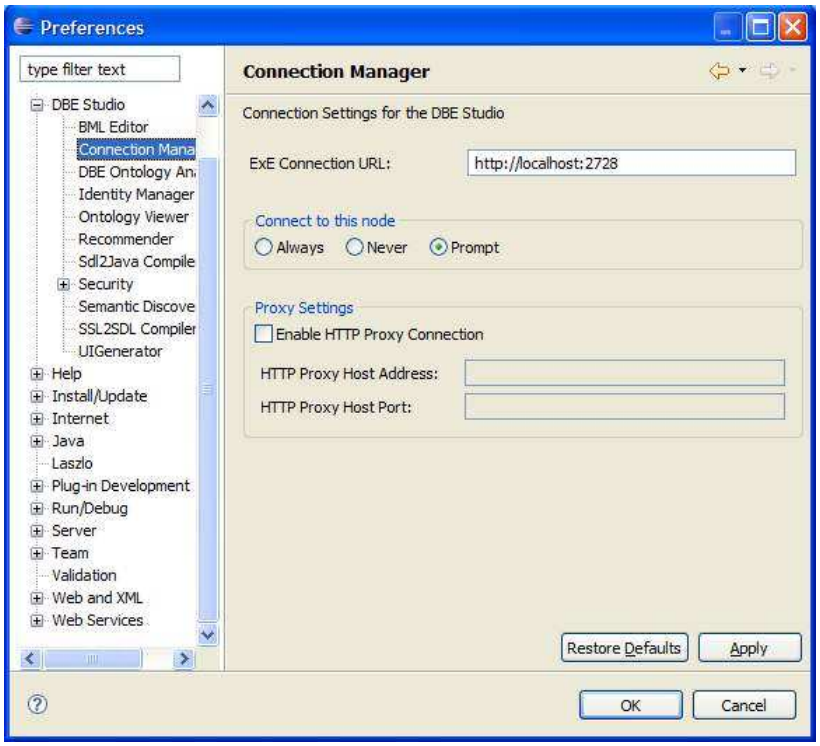


Figure 114 DBE Studio Preference Page – Connection Manager

There were no issues found with the connection manager's preference page.

8.3 BML Editor Preference Page

The DBE Studio BML editor preference page includes options that are specific to the BML editor. Below is a screenshot of how the DBE Studio BML editor preference page appears once it has been opened by the user.

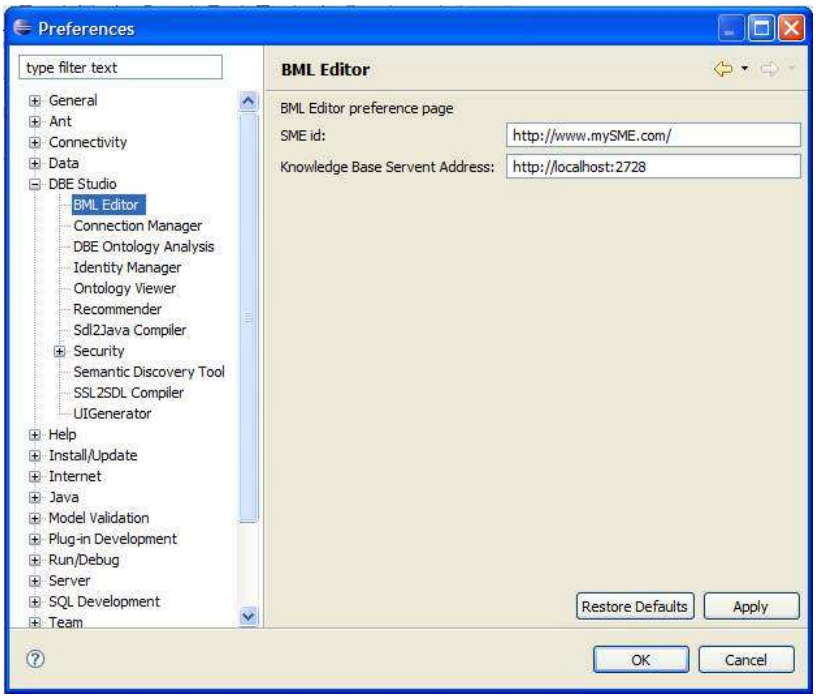


Figure 115 DBE Studio Preference Page – BML Editor

With the DBE Studio BML editor preference page visually introduced, an evaluation of it is now presented:

- The servent URL text input field should be removed and the servent URL preference be read from the connection manager instead.
- The "SME id" fields should be moved to general preferences as it is an option that is shared by both this preference page and the "DBE Ontology Analysis" preference page.

8.4 DBE Ontology Analysis Preference Page

The DBE Studio ontology analysis preference page includes options that are specific to ontologies. Below is a screenshot of how the DBE Studio ontology analysis page appears once it has been opened by the user.

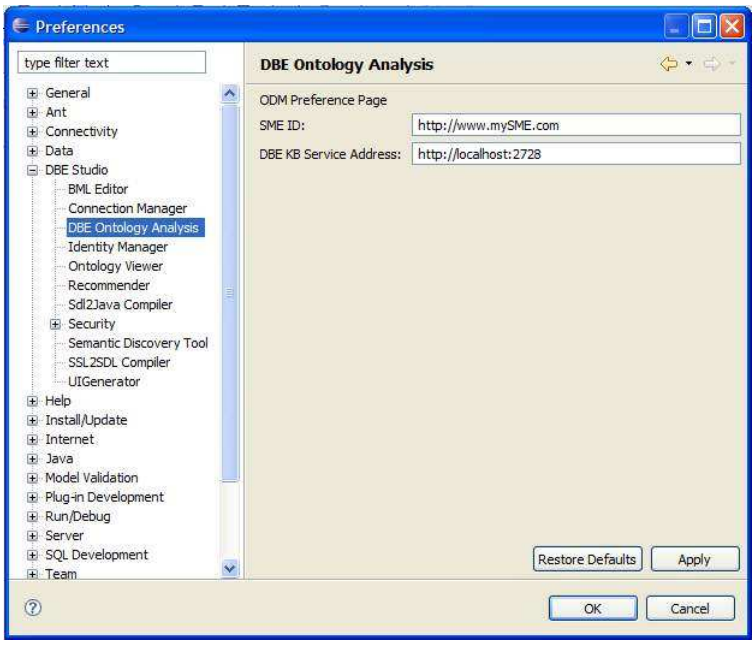


Figure 116 DBE Studio Preference Page – Ontology Analysis

With the DBE Studio ontology analysis preference page visually introduced, an evaluation of it is now presented:

- It is recommended that the "Ontology Viewer" preference page be merged with this page.
- The servent URL text input field should be removed and the servent URL preference be read from the connection manager instead.
- The "SME id" fields should be moved to general preferences as it is an option that is shared by both this preference page and the "DBE Ontology Analysis" preference page.
- The text describing the preference page should be renamed from "DBE Ontology Analysis" to "Ontology Analysis".

8.5 Ontology Viewer Preference Page

The DBE Studio ontology viewer preference page includes options that are specific to the ontology viewer (see Figure 63). Below is a screenshot of how the DBE Studio ontology viewer page appears once it has been opened by the user.

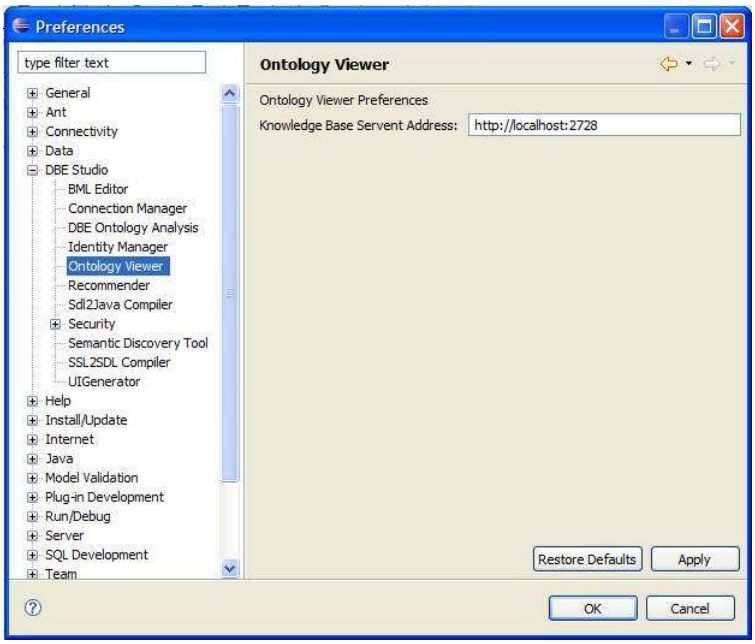


Figure 117 DBE Studio Preferences – Ontology Viewer

With the DBE Studio ontology viewer preference page visually introduced, an evaluation of it is now presented:

- As there is only one input field and already a preference page associated with ontologies it would be recommended that this page be merged with the "DBE Ontology Analysis" page in accordance with PG5.

8.5.1 Connection Management Issues

After entering a new value in the "Knowledge Base Servent Address" and applying the setting or clicking the "Ok" button to dismiss the preferences, a dialog box is popped up trying to initiate a connection to the new address. This is unexpected behaviour as discussed in section 2.3.

8.6 Recommender Preference Page

The DBE Studio recommender preference page includes options that are specific to the recommender service running on a servent (normally specified through the general preference page, Figure 113). Below is a screenshot of how the DBE Studio recommender page appears once it has been opened by the user.

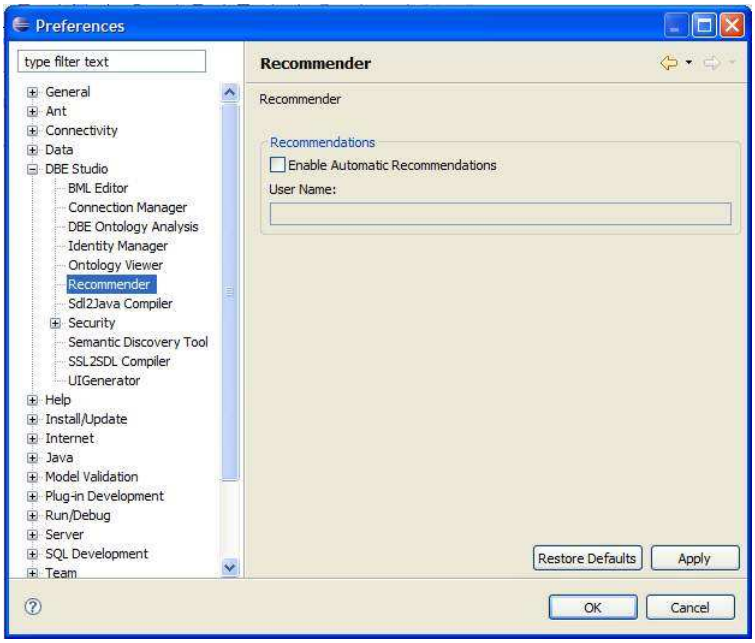


Figure 118 DBE Studio Preferences - Recommender

With the DBE Studio recommender preference page visually introduced, an evaluation of it is now presented:

- The recommender is a feature that is well hidden from the user. In order to make it more visible to the user the suggestion of adding a check box reading "Enable Recommendations" could be implemented. This check box would be added to the keyword search view and as an option to a query within the query editor, both components of the semantic discovery perspective.
- The preference page satisfies all five perspective guidelines found in Table 14.

8.7 Sdl2Java Compiler Preference Page

The DBE Studio Sdl2Java preference page includes options that are specific to the generation of Java skeletons from SDL. Below is a screenshot of how the DBE Studio Sdl2java page appears once it has been opened by the user.

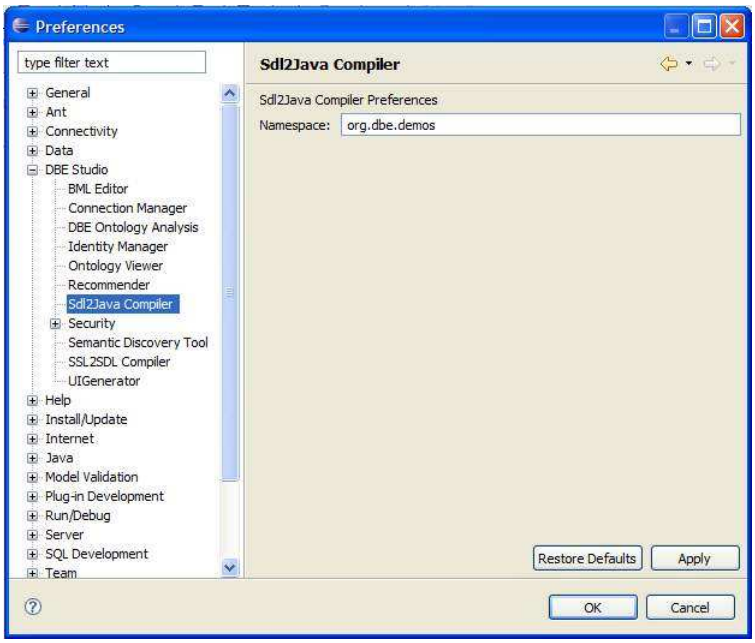


Figure 119 DBE Studio Preferences – Sdl2Java

With the DBE Studio Sdl2Java compiler preference page visually introduced, an evaluation of it is now presented:

- The namespace option should be merged into the SDL2Java wizard that generates the Java Skeletons as this is a parameter of the compiler that would be conceivably changed frequently by a user of the compiler.
- The capitalization of "Sdl2Java" should be changed to "SDL2Java".

8.8 Semantic Discovery Tool Preference Page

The DBE Studio Semantic Discovery Tool (SDT) preference page includes options that are specific to the generation of Java skeletons from SDL. Below is a screenshot of how the DBE Studio SDT page appears once it has been opened by the user.

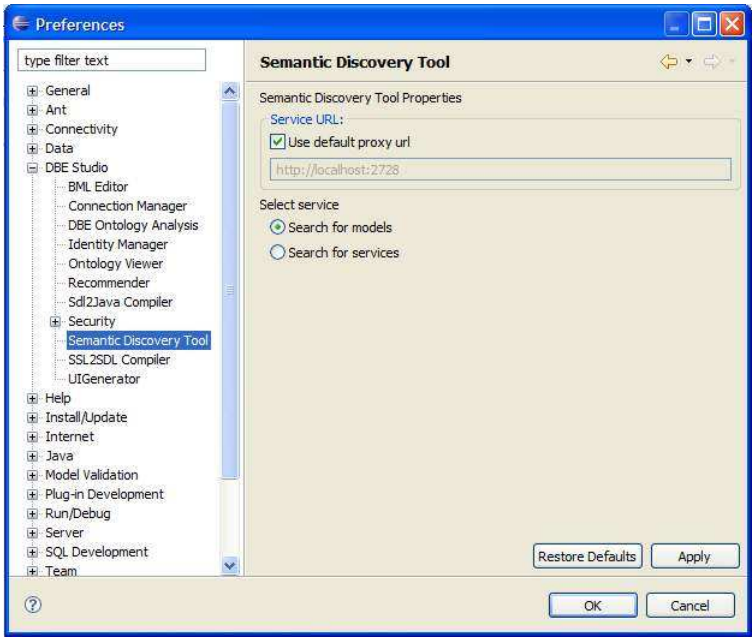


Figure 120 DBE Studio Preferences – Semantic Discovery Tool

With the DBE Studio SDT preference page visually introduced, an evaluation of it is now presented:

- The preference page satisfies all five perspective guidelines found in Table 14.

- The servent URL text input field should be removed and the servent URL preference be read from the connection manager instead.

8.9 SSL2SDL Compiler Preference Page

The DBE Studio SSL2SDL compiler preference page includes options that are currently specific to the transformation of SSL to SDL. Below in Figure 121 is how the SSL2SDL compiler preference page appears.

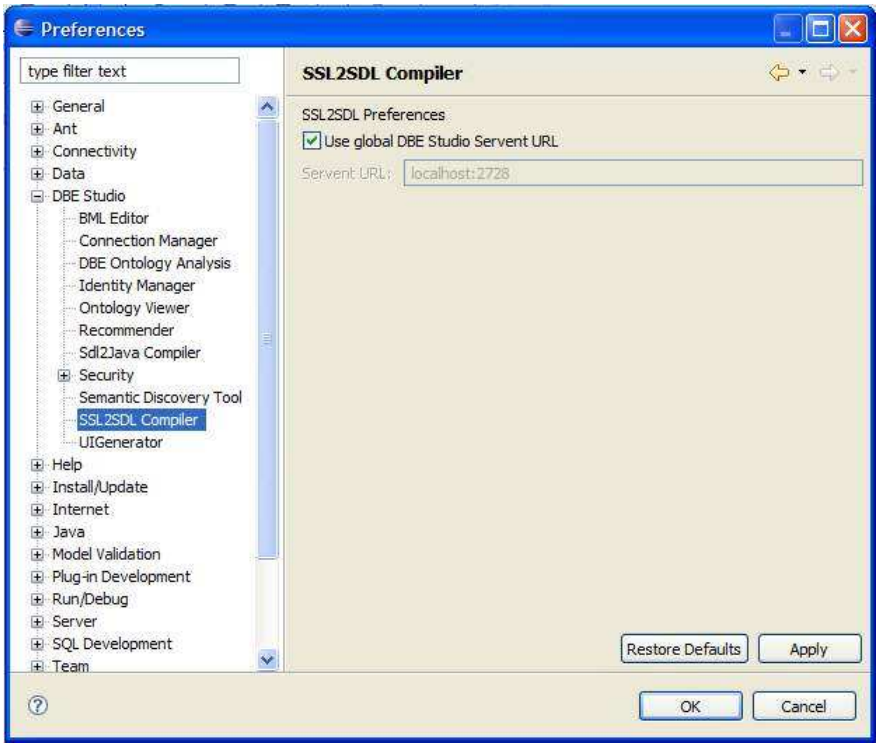


Figure 121 DBE Studio Preferences – SSL2SDL Compiler

The preference page satisfies all five perspective guidelines found in Table 14. However, the servent URL text input field should be removed and the servent URL preference be read from the connection manager instead.

8.10 UI Generator Preference Page

The DBE Studio user interface generator preference page includes options that are specific to the generation of user interfaces within the DBE Studio. Below is a screenshot of how the DBE Studio user interface generator preference page appears once it has been opened by the user.

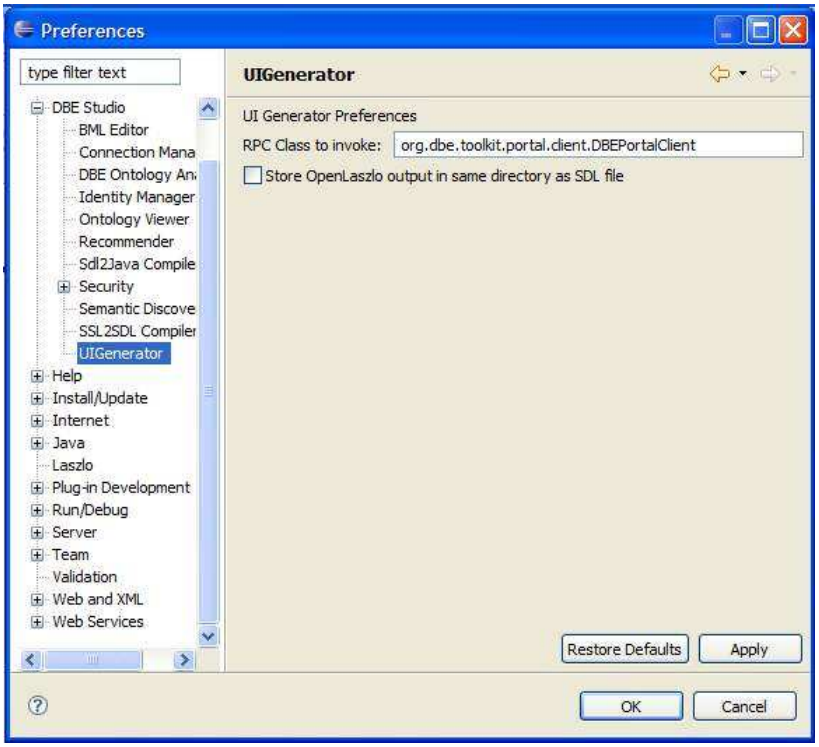


Figure 122 DBE Studio Preferences – UI Generator

With the DBE Studio service exporter preference page visually introduced, an evaluation of it is now presented:

- The name "UIGenerator" should be changed to "UI Generator" (note the space).
- The preference page satisfies all five perspective guidelines found in Table 14

9 UI Enhancements to Date

In this section, some of the changes that have been made since the initial draft [2] of this evaluation are listed. As this is on-going work, progressive updates and improvements are constantly being made by partners to the DBE Studio. Changes listed here are those from the initial draft document. The changes will be listed in the same order as the DBE Studio components that are listed in this document.

9.1 General DBE Studio Wide Suggestions

One suggestion in the draft document was that a connection manager (section 2.3) for the DBE Studio be implemented. The motivation behind this was to manage all connections and related settings made from the DBE Studio to the Servent. By managing all connections it not only allows the reuse of networking resources but also to address usability issues related to connection management within DBE Studio plugins. Perhaps the biggest issue related to connection management found in the DBE Studio was blocking servent calls where DBE Studio would present a dialog informing the user that it was waiting for the Servent to supply data. This had the effect of rendering a multi-task environment such that user interfaces are, into a mono-task environment. To remedy this, the connection manager has implemented asynchronous connection management using a call back architecture. The connection manger is also the authorative component that provides Servent URL information. This has the effect of centralizing Servent URL preferences to one preference page rather than multiple preference pages. The connection manager was also able to remedy the issue of some plugins not supporting a HTTP proxy. Another notable feature of the connection manger is that when a connection is performed the user is notified of this fact and asked if the connection is allowed. This notification can be shown every time a connection is made or only once by selecting the option on the dialog to always allow connections to the Servent. The integration of the connection manger would not have been possible without the help of all partners involved in the DBE Studio.

9.2 Perspectives

In the draft evaluation it was noted that the Ontology Perspective (section 3.2) did not open with the correct ratio of view to editor. This has been corrected by the Technical University of Crete and now makes opening the perspective a more familiar experience for the user. Below is how the current perspective (Figure 124) opens compared to the previous version (Figure 123).

DBE Studio 0.2.x

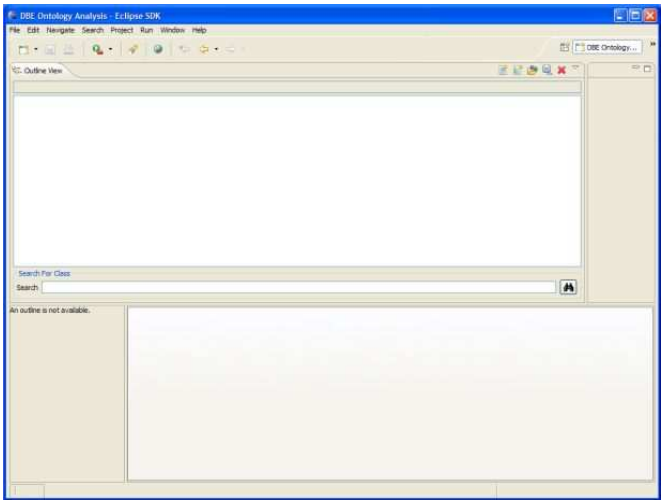


Figure 123 Ontology Analysis Perspective, 0.2.x version

DBE Studio 0.3.0

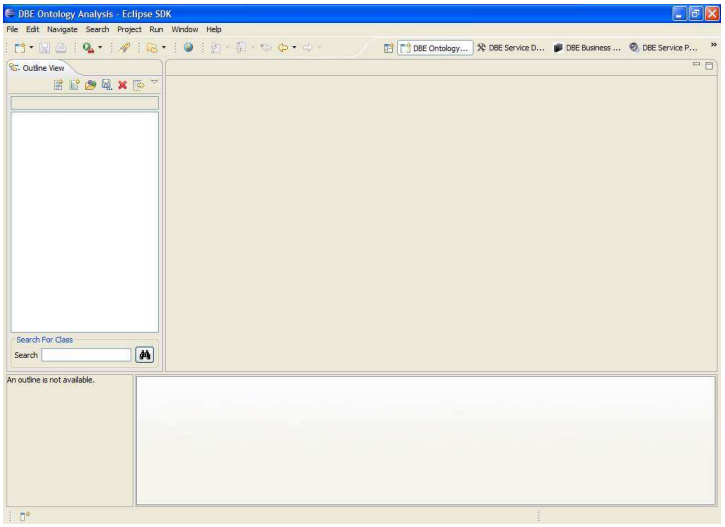


Figure 124 Ontology Analysis Perspective, 0.3.0 version

9.3 Editors

In the category of editors, the BML Data editor (section 4.4) had many non-standard aspects at the time of the initial evaluation. However, along with the recommendations made in the initial evaluation, the University of Central England was able to create a much more useable editor and by adopting the Eclipse HIG a more familiar editor, with respect to Eclipse UIs, was created. Shown below is the old, Figure 125, and new version, Figure 126.

DBE Studio 0.2.x

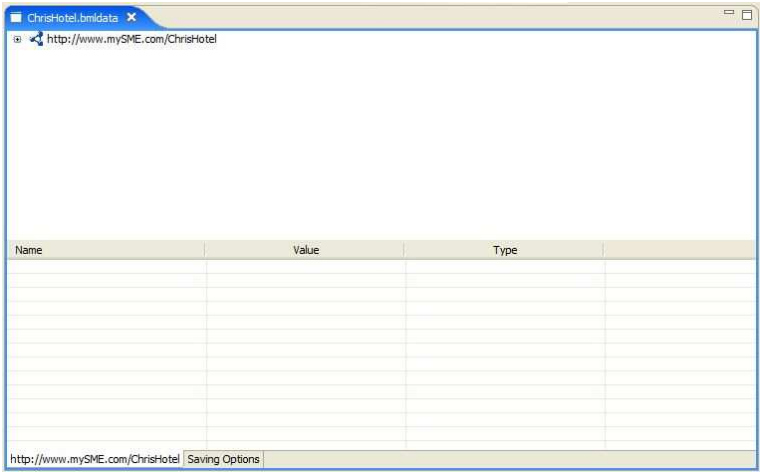


Figure 125 BML Data Editor, 0.2.x version

DBE Studio 0.3.0

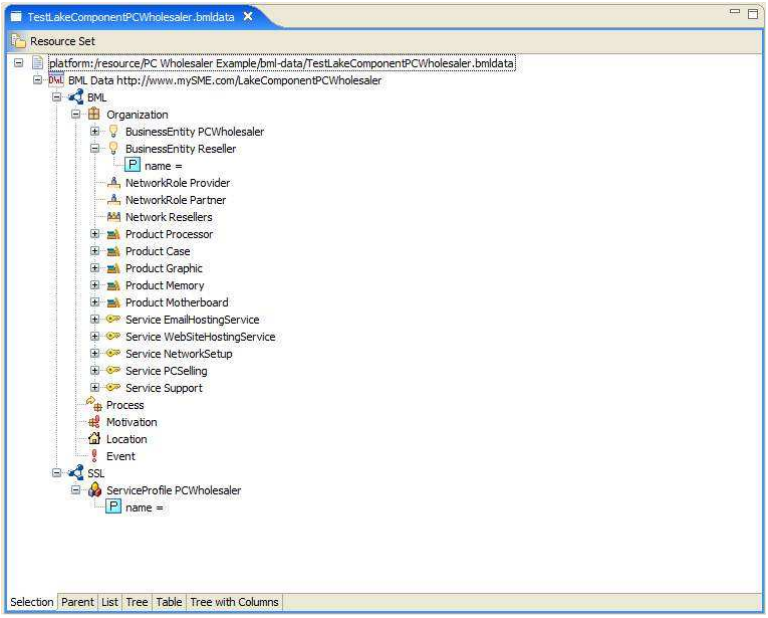


Figure 126 BML Data Editor, 0.3.0 version

Although not formally reviewed in the draft, the Service Manifest editor (section 4.9) has been improved by Soluta from a number of informal comments that were included in the draft evaluation. Those comments suggested that the input supplied to the editor be validated and that the editor be streamlined so as not to use as many editor tabs. Also suggested was to use the standard Eclipse mechanisms to save a service manifest (using Control + S) and that dates inserted in the service manifest be done so automatically where possible. Below is a visual comparison of the new (Figure 127) and old (Figure 128) editor.

DBE Studio 0.2.x version

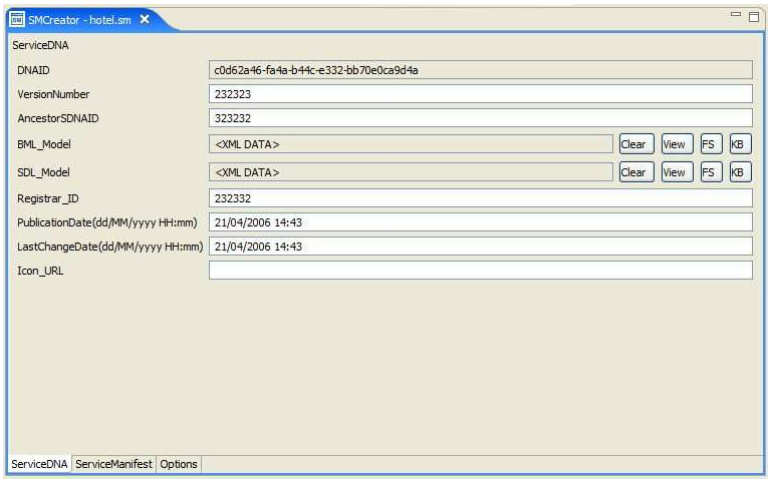


Figure 127 Service Manifest Editor, 0.2.x version

DBE Studio 0.3.0 version

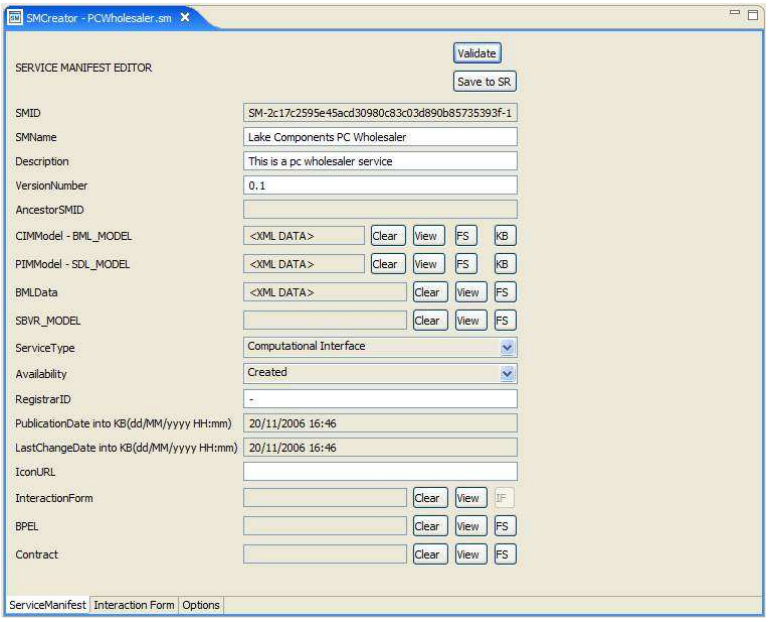


Figure 128 Service Manifest Editor, 0.3.0 version

9.4 Views

There were a number of suggestions made in relation to views in the initial draft evaluation. Of those that were evaluated the following views have implemented the noted changes:

- Ontology Outline View (section) - a full menu is now accessible from the view toolbar.
- Query Navigator View - a full menu is now accessible from the view toolbar.
- Query Template View – a button to remove properties has been added. An additional button to edit the current template has also been added.
- Keyword Search View – a better descriptive and terser text has been added to the wizard description header.

9.5 Wizards

When first creating a resource in the DBE Studio many users will find themselves in the "New" resource dialog (see 6.2). It's here where a user selects the appropriate wizard to create the resource.

From the draft evaluation it was noted that a number of wizards listed in the "New" resource dialog were missing descriptions. The query (section 6.2.4) and query template wizards (section 6.2.5) have now been updated to supply this information.

The SDL Import wizard (section 6.4.2) has been reduced to only 2 pages following the suggestion that the 3rd page that contained network configuration information be removed. This was made possible with the introduction of the connection manager plugin.

Inline with the new BML Data editor (section 9.3) there has been a reworking of the BML Data wizard (section 6.2.1). This provides a more streamlined and familiar approach to creating a BML Data file.

The query wizard (section 6.2.4) has been updated so that its main description provides an accurate reflection of its functionality. Also tabbing and initial cursor focus issues noted in the draft evaluation have been addressed. Changes to the query template wizard (section 6.2.5) have been made so that the description of the wizard is accurately and tersely displayed to the user on both pages of the wizard.

The Test Case Generator wizard (section 6.3.4) has been simplified as a result of the suggestions in the draft evaluation. It is no longer necessary to set an eclipse environment variable (SERVENT_HOME) with the assumed value of "C:\servent". The related test case scenario wizard (section 6.2.9) was also updated. Now the finish button is disabled until all required fields are populated and on the second page the "Remove" button now functions as expected.

There was a major overhaul of the DBE Studio Service Exporter plugin. The changes in this wizard can be shown visually in the following tables (Table 15 and Table 16).

In this section the refactored DBE Studio Service Exporter UIs are presented after the application of the recommendations specified in section 6.4.1. This visual comparison provides an immediate and clear justification on the value of the recommendations contained in this document.

DBE Studio version 0.2.x

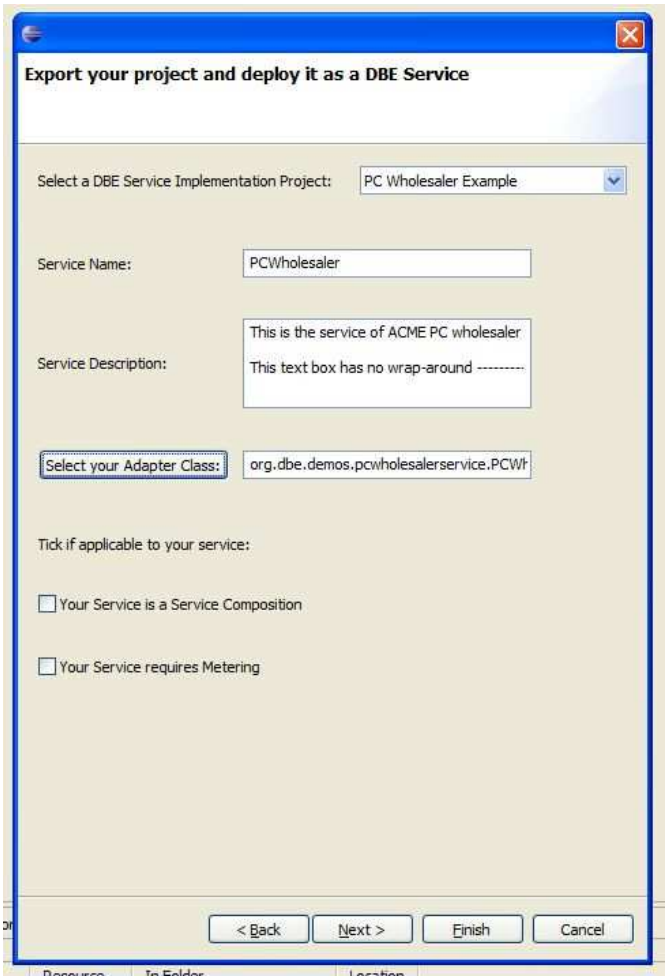


Figure 129 DBE Studio Wizards – Exporter Before Recommendations

DBE Studio version 0.3.0

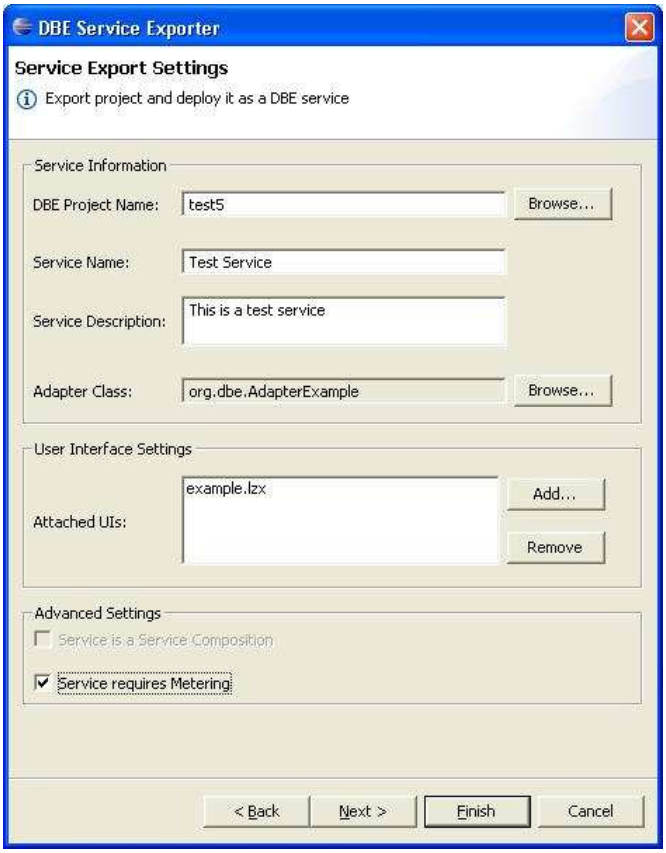


Figure 130 DBE Studio Wizards – After Before Recommendations

Table 15 DBE Studio Wizards – Application of UI Recommendations to the 1st Exporter Page

DBE Studio version 0.2.x

Figure 131 DBE Studio Wizards – Exporter Before Recommendations

DBE Studio version 0.3.0

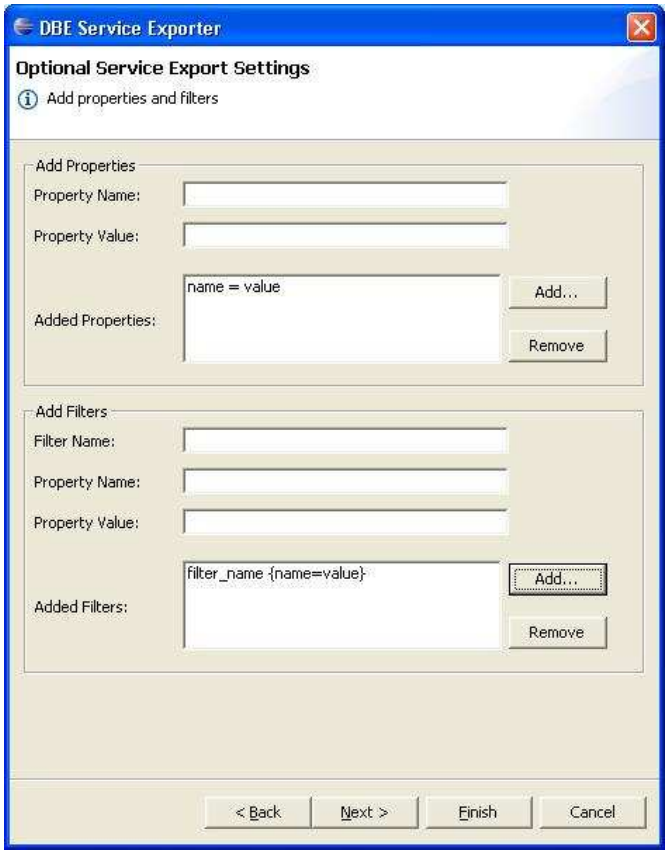


Figure 132 DBE Studio Wizards – Exporter After Recommendations

Table 16 DBE Studio Wizards – Application of UI Recommendations to the 2nd Exporter Page

9.6 Help

In the initial draft it was noted that there was missing help for a number of components. These observations have been noted and as a result, the SDL Editor (Soluta) and the DBE Project Wizard

(Intel Ireland Ltd.) help documentation have been added in order to aid users, new to the DBE Studio.

9.7 Preferences

As tools change, so too do their preferences and this is evident by comparing the preference pages from the initial draft evaluation to this current document. Although preferences are in a state of flux, it has been always a goal to reduce the number of configuration parameters in the DBE Studio. As discussed in section 9.1, with the introduction of the connection manager the number of places where connection information has to be configured will be reduced to one main preference page in the DBE Studio preferences.

10 Conclusion

This document has evaluated the DBE Studio's UI in respect to human interface guidelines issued by the Eclipse foundation [5]. In the evaluation, a number of general DBE Studio-wide recommendations were made, followed up by comprehensive suggestions for each individual DBE Studio user interface element. To share these recommendations with the developers and the wider technical community, the recommendations are being published on the DBE Studio sourceforge web site under the "Feature Request" section. It should be noted that although implementing some of these recommendations may seem straightforward, some will be time-consuming. Given the very limited time and resources remaining among partners it is not practical to expect that they will all be implemented during the timeline of the project. However, it is reassuring to note that even at the time of writing, numerous of these recommendations have already been taken on board, some even by partners that have no funded resources remaining in the project.

It was explained in the introduction of this document that this evaluation of the DBE Studio follows a bottom-up approach. The reason for this was due to the structure, practicalities and time-lines of the implementation and integration of the various DBE Studio components. In particular it was desirable to avoid the chance that such an exercise necessitating significant reengineering or even merging of components in the DBE Studio. As this evaluation is indeed a bottom-up approach it is prudent to mention possible next steps that could be carried out to drive further recommendations. With all relevant suggestions contained in this document integrated into the DBE Studio, the DBE Studio will be an even more stable platform. With this in mind, it would be seen as a valuable exercise to perform usability testing on the DBE Studio. In this type of testing the DBE Studio would be tested upon a user-base that is representative of the users using the DBE Studio. These users would be asked to navigate the DBE Studio as they would normally but, for example, with the request that they speak through their actions using a "talk-aloud protocol" [11]. Another useful exercise to carry out would be to analyze the workflows (the basic workflows of the DBE Studio are shown in Figure 3 and Figure 4) contained within the DBE Studio and try and see where optimizations could be applied. Such optimizations may entail automating, merging or perhaps replacing certain components.

This document has illustrated some of the significant optimisations that have already been made as a result of the adoption of draft recommendations. Given the demonstrated openness and dedication of the DBE Studio developers, DBE Studio users can look forward to further user-interaction enhancements in future DBE Studio releases where time and resources permit.

References

[1] DBE Studio Project Website, http://sourceforge.net/projects/dbestudio

[2] Initial DBE Studio UI Evaluation Draft, 05/07/2006, http://sourceforge.net/mailarchive/forum.php?thread_id=19159320&forum_id=47373

[3] DBE Studio Release Version 0.3.0, http://sourceforge.net/mailarchive/message.php?msg_id=37531899

[4] DBE Studio Feature Request System, http://sourceforge.net/tracker/?group_id=143906&atid=757079

[5] Eclipse User Interface Guidelines Version 2.1, http://www.Eclipse.org/articles/Article-UI-Guidelines/Index.html

[6] "Official Guidelines for User Interface Developers and Designers", Windows Human Interface Guidelines, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwue/html/welcome.asp

[7] Apple Human Interface Guidelines, http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/index.html

[8] Gnome Human Interface Guidelines 2.0, http://developer.gnome.org/projects/gup/hig/2.0/

[9] Adobe Systems, Flash, http://www.adobe.com/flash

[10] Adobe Systems, Director, http://www.adobe.com/director

[11] C. Lewis and J. Rieman, "Task-Centered User Interface Design: A Practical Introduction", http://www.hcibib.org/tcuid/

[12] Publications of Jakob Nielsen (www.useit.com), http://www.interaction-design.org/references/authors/jakob_nielsen.html

[13] The Eclipse Framework, http://www.eclipse.org

[14] The Digital Business Ecosystem, http://www.digital-ecosystem.org

[15] The Execution Environment, http://sourceforge.net/projects/swallow

[16] The Evolutionary Environment, http://evenet.sourceforge.net

[17] ISUFI, DBE Deliverable: D15.1 - BML First Release

[18] Log4J Apache Project, http://logging.apache.org/log4j

[19] Ganymede Log4J plugin for Eclipse, http://personalwebs.oakland.edu/~ba2opfer/ganymede/

[20] Log4J Integration plugin for Eclipse, http://www.eclipse-plugins.info/eclipse/plugin_details.jsp?id=814

[21] Logging within the Eclipse framework, http://wiki.eclipse.org/index.php/FAQ_How_do_I_use_the_platform_logging_facility%3F

[22] Extending Eclipse's Welcome Page, http://help.eclipse.org/help32/index.jsp?topic=/org.eclipse.platform.doc.isv/guide/ua_intro_universal_extending.htm

[23] DBE Studio Examples, version 0.3.0, http://sourceforge.net/project/showfiles.php?group_id=143906&package_id=167565

[24] Business Process Execution Language Specifications, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

[25] Eclipse BPEL Project, http://www.eclipse.org/bpel/

[26] OpenLaszlo Project, http://www.openlaszlo.org

[27] SDL to Kb Export Plugin status, http://sourceforge.net/tracker/index.php?func=detail&aid=1605749&group_id=143906&atid=757076

11 Appendix

11.1 Installation

As the DBE Studio is built on top of the Eclipse platform it can take advantage of the Eclipse install and update mechanism with which existing Eclipse users would be already familiar. For both first time users and existing users of the DBE Studio, a simple procedure enables them to manually (or automatically) install or update the latest version of the DBE Studio within their Eclipse installation. Ensuring that the required feature dependencies were installed, a first time user can search for a DBE Studio feature using the Eclipse 'Software Updates, Find and Install' mechanism. After entering the update site URL for the DBE Studio[2], they can select the latest feature and install it within their existing Eclipse installation.

For existing users of the DBE Studio, it is possible to manually search for new feature releases or set up automatic updates which will automatically search for new feature releases and install the update if desired by the user.

Perhaps the only problem with this approach is that users need to have the additional prerequisite Eclipse plugins (EMF and GEF). Unfortunately for users, they need to install these manually (will change with Eclipse 3.2). A proposed solution to this would be to create a full standalone version of the DBE Studio packaging it will all the Eclipse runtimes and plugins. This has been implemented and "all in one" distributions of the DBE Studio are now available for download from the DBE Studio sourceforge web site

[2] http://dbestudio.sourceforge.net/install