

Architecture Specification

for
NCI Protégé Extension
Version 1.0, Release 1.0

Last Updated: August 17, 2006

Owner: <name>
Telephone: <number>
Email: <address>

National Cancer Institute Center for Bioinformatics
6116 Executive Blvd.
Rockville, MD 20852



Contents

List of Tables	ii
List of Figures	iii
1. Overview.....	1
1.1 Purpose of this Specification	1
1.2 Background: Core Protégé and OWL Plug-in	2
1.3 About the NCIEdit Tab Plug-In.....	3
2. UI Component Architecture	5
2.1 About the Edit tab.....	6
2.2 About the Split Tab.....	21
2.3 About the Pre-Merge Tab	23
2.4 About the Merge Tab.....	25
2.5 About the Pre-Retire Tab.....	26
2.6 About the Retire Tab	28
2.7 About the Report Writer Tab.....	29
2.8 About the Batch Loader Tab.....	31
2.9 About the Batch Editor Tab.....	34
2.10 About the Parontology Tree Tab	37
2.11 About the Copy Tab	40
2.12 About the NCIEdit Tab Plug-In Utility Classes	42
3. Performance Considerations	43
4. Third-party Tool Dependencies and Requirements	44
5. Document Revision History	45
6. Document Approval.....	46
6.1 Approvers List	46
6.2 Reviewers List	46

List of Tables

Table 1. UI components of the Edit tab	6
Table 2. UI components of the Basic Data sub-tab.....	9
Table 3. UI Components of the Relations sub-tab	12
Table 4. UI Components of the Properties sub-tab	16
Table 5. UI Components of the Split tab	21
Table 6. UI Components of the Pre-Merge tab.....	23
Table 7. UI Components of the Merge tab.....	25
Table 8. UI Components of the Pre-Retire tab.....	26
Table 9. UI Components of the Retire tab	28
Table 10. UI Components of the Report Writer tab.....	29
Table 11. UI Components of the Batch Loader tab	31
Table 12. UI Components of the Batch Editor tab.....	34
Table 13. Data elements for a batch file	35
Table 14. UI components of the Paratomy Tree tab	37
Table 15. UI components of the Copy tab	40
Table 16. Utility classes	42
Table 17. NCIEditTab architecture/standards usage.....	42
Table 18. Third-party tool requirements/dependencies	44
Table 19. <Document Title> Document Change Log.....	45
Table 20. Approvers for this document	46
Table 21. Reviewers for this document	46

List of Figures

Figure 1. OWL plug-in main user interface	2
Figure 2. Multi-user editing environment of NCI Protégé extension	4
Figure 3. NCI Protégé Extension configuration.....	4
Figure 4. NCIEditTab user interface design	5
Figure 5. Edit tab	6
Figure 6. Message confirming deletion of data.....	8
Figure 7. Create New Annotation Property dialog box for FULL_SYN data	10
Figure 8. Create New Annotation Property dialog box for definitions.....	11
Figure 9. Relations sub-tab	12
Figure 10. Create a Restriction dialog box	13
Figure 11. Select a class dialog box.....	14
Figure 12. Add an Object-Valued Property dialog box	15
Figure 13. Properties sub-tab	16
Figure 14. Select Property dialog box.....	17
Figure 15. Add Annotation dialog box	17
Figure 16. Select Property dialog box.....	19
Figure 17. Create <Property Name> Annotation Property dialog box	19
Figure 18. Review window	20
Figure 19. Split tab.....	21
Figure 20. Tree representation of a class	22
Figure 21. Enter Class Identifiers dialog box	22
Figure 22. Pre-Merge tab	23
Figure 23. Enter Notes dialog box	24
Figure 24. Merge tab.....	25
Figure 25. Pre-Retire tab.....	26
Figure 26. Retire tab	28
Figure 27. Report Writer tab.....	29
Figure 28. Sample report: Lymph_Node_Sinus	30
Figure 29. Batch Loader tab.....	31
Figure 30. Batch Loader input file format	32
Figure 31. Batch Loader dialog box	32
Figure 32. Sample batch loader log file	33
Figure 33. Batch Editor tab.....	34
Figure 34. Batch Input file	35
Figure 35. Sample Batch Editor log file	36
Figure 36. Partonomy Tree tab	37
Figure 37. Select Transitive Properties dialog box	38
Figure 38. Sample partonomy tree.....	39
Figure 39. Copy tab	40

1. Overview

1.1 Purpose of this Specification

This design document details the design of the Protégé/OWL NCIEdit Plug-In (abbreviated as NCIEdit Tab Plug-In), Release 1.0. It describes the major user interface components that make up NCIEdit Tab, interfaces to other products, third-party tool requirements or dependencies, and document review and approval requirements.

The intended audience for this document includes, but is not limited to, the following teams: development, management, product support technical personnel, technical communication, and technical consultants.

1.2 Background: Core Protégé and OWL Plug-in

The base system of Protégé is an open-source development environment for ontologies and knowledge-based systems. The OWL Plug-in is an extension of Protégé with support for the Web Ontology Language (OWL).

The base systems of Protégé and the OWL Plug-in are being developed at Stanford Medical Informatics. Protégé and the OWL plug-in support ontology editing in a multi-user client/server environment. A group of users at geographically dispersed locations can edit the same ontology data concurrently.

Figure 1 shows the main user interface of the OWL Plug-in.

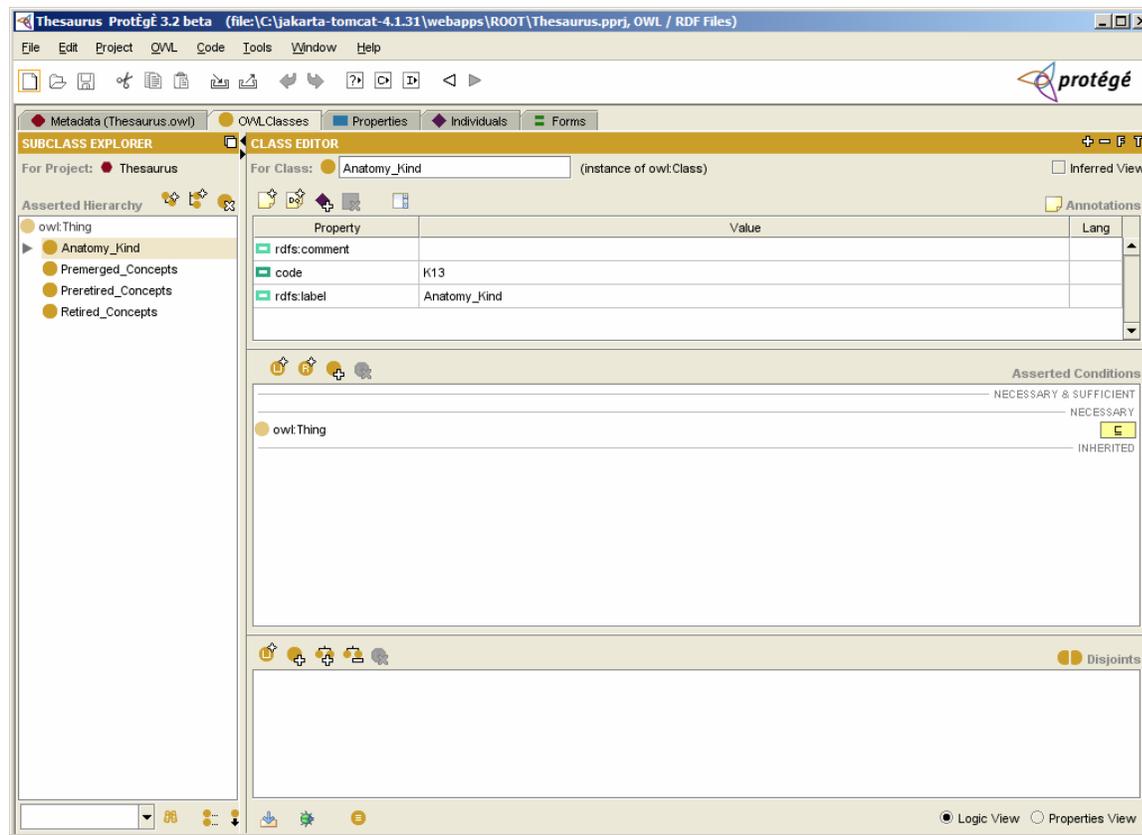


Figure 1. OWL plug-in main user interface

1.3 About the NCIEdit Tab Plug-In

The NCIEdit Tab Plug-In is a Protégé tab plug-in being developed for an NCICB-specific editing environment. This plug-in provides additional editing capabilities that are not available in the Protégé OWL Plug-in.

The NCIEdit Tab Plug-In enables NCICB users to do the following:

- Automatically assign a unique code to each newly created OWL named class
Note: For simplicity, this document will refer to an OWL named class as simply a *class*.
- Use a customized dialog box to edit synonyms with source data (FULL_SYN properties)
- Use a customized dialog box to edit definitions with qualifiers (DEFINITION properties)
- Parse XML-formatted complex annotation property values (such as DEFINITION, GO_ANNOTATION, and LONG_DEFINITION) and properly display the corresponding qualifiers
- Set edit restrictions (OWL anonymous class) and classes through different UI components
- Edit object-valued annotation properties (associations between classes) through a separate UI component
Note: In the OWL Plug-in, all annotation properties are shown in the same table.
- Edit simple and complex annotation properties (such as GO_ANNOTATION and LONG_DEFINITION) through different UI components
- Split an existing class into two classes
- Flag two existing classes for a merge (the pre-merge action)
- Merge one class with another
- Flag existing classes for retirement (the pre-retire action)
- Retire a class
- Generate reports
- Load a batch of classes to the knowledge base
- Edit a batch of classes
- Generate a paronomy tree
- Clone classes
- Edit two classes at the same time using cut and paste functions
- Alert a user of any change made by other users to ensure data integrity
- Enforce NCI-specific editing business rules.

Figure 2 and Figure 3 on page 4 show a top-level architecture of the Protégé ontology editing environment in which the NCI Protégé Extension will be used.

Figure 2 shows a Protégé server running at a centralized location. The server is connected to a MySQL database that enables all clients to access the common knowledge base.

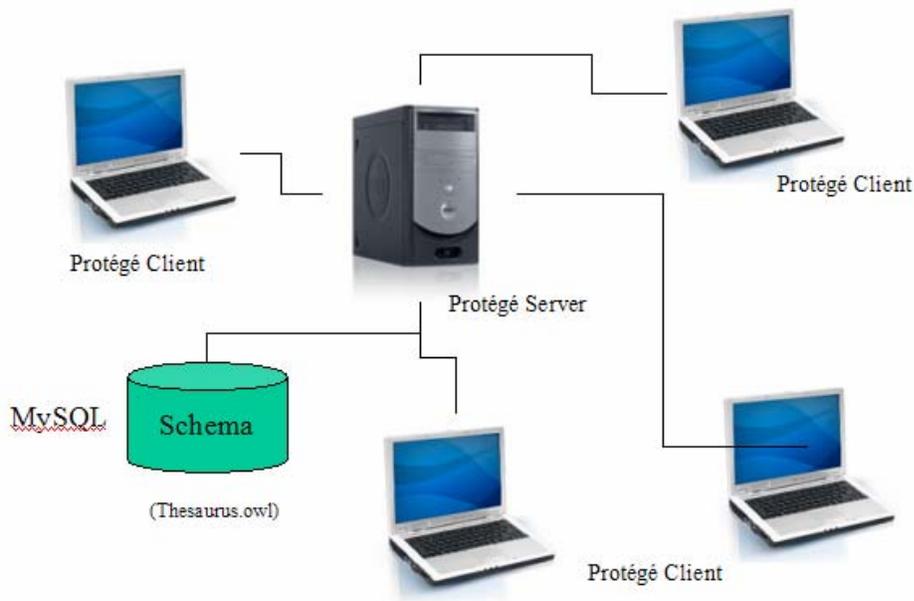


Figure 2. Multi-user editing environment of NCI Protégé extension

Figure 3 shows the location of the configuration files used by the NCI Protégé Extension. The code generator server assigns a unique code to classes.

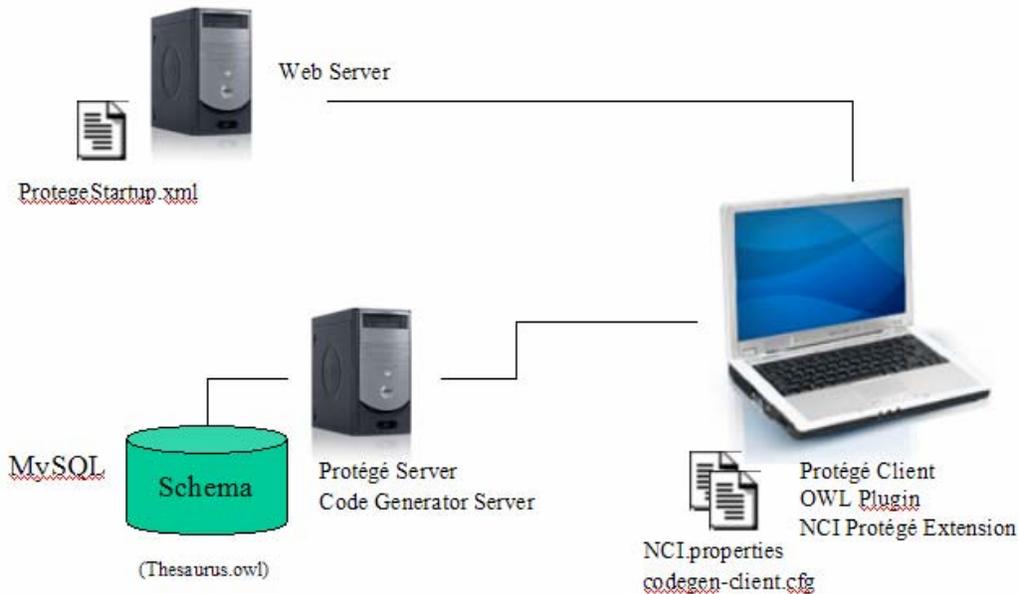


Figure 3. NCI Protégé Extension configuration

2. UI Component Architecture

The NCIEdit Tab Plug-In is a Protégé tab widget plug-in. All Protégé tab widgets extend AbstractTabWidget and implement the initialize() method of AbstractTabWidget.

The NCIEdit Tab Plug-In starts with the NCIEditTab class, which extends AbstractTabWidget. The initialize() method of the NCIEditTab class is used to construct a user interface with the look and feel shown in Figure 4.

Note: The initialize method also instantiates several utility classes. For more details, see *About the NCIEdit Tab Plug-In Utility Classes* on page 42.

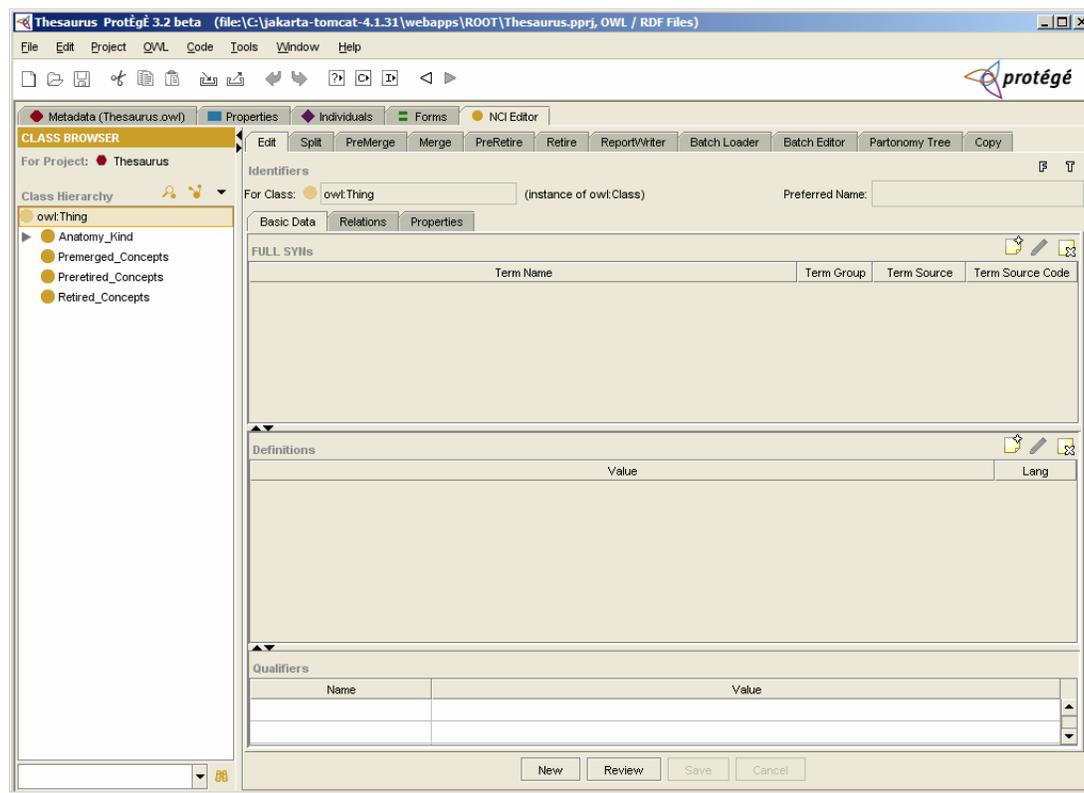


Figure 4. NCIEditTab user interface design

The vertical pane in the left side of the main window is a container for a class browser. This component enables users to browse the taxonomy of an underlying ontology and provides a means to search the ontology for classes or concepts that match user-specified search criteria.

On the right is a tabbed pane with the following tabs: Edit, Split, Pre-merge, Merge, Pre-retire, Retire, Report Writer, Batch Loader, Batch Editor, Partonomy Tree, and Copy. Each tab has its own interface components, which are covered in this specification.

2.1 About the Edit tab

The Edit tab is used to maintain annotation and relation data for classes, including the following:

- Special annotation properties that uniquely identify a class, such as the name, preferred name, and code
- Basic data about a class, including FULL_SYN annotation properties, DEFINITION annotation properties, and DEFINITION qualifiers
- Relation data, including restrictions, named superclasses, and associations
- Other annotation property data, including simple properties such as an annotation property without qualifiers, and complex properties such as GO_ANNOTATION and LONG DEFINITION, with one or many qualifiers.

Figure 5 illustrates the UI components of the Edit tab.

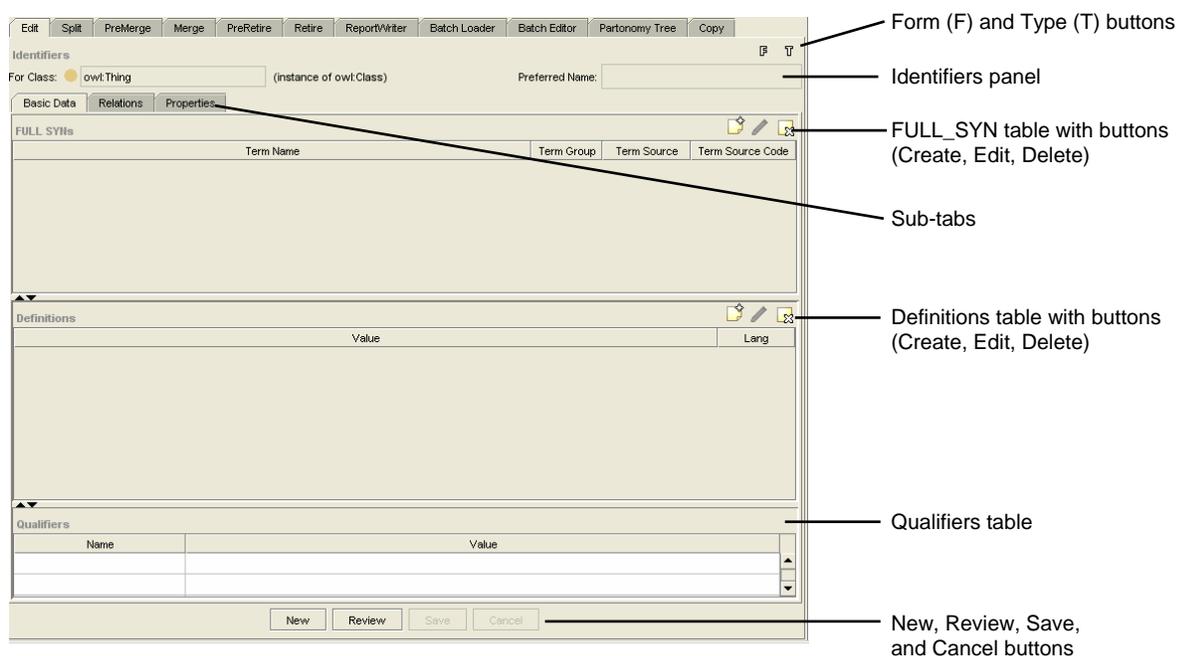


Figure 5. Edit tab

Table 1 describes each UI component and references relevant sections in this specification.

Table 1. UI components of the Edit tab

UI Component	Description
Identifiers panel	Shows class identifiers.
Form button (F)	Provides a shortcut to the original OWL Form Tab.
Type button (T)	Opens an OWL plug-in Class Editor window, which enables users to see the Inferred View of the selected class.
Basic Data sub-tab	See <i>The Basic Data Sub-tab</i> on page 9.
Relations sub-tab	See <i>The Relations Sub-tab</i> on page 12.
Properties sub-tab	See page <i>The Properties Sub-tab</i> on page 16.
New button	Creates a new class.

UI Component	Description
Review button	Launches a window for previewing all data of the selected class. For more details, see <i>The Review Window</i> on page 20.
Save button	Saves all changes made to the selected class to the knowledge base.
Cancel button	Discards all actions performed on the selected class.

2.1.1 Conventions for Buttons

Many of the tabbed components include buttons in the header areas and on a button panel at the bottom of the interface. Buttons are sometimes unavailable until a user performs a task such as selecting a table row. For tabs that use a Class Hierarchy, a button may be unavailable until a user selects a class.

2.1.2 The Create, Edit, and Delete Procedures

Each of the sub-tabs under the Edit tab is divided into various UI components. Each of these components has its own header area, which includes buttons for creating, editing, and deleting data. The buttons are generally available after certain other actions occur (e.g., the user selects a table row or specifies values in a dialog box). The procedures triggered by these buttons are similar for each UI component, with minor variations that are noted throughout this specification. Following are the basic steps for each procedure.

2.1.2.1 Creating Data

To create data, users follow these steps:

1. Click the **Create** button in the component header area.
This action opens a dialog box for creating data.
2. Specify name and qualifier values.
3. Click **OK** to create a new property, or click **Cancel** to abort the procedure.

If a user creates new data, the data appears in the table below the component header area.

2.1.2.2 Editing Data

To edit previously created data, users follow these steps:

1. Select the table row for the data to be edited.
2. Click the **Edit** button in the component header area.
This action opens a dialog box that is similar to the one used for creating data. The dialog box title includes the word *edit*, and the previously created data values are displayed.
3. Make any necessary changes.
4. Click **OK** to accept the changes, or click **Cancel** to abort the procedure.

2.1.2.3 Deleting Data

To delete previously created data, users follow these steps:

1. Select the table row for the data to be deleted.
2. Click the **Delete** button in the component header area.
A message box similar to the one shown in Figure 6 appears.
3. Click **Yes** to confirm deletion, or click **No** to cancel deletion.

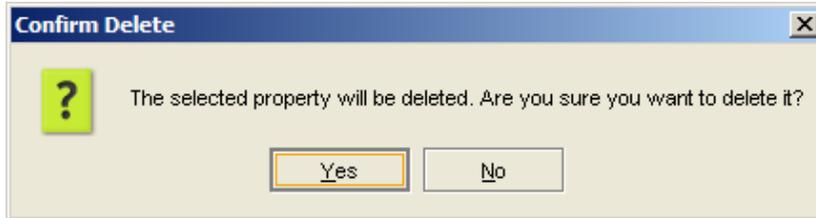


Figure 6. Message confirming deletion of data

2.1.3 The Basic Data Sub-tab

The Basic Data sub-tab is the first of the three Edit sub-tabs. Its layout is shown in Figure 5 on page 6. Table 2 describes each UI component and references relevant sections in this specification.

Table 2. UI components of the Basic Data sub-tab

UI Component	Description
FULL_SYN	Displays FULL_SYN values. The header area provides three buttons for creating, editing, and deleting FULL_SYN annotation properties. For more details, see <i>Using the FULL_SYN Component</i> on page 9.
Definitions	Displays definition values. The header area provides three buttons for creating, editing, and deleting definition properties. For more details, see <i>Using the Definitions Component</i> on page 11.
Qualifiers	Displays qualifier values for definition properties. To edit these values, users click the Edit button in the header area of the Definitions component.

2.1.3.1 Using the FULL_SYN Component

The FULL_SYN component of the Basic Data sub-tab provides a four-column table that displays the name and three qualifier values of a FULL_SYN property. The header area provides three buttons that enable users to create, edit, and delete FULL_SYN annotation properties.

Example 1 shows the FULL_SYN annotation property for a *Blood* class. The term name and three qualifiers (Term Group, Term Source, and Term Source Code) are stored in the knowledge base in XML format. The column on the right shows the corresponding XML value for the Term Name and the three qualifiers.

Example 1. FULL_SYN annotation

Class:	Blood	
Term Name:	peripheral blood	Syn_Term_Name
Term Group:	PT	Syn_Term_Type
Term Source:	NCI_GLOSS	Syn_Source
Term Source Code:	CDR0000046011	Syn_Source_Code

Note: Unlike the OWL plug-in, which displays property values in their native XML format, the Protégé extension displays these values under their respective column headings for easy identification.

The FULL_SYN component uses consistent dialog box interfaces for creating, editing, and deleting data:

- When a user clicks the **Create** button in the FULL_SYN header area, the Create New Annotation Property dialog box shown in Figure 7 opens. Users can specify values for term names and term qualifiers, then **OK** to accept them or **Cancel** to abort the procedure.
- When a user clicks the **Edit** button, a similar dialog box opens. The title bar of the dialog box reads *Edit an Annotation Property*, and the values of the selected property are displayed.
- When a user clicks the **Delete** button, the standard message prompt shown in Figure 6 on page 8 appears.

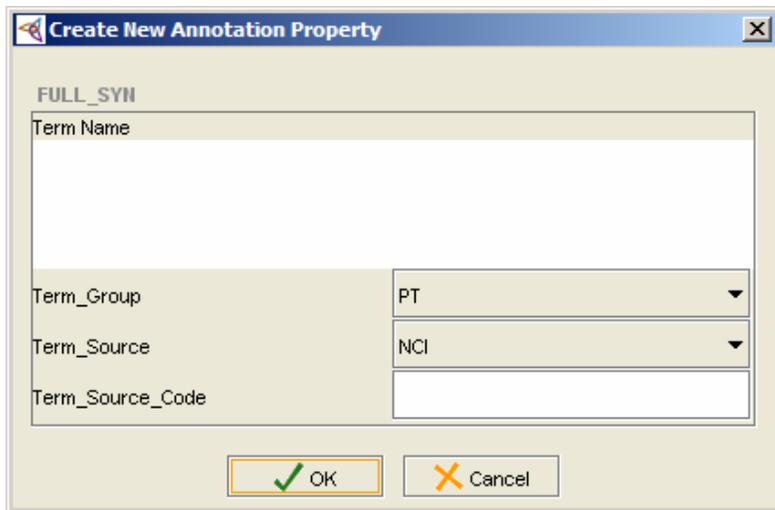


Figure 7. Create New Annotation Property dialog box for FULL_SYN data

2.1.3.2 Using the Definitions Component

The Definitions component of the Basic Data sub-tab provides a two-column table that displays the values for definition properties. Each definition property can have the following qualifiers:

- Definition attribution (optional)
- Definition reviewer name
- Definition review date
- Definition source

Note: These values appear in a separate table labeled *Qualifiers*. All entries in the Qualifier table are non-editable until a user clicks the **Edit** button in the Definitions area.

The Definitions component uses consistent dialog box interfaces for creating, editing, and deleting data:

- When a user clicks the **Create** button in the Definitions header area, the Create New Annotation Property dialog box shown in Figure 8 opens. Users can specify definition values, then click **OK** to accept them or **Cancel** to abort the procedure.
- When a user clicks the **Edit** button, a similar dialog box opens. The title bar of the dialog box reads *Edit an Annotation Property*, and the values of the selected property are displayed.
- When a user clicks the **Delete** button, the standard message prompt shown in Figure 6 on page 8 appears.

The screenshot shows a dialog box titled "Create New Annotation Property". It features a "DEFINITION" section with a text area for "Description (definition value)". Below this are four input fields: "Definition_Attribution" (empty), "Definition_Review_Date" (2006-07-23), "Definition_Reviewer_Name" (Kim Ong), and "Definition_Source" (NCI). At the bottom are "OK" and "Cancel" buttons.

Figure 8. Create New Annotation Property dialog box for definitions

2.1.4 The Relations Sub-tab

Figure 9 shows the layout of the Relations sub-tab.

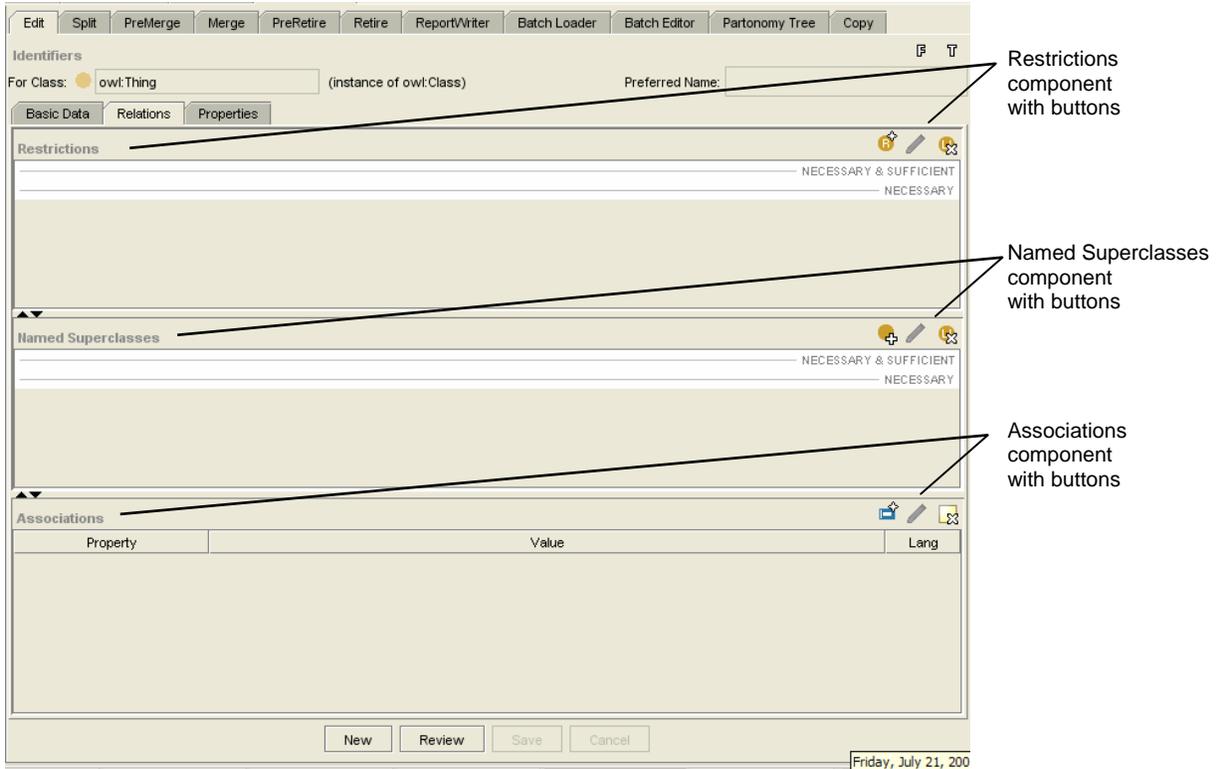


Figure 9. Relations sub-tab

Table 3 describes the sub-tab UI components and references relevant sections in this specification.

Table 3. UI Components of the Relations sub-tab

UI Component	Description
Restrictions	Displays restriction values. The header area provides three buttons for creating, editing, and deleting restrictions. For more details, see <i>Using the Restrictions Component</i> on page 13.
Named Superclasses	Displays named superclass values. The header area provides three buttons for creating, editing, and deleting named superclasses. For more details, see <i>Using the Named Superclasses Component</i> on page 14.
Associations	Displays association values. The header area provides three buttons for creating, editing, and deleting associations. For more details, see <i>Using the Associations Component</i> on page 15.

2.1.4.1 Using the Restrictions Component

The Restrictions component uses consistent dialog box interfaces for creating, editing, and deleting data:

- When a user clicks the **Create** button in the Restrictions header area, the Create a Restriction dialog box shown in Figure 10 opens. Users can specify restriction values, then click **OK** to accept them or **Cancel** to abort the procedure.
- When a user clicks the **Edit** button, a similar dialog box opens. The title bar of the dialog box reads *Edit a Restriction*, and the values of the selected property are displayed.
- When a user clicks the **Delete** button, the standard message prompt shown in Figure 6 on page 8 appears.

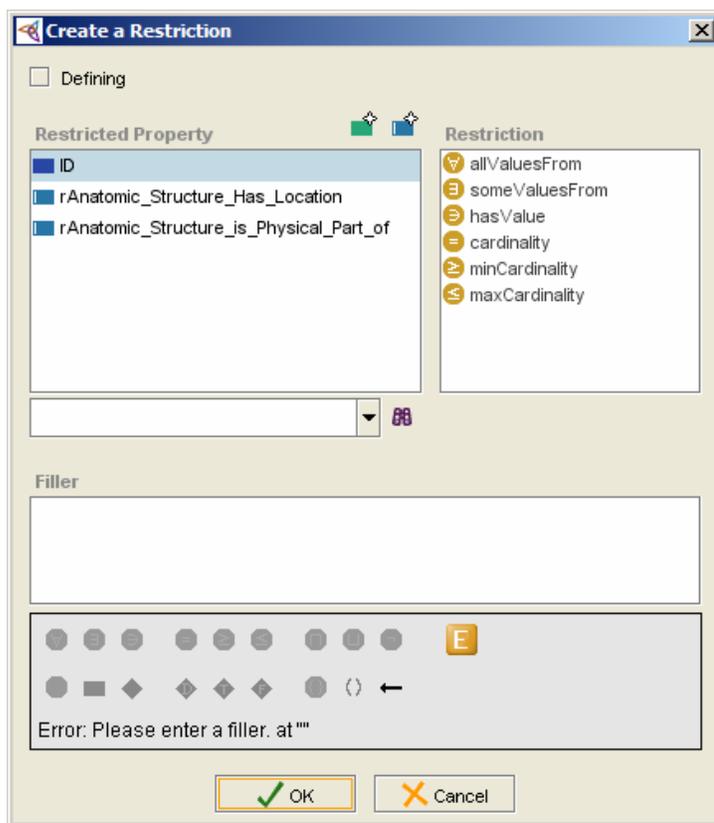


Figure 10. Create a Restriction dialog box

2.1.4.2 Using the Named Superclasses Component

The Named Superclasses component uses consistent dialog box interfaces for creating, editing, and deleting data:

- When a user clicks the **Create** button in the Named Superclasses header area, the Select a class dialog box shown in Figure 11 opens. Users can specify named superclass values, then click **OK** to accept them or **Cancel** to abort the procedure.
- When a user clicks the **Edit** button, a similar dialog box opens. The title bar of the dialog box reads *Edit a Named Superclass*, and the values of the selected property are displayed.
- When a user clicks the **Delete** button, the standard message prompt shown in Figure 6 on page 8 appears.

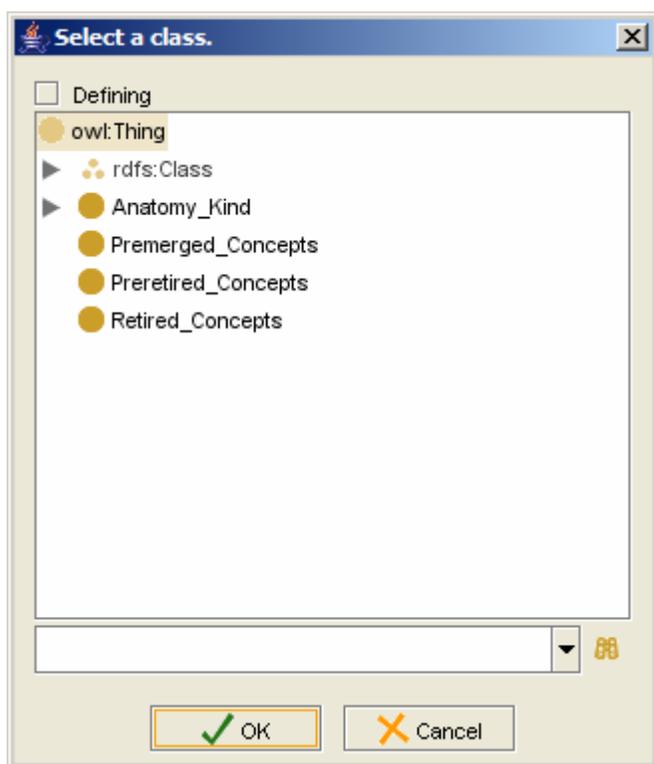


Figure 11. Select a class dialog box

2.1.4.3 Using the Associations Component

The Associations component uses consistent dialog box interfaces for creating, editing, and deleting data:

- When a user clicks the **Create** button in the Associations header area, the Add an Object-Valued Property dialog box shown in Figure 12 opens. Users can specify association values, then click **OK** to accept them or **Cancel** to abort the procedure.
- When a user clicks the **Edit** button, a similar dialog box opens. The title bar of the dialog box reads *Edit an Object-Valued Property*, and the values of the selected association are displayed.
- When a user clicks the **Delete** button, the standard message prompt shown in Figure 6 on page 8 appears.

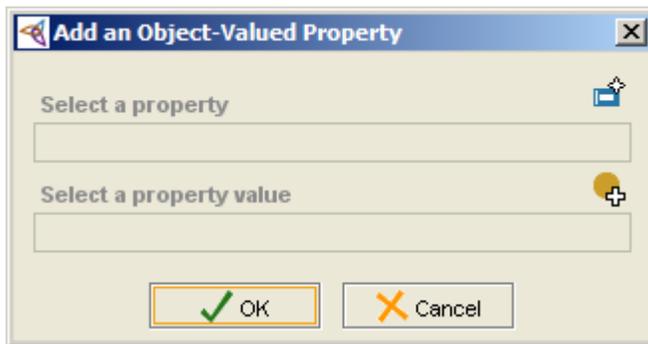


Figure 12. Add an Object-Valued Property dialog box

2.1.5 The Properties Sub-tab

Figure 13 shows the layout of the Properties sub-tab.

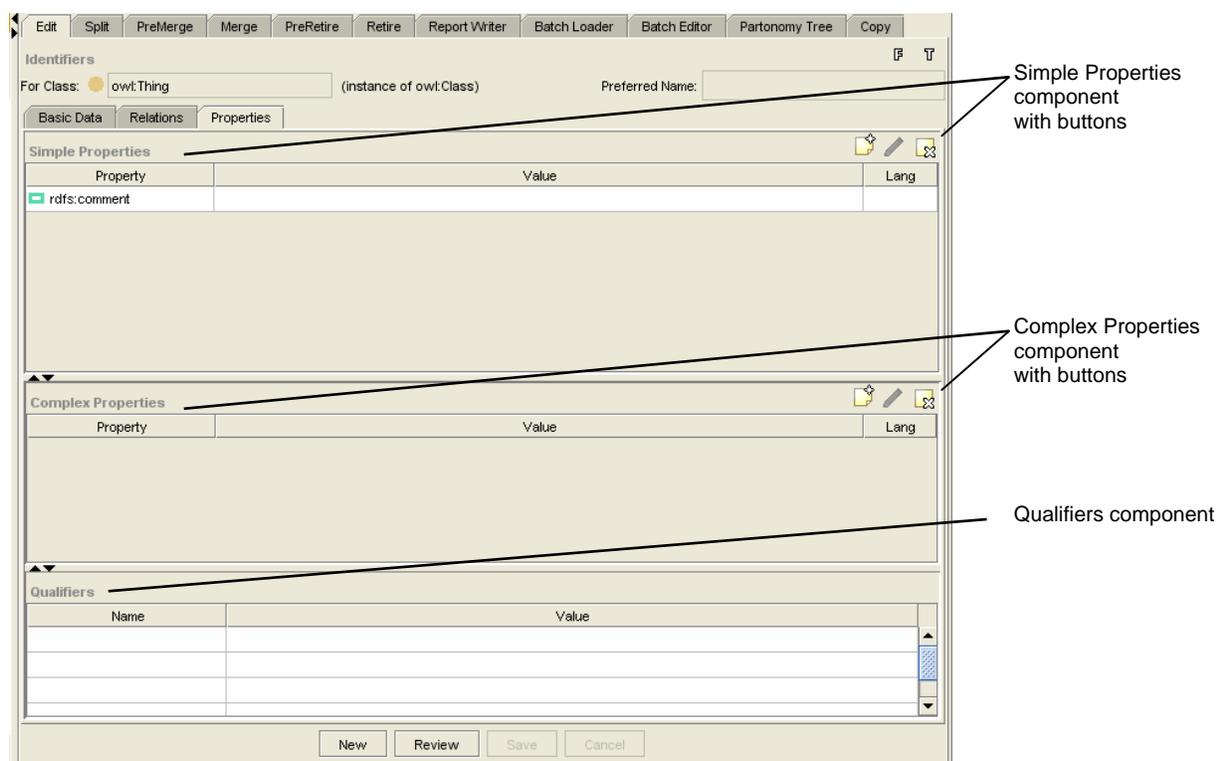


Figure 13. Properties sub-tab

Table 4 describes the sub-tab UI components and references relevant sections.

Table 4. UI Components of the Properties sub-tab

UI Component	Description
Simple Properties	Displays simple property values. The header area provides three buttons for creating, editing, and deleting simple properties. For more details, see <i>Using the Simple Properties Component</i> on page 17.
Complex Properties	Displays complex property values. The header area provides three buttons for creating, editing, and deleting complex properties. For more details, see <i>Using the Complex Properties Component</i> on page 18.
Qualifiers	Displays qualifier values for complex properties. To edit these values, users must click the Edit button in the header area of the Complex Properties component.

2.1.5.1 Using the Simple Properties Component

The Simple Properties component uses consistent dialog box interfaces for creating, editing, and deleting data:

- When a user clicks the **Create** button in the Restrictions header area, the Select Property dialog box shown in Figure 14 opens. The user follows these steps:
 - a. Selects a property value, then clicks **OK**. This action opens the Add Annotation dialog box shown in Figure 15.
 - b. Specifies a value for the new property.
 - c. Clicks **OK** to accept the new property, or clicks **Cancel** to abort the procedure.
- When a user clicks the **Edit** button, a similar dialog box opens. The title bar of the dialog box reads *Edit Annotation*, and the values of the selected property are displayed.
- When a user clicks the **Delete** button, the standard message prompt shown in Figure 6 on page 8 appears.

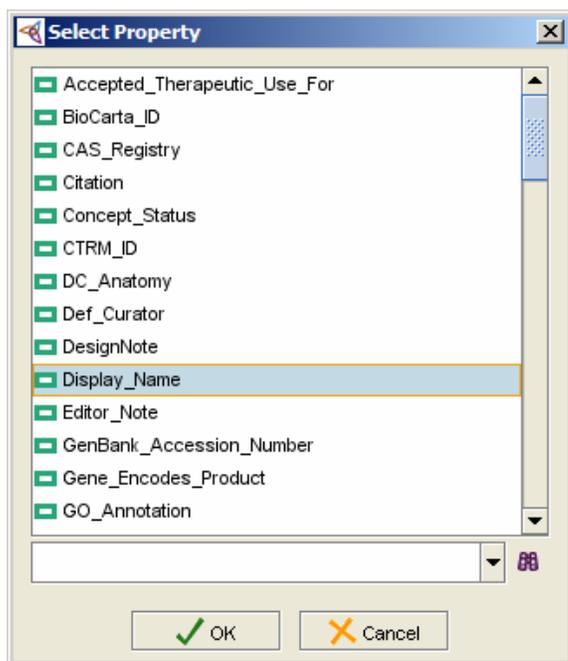


Figure 14. Select Property dialog box

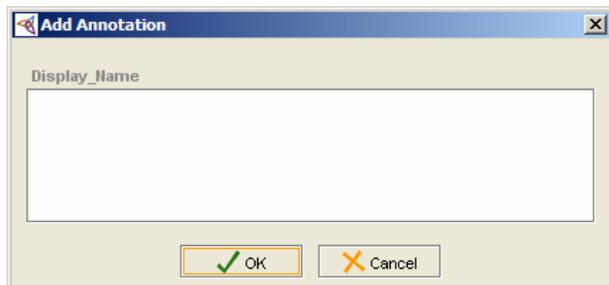


Figure 15. Add Annotation dialog box

2.1.5.2 Using the Complex Properties Component

Each complex annotation property can have many qualifiers. For example, a GO_ANNOTATION property can have the following qualifiers:

- GO_ID
- GO_Source_Data
- GO_Evidence
- GO_Source

A LONG_DEFINITION property can have the same qualifiers as the regular DEFINITION property described earlier:

- Definition attribution (optional)
- Definition reviewer name
- Definition review date
- Definition source

All qualifier names and values appear in a separate Qualifiers table shown in Figure 13 on page 16. To edit qualifier values, users click the **Edit** button in the header area of the Complex Properties component.

The Complex Properties component uses consistent dialog box interfaces for creating, editing, and deleting data:

- When a user clicks the **Create** button in the Restrictions header area, the Select Property dialog box shown in Figure 16 on page 19 opens. The user follows these steps:
 - a. Selects a property value, then clicks **OK**. This action opens the Create <Property Name> Annotation Property dialog box shown in Figure 17 on page 19.
 - b. Specifies a value for the new property.
 - c. Clicks **OK** to accept the new property, or clicks **Cancel** to abort the procedure.
- When a user clicks the **Edit** button, a similar dialog box opens. The title bar of the dialog box reads *Edit <Property Name> Annotation Property*, and the values of the selected property are displayed.
- When a user clicks the **Delete** button, the standard message prompt shown in Figure 6 on page 8 appears.

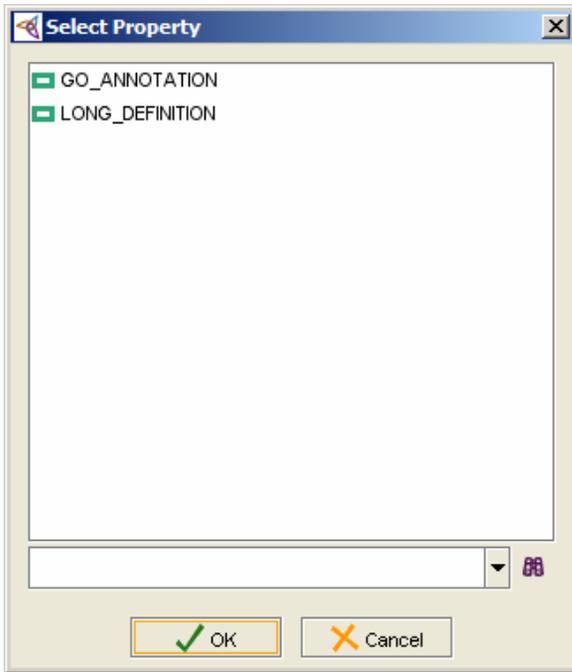


Figure 16. Select Property dialog box

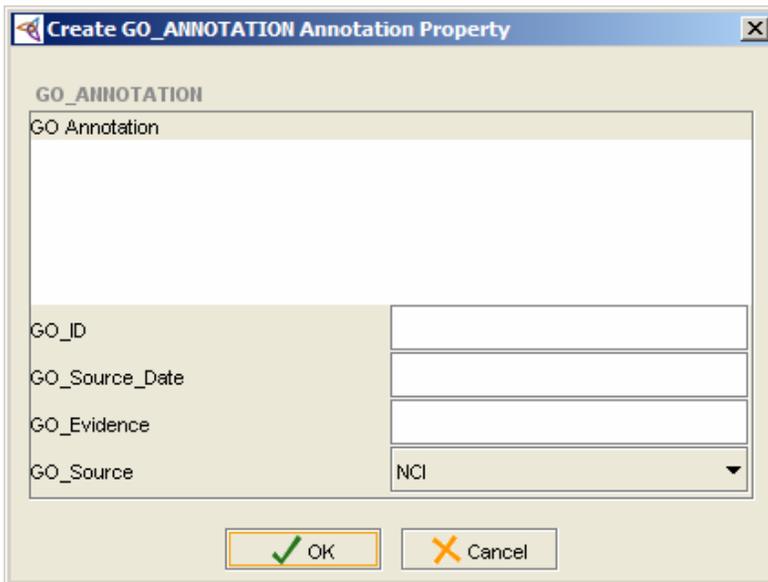


Figure 17. Create <Property Name> Annotation Property dialog box

2.1.6 The Review Window

The set of buttons at the bottom of the Edit tab includes a **Review** button. Clicking this button opens a window that displays all data of a selected class. The windows title bar reads *Review <Property Name>*.

Figure 18 shows an example of the Review window.

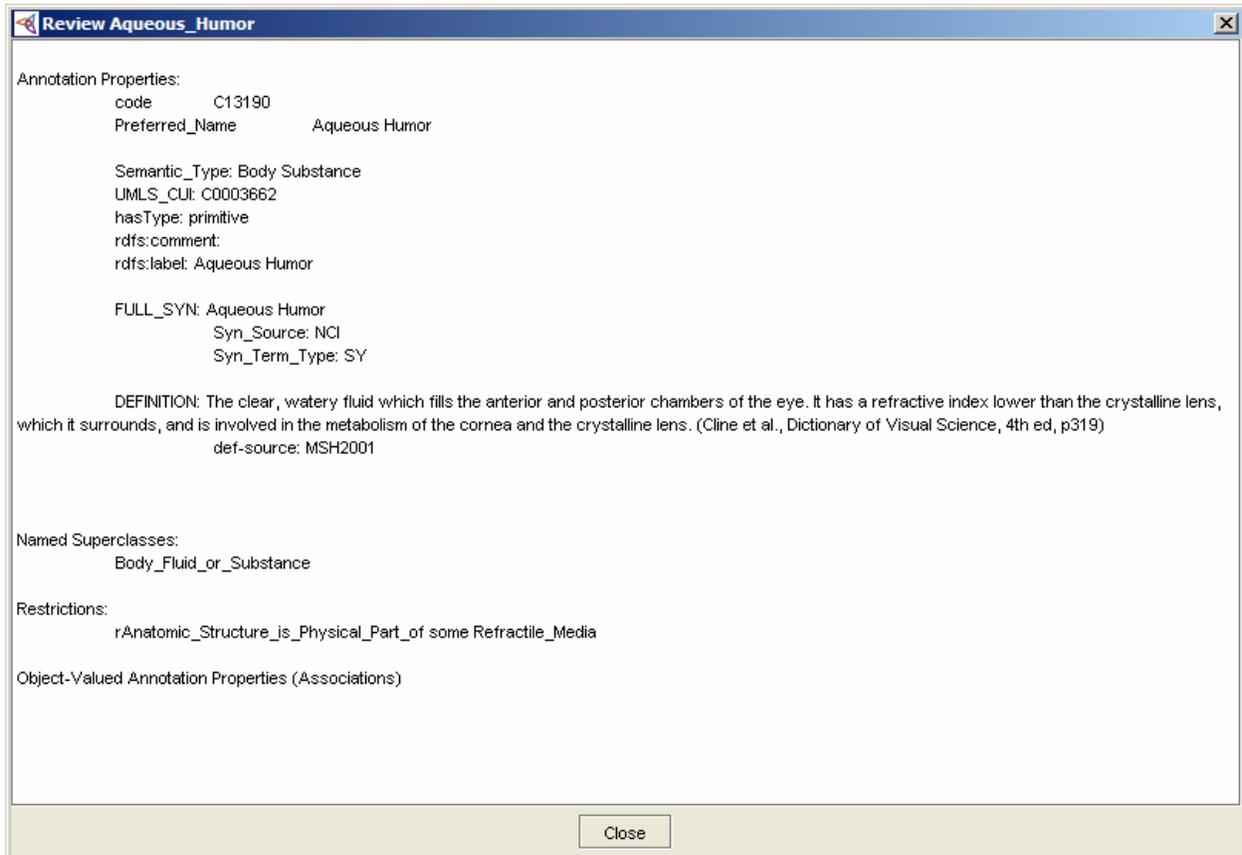


Figure 18. Review window

2.2 About the Split Tab

The Split tab includes two horizontal panes that display, respectively, existing concepts and new concepts. Figure 19 shows the design of this tab.

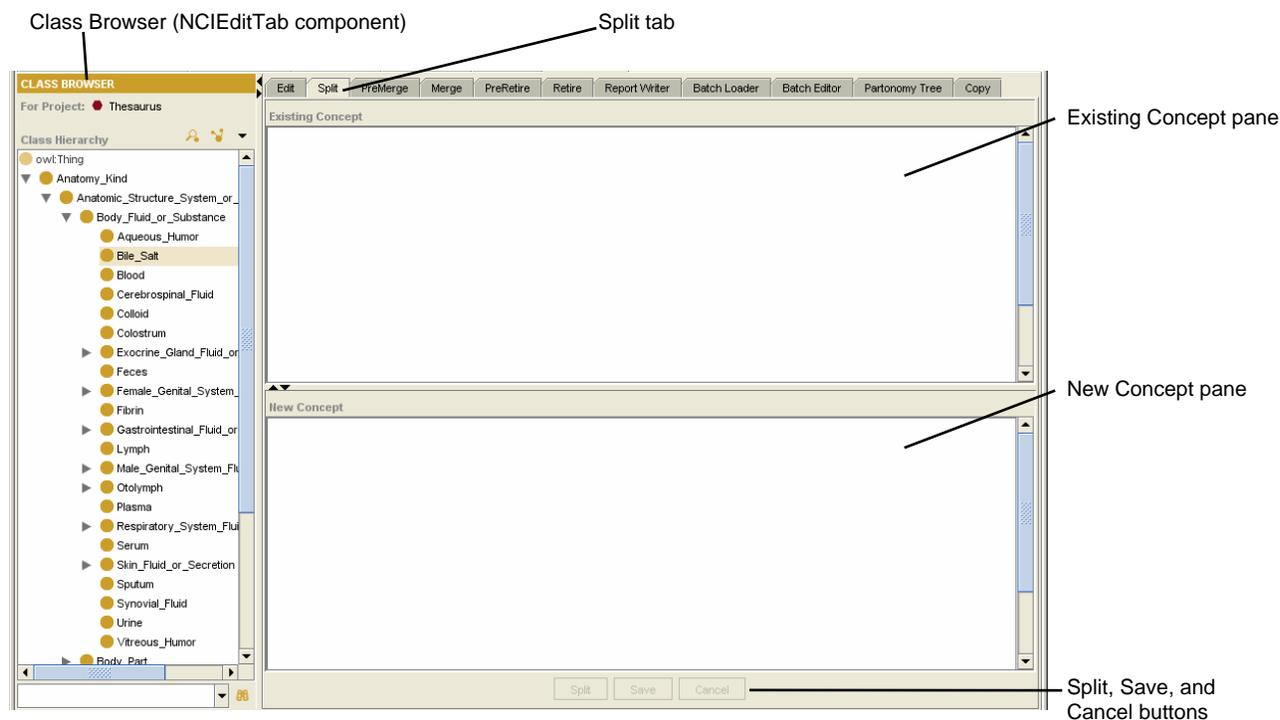


Figure 19. Split tab

Table 5 describes each UI component and references relevant sections in this specification.

Table 5. UI Components of the Split tab

UI Component	Description
Existing Concept pane	Shows a tree representation of a selected existing class. See Figure 20 on page 22.
New Concept pane	Shows a tree representation of a new class that has been cloned from an existing class.
Split button	Creates a new class based on the content of the class shown in the Existing Concept pane. This is known as <i>cloning a class</i> .
Save button	Saves a newly created class to the knowledge base.
Cancel button	Aborts the operation.

2.2.1 Using the Split Tab

Using the Split tab, users can perform the following tasks:

1. Drag a class from the Class Hierarchy on the left and drop it into the Existing Concept pane. Figure 20 shows a sample tree representation.



Figure 20. Tree representation of a class

2. Click the **Split** button to clone a new class based on the content of the selected existing class.

This action opens the Enter Class Identifiers dialog box shown in Figure 21. Here, users add a label and a preferred name for the new class, then click **OK**.



Figure 21. Enter Class Identifiers dialog box

The new class is represented by a tree appearing in the New Concept pane.

3. (Optional) Edit a class using a right-click menu that offers the following commands:
 - Expand
 - Delete
 - Modify Property
 - Modify Restriction
 - Add Property
 - Add Restriction
 - Add Parent
 - Add Association
4. Click the **Save** button to save the new class to the knowledge base, or Click the **Cancel** button abort the operation.

2.2.2 Assigned Properties for a Split Class

To establish an audit trail, the new class is assigned a *Split_From* annotation property. The value of the property is the code of the original class.

2.3 About the Pre-Merge Tab

The Pre-Merge tab enables users to flag two classes for a merge. Figure 22 shows the design of this tab.

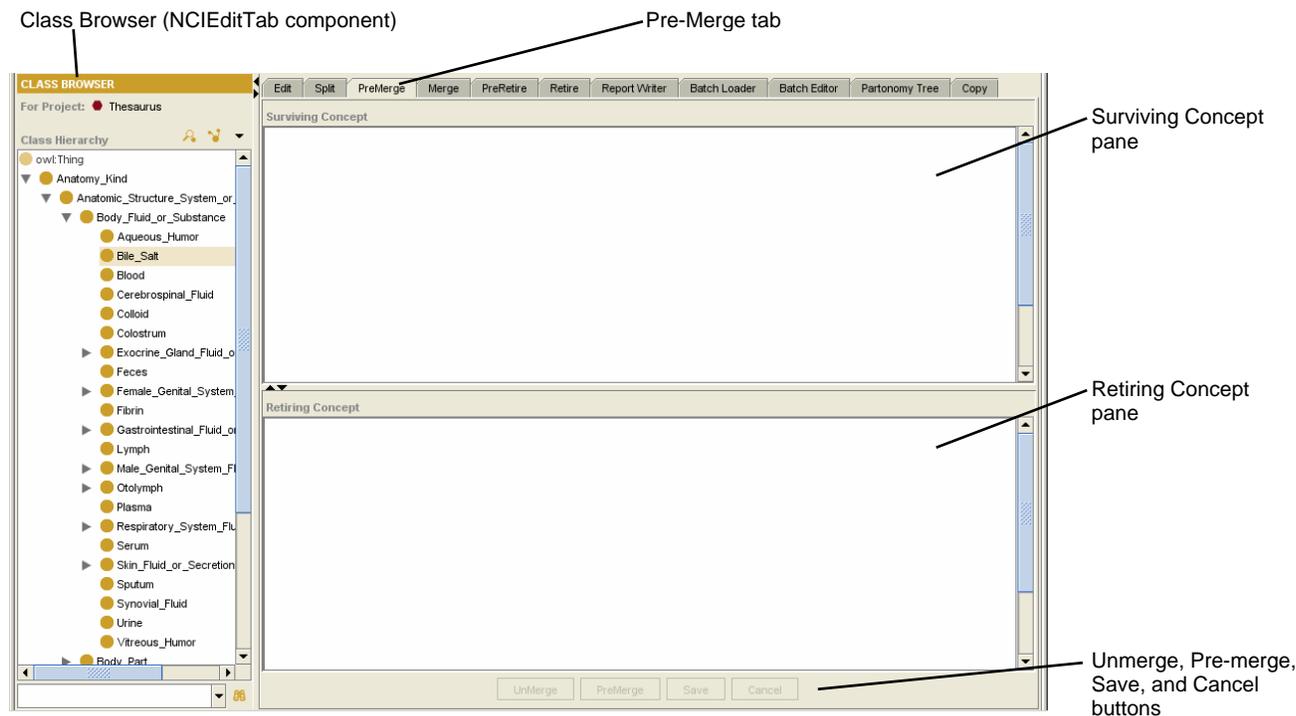


Figure 22. Pre-Merge tab

Table 6 describes each UI component and references relevant sections in this specification.

Table 6. UI Components of the Pre-Merge tab

UI Component	Description
Surviving Concept pane	Shows a tree representation of a selected class that has been designated as a <i>surviving</i> class.
Retiring Concept pane	Shows a tree representation of a selected class that has been designated as a <i>retiring</i> class.
Unmerge button	Removes a previously set pre-merge flag.
Pre-merge button	Flags two classes for a merge.
Save button	Saves concepts with the properties described in <i>Assigned Properties for a Merged Class</i> on 24.
Cancel button	Reverses a pre-merge operation. All users can perform this function.

2.3.1 Using the Pre-Merge Tab

Using the Pre-Merge tab, users follow these steps:

1. Drag a class from the Class Hierarchy on the left and drop it into the Surviving Concept pane.
2. Drag a class from the Class Hierarchy on the left and drop it into the Retiring Concept pane.
3. Click the **Pre-Merge** button to flag the two selected classes for a merge.

This action opens the Enter Notes dialog box shown in Figure 23.

4. Specify values for an **Editor's Note** and a **Design Note**, then click **OK** to add the values to the retiring class as annotation properties; or

Click **Cancel** to abort the operation.

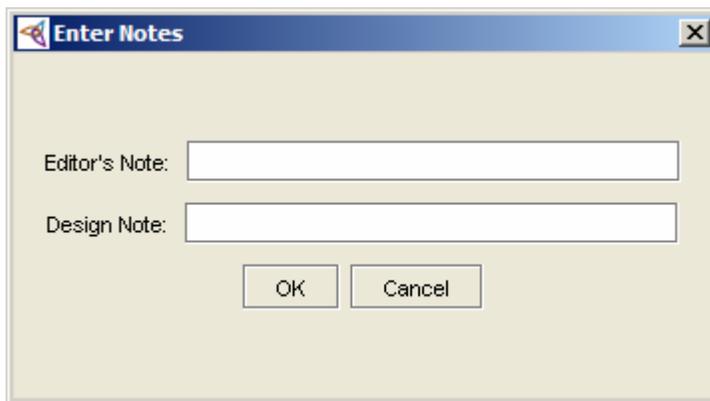


Figure 23. Enter Notes dialog box

Note: Only an authorized user such a lead editor can perform a merge action.

2.3.2 Assigned Properties for a Merged Class

For the purpose of cross-referencing, the following annotation properties are assigned:

- Surviving concept = *Merge_Source*
- Retiring concept = *Merge_Target*

Note: The value of each property is the code of the referenced class.

- For easy identification by the user who actually performs the merge action, a superclass called *Pre-retired_Concepts* is assigned to the retiring concept until the merge action is complete.

2.4 About the Merge Tab

The Merge tab enables authorized users to merge two concepts that have already been flagged for a merge using the pre-merge action. Figure 24 shows the design of this tab.



Figure 24. Merge tab

Table 7 describes each UI component and references relevant sections in this specification.

Table 7. UI Components of the Merge tab

UI Component	Description
Surviving Concept pane	Shows a tree representation of a selected class that has been flagged for a merge. This class has a <i>Merge_Source</i> annotation property.
Retiring Concept pane	Shows a tree representation of a selected class that has been designated as a <i>retiring class</i> .
Unmerge button	Removes a previously set pre-merge flag.
Merge button	Flags two classes for a merge.
Save button	Merges the classes and saves the changes to the knowledge base.
Cancel button	Reverses a pre-merge operation. All users can perform this function.

2.5 About the Pre-Retire Tab

The Pre-Retire tab enables users to suggest a class for retirement. The design of this tab is shown in Figure 25.

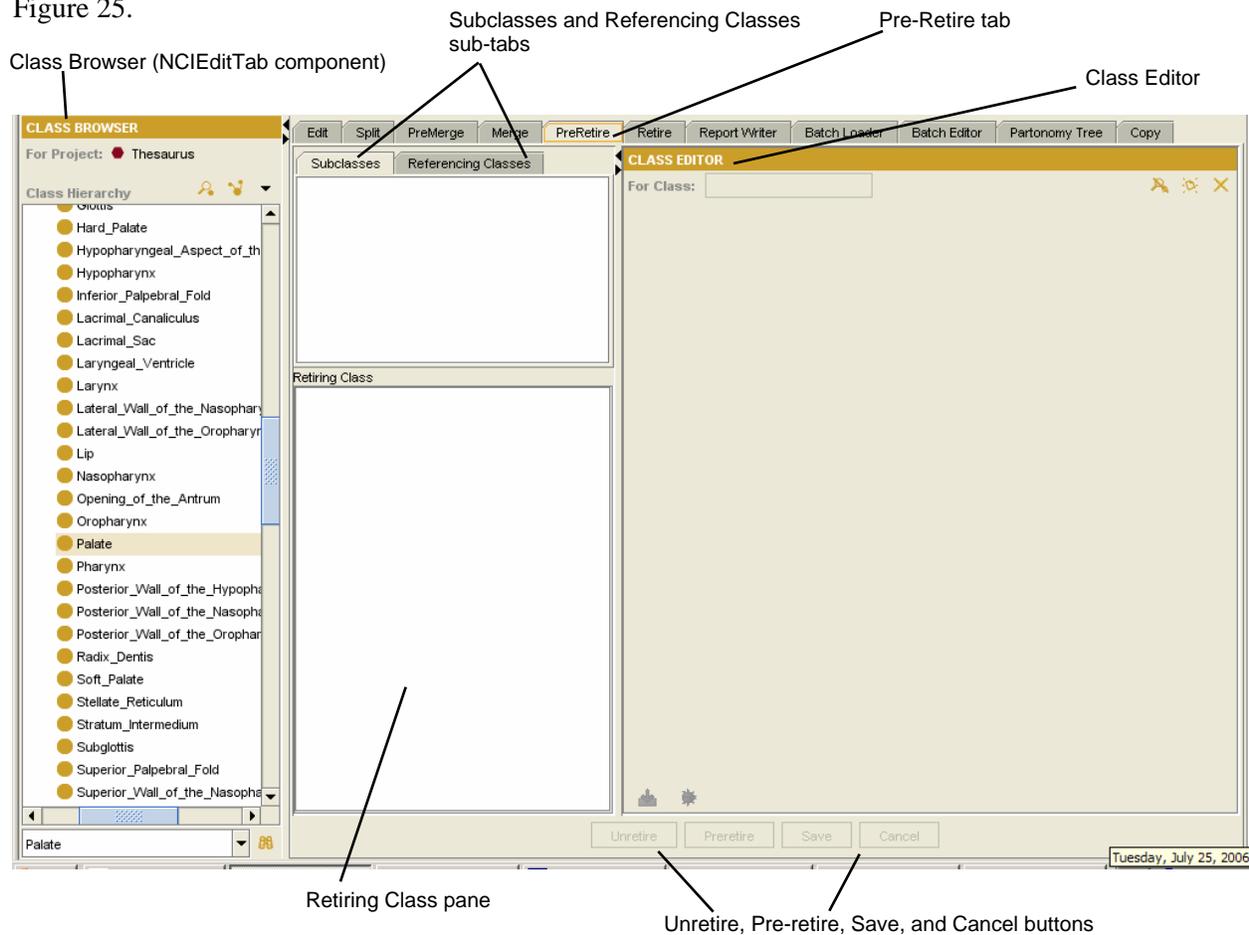


Figure 25. Pre-Retire tab

Table 8 describes each UI component and references relevant sections in this specification.

Table 8. UI Components of the Pre-Retire tab

UI Component	Description
Subclasses sub-tab	Shows the names of all subclasses of the currently selected class.
Referencing Classes sub-tab	Shows the names of all subclasses that have the currently selected class as a referred class.
Retiring Classes pane	Shows a tree representation of a selected class that has been designated as a <i>retiring</i> class.
Class Editor	Used to re-tree a selected subclass (that is, assign the subclass to another superclass) or display and modify a selected referring class. This component is an existing OWL plug-in.
Unretire button	Reverses a pre-retire action.
Pre-retire button	Flags a selected class for retirement. Similar to the pre-merge action, the user provides a Scope Note and Editor Note. See Figure 23 on page 24.
Save button	Saves a selected class to the knowledge base with a pre-retire flag.
Cancel button	Discards changes made to a selected class.

2.5.1 Eligibility of Classes for Retirement

A class is eligible for retirement when it meets the following conditions:

- It does not have any subclasses.
- It is not the value of any restriction that belongs to any other class.

In other words, all subclasses of this class must be re-treed (that is, reassigned to another superclass), and all restrictions belonging to a referring class must either be removed or redirected to another class before retirement of a class can take place.

2.5.2 Using the Pre-Retire Tab

Using the Pre-Retire tab, users can perform the following tasks:

- Drag a class from the Class Hierarchy on the left and drop it into the Retiring Class pane. This action triggers the following events:
 - The Subclasses sub-tab shows the names of all subclasses of the selected class.
 - The Referring Classes sub-tab shows the names of all other classes that have the selected class as a referred class.

Example 2. Referred class

Assume that the class *Plasma* has the restriction *Anatomic_Structure_is_Physical_part_of* Blood. When a user drags the class *Blood* from the Class Hierarchy to the Retiring Class Tree Panel, the class name *Plasma* appears in the Referencing Class tab. This is because *Plasma* has a restriction with a filler value equaling the class name of the selected class *Blood*.

- Re-tree a subclass by selecting the name of that subclass from the Subclasses sub-tab. This action causes data about the selected subclass to appear in the Class Editor panel. Users can then use this panel to re-tree the subclass.
- Modify the restrictions of a subclass by selecting a subclass name from the Referring Classes sub-tab. This action causes the data about the selected subclass to appear in the Class Editor panel. Users can then modify the restrictions to satisfy the pre-conditions described in *Eligibility of Classes for Retire*.

Note: Like the merge action, only authorized users can perform a retire action.

2.6 About the Retire Tab

The Retire tab enables authorized users to perform a retire action on a class that has been flagged by a pre-retire action. Figure 26 shows the design of the Retire tab.

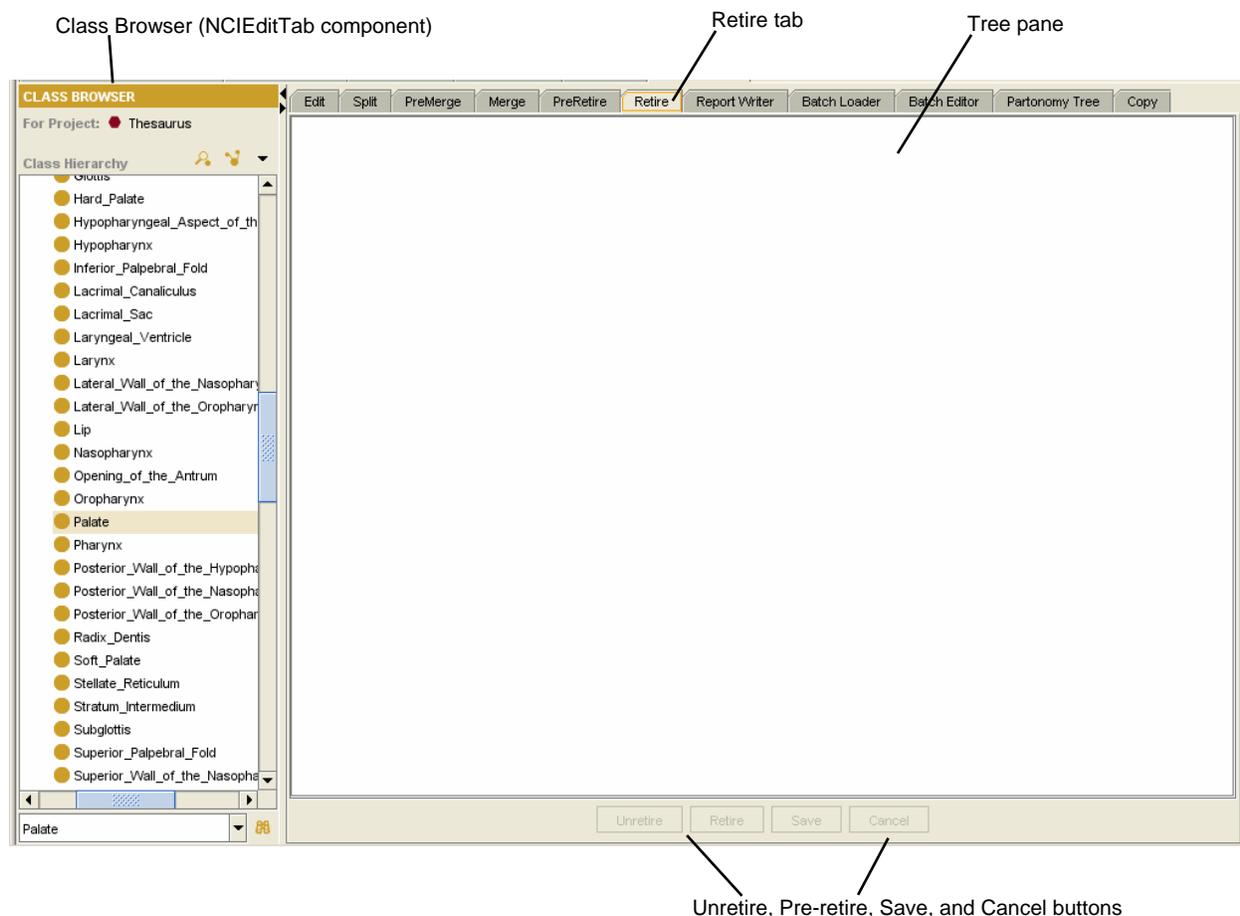


Figure 26. Retire tab

Table 9 describes each UI component and references relevant sections in this specification.

Table 9. UI Components of the Retire tab

UI Component	Description
Tree pane (right side)	Shows a tree representation of a selected class that has been flagged for retirement. The user drags a class from the Class Hierarchy and drops it into this pane.
Unretire button	Removes a previously set pre-retire flag.
Retire button	Retires a class.
Save button	Saves changes to the knowledge base and formally retires the class by re-treeing it from <i>pre-retired</i> to <i>retired</i> .
Cancel button	Aborts the retire operation.

2.7 About the Report Writer Tab

The Report Writer tab enables users to produce reports for selected classes. Figure 27 shows the design of this tab.

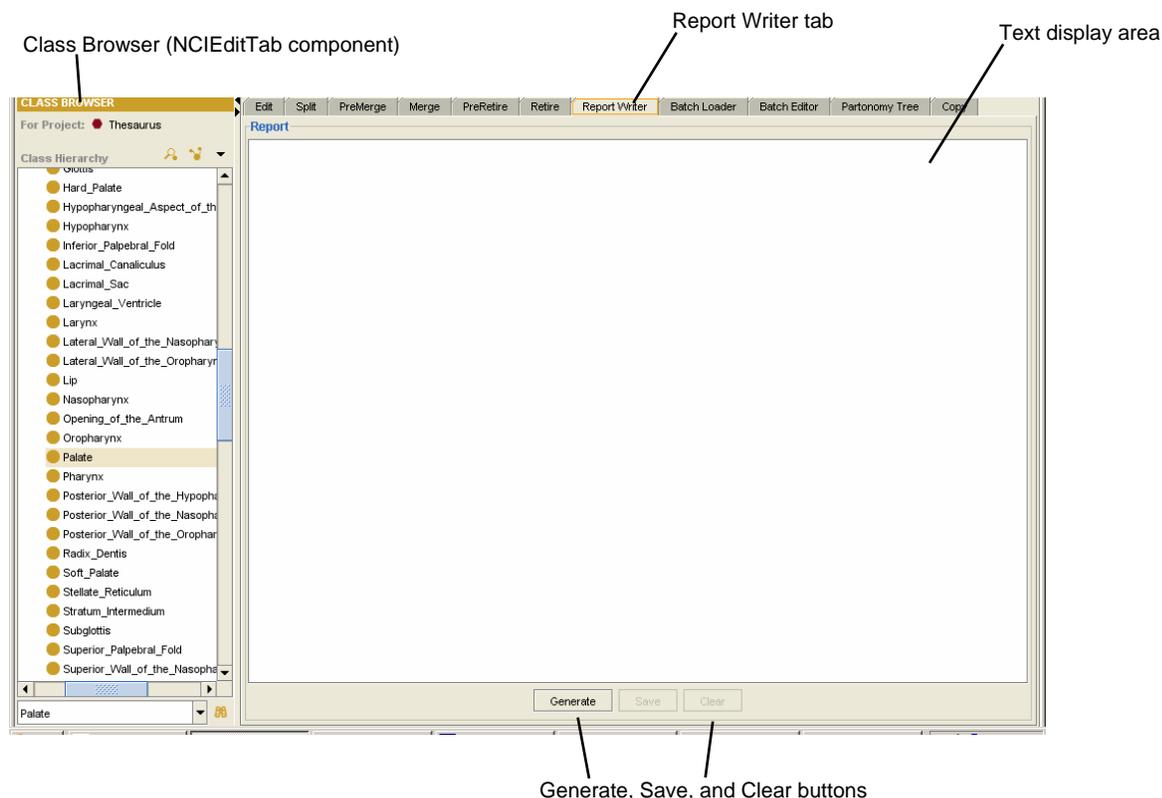


Figure 27. Report Writer tab

Table 10 describes each UI component and references relevant sections in this specification.

Table 10. UI Components of the Report Writer tab

UI Component	Description
Text display area (right side)	Displays the content of a report generated from a selected class in the Class Hierarchy.
Generate button	Triggers the display of report content for a selected class.
Save button	Saves the report to an ASCII file.
Clear button	Clears the text display area.

Figure 28 shows a sample report generated from the class *Lymph_Node_Sinus*.

```
Lymph_Node_Sinus
  rdfs:subClassOf: Lymph_Node_Tissue
  rdfs:subClassOf: Sinus
  rdfs:label: Lymph Node Sinus
  rdf:type: owl:Class
  code: C33033
  FULL_SYN: Lymph Node Sinus|PT|NCI
  Semantic_Type: Tissue
  Preferred_Name: Lymph Node Sinus
  hasType: primitive
  ID: 33033

Lymph_Node_Subcapsular_Sinus
  rdfs:subClassOf: Lymph_Node_Sinus
  rdfs:label: Lymph Node Subcapsular Sinus
  rdf:type: owl:Class
  code: C33642
  FULL_SYN: Subcapsular Sinus|SY|NCI
  FULL_SYN: Lymph Node Subcapsular Sinus|PT|NCI
  Semantic_Type: Tissue
  Preferred_Name: Lymph Node Subcapsular Sinus
  hasType: primitive
  ID: 33642
```

Figure 28. Sample report: Lymph_Node_Sinus

2.8 About the Batch Loader Tab

The Batch Loader tab loads a batch of classes into the knowledge base. Figure 29 shows the design of this tab.

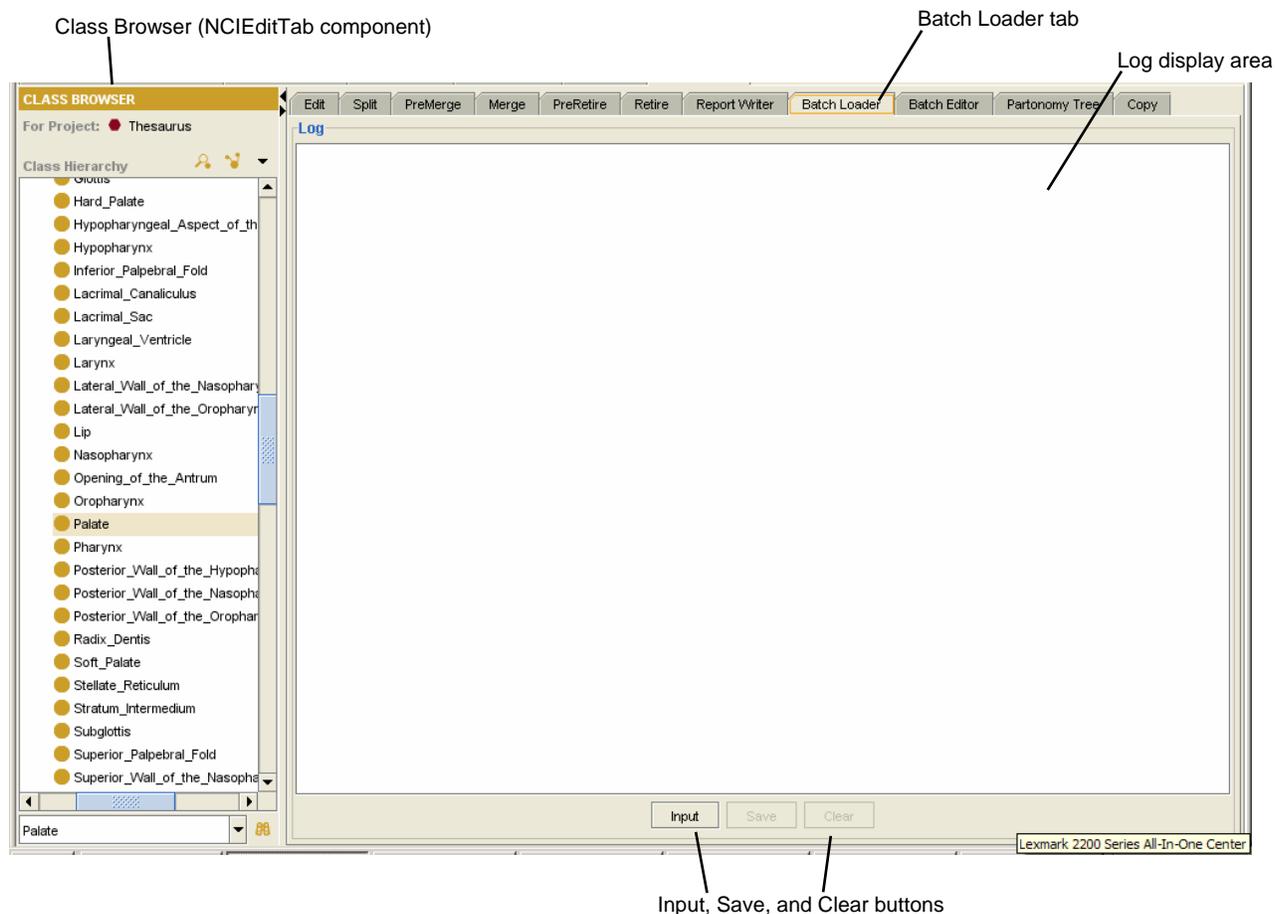


Figure 29. Batch Loader tab

Table 11 describes each UI component and references relevant sections in this specification.

Table 11. UI Components of the Batch Loader tab

UI Component	Description
Log display area (right side)	Displays the log status (that is, the status of the batch load).
Input button	Accepts a tab-delimited input file and starts the batch load process.
Save button	Saves the log to an ASCII file.
Clear button	Clears the text display area.

2.8.1 Using the Batch Loader Tab

2.8.1.1 Preparing a Batch Input File

To load a batch of classes into the knowledge base, users must first prepare a tab-delimited input file as shown in Figure 30. Each input line should contain a subclass name, the preferred name of the subclass, and the superclass of the subclass.

```
Olfactory_Cistern_sub_1 Olfactory_Cistern_pt_1 Olfactory_Cistern
Olfactory_Cistern_sub_2 Olfactory_Cistern_pt_2 Olfactory_Cistern
Olfactory_Cistern_sub_3 Olfactory_Cistern_pt_3 Olfactory_Cistern
Olfactory_Cistern_sub_4 Olfactory_Cistern_pt_4 Olfactory_Cistern
Olfactory_Cistern_sub_5 Olfactory_Cistern_pt_5 Olfactory_Cistern
Olfactory_Cistern_sub_6 Olfactory_Cistern_pt_6 Olfactory_Cistern
Olfactory_Cistern_sub_7 Olfactory_Cistern_pt_7 Olfactory_Cistern
Olfactory_Cistern_sub_8 Olfactory_Cistern_pt_8 Olfactory_Cistern
```

Figure 30. Batch Loader input file format

2.8.2 Loading a Batch File

To load the prepared file, users follow these steps:

1. Click the **Input** button.

The dialog box shown in Figure 31 appears.

2. Click the **Browse** buttons to locate the input file and an output log file.
3. Click the **Start** button.

A progress bar shows that the load has begun. The batch load process produces the following results:

- The classes appearing in the first field of each input line are created and placed under a superclass as specified in the third field of each input line.
 - Each class is automatically assigned annotation properties.
 - When the process is completed, a log similar to the one shown Figure 32 (on the following page) appears in the text area.
4. (Optional) Click the **Save** button to save the log to an ASCII file.

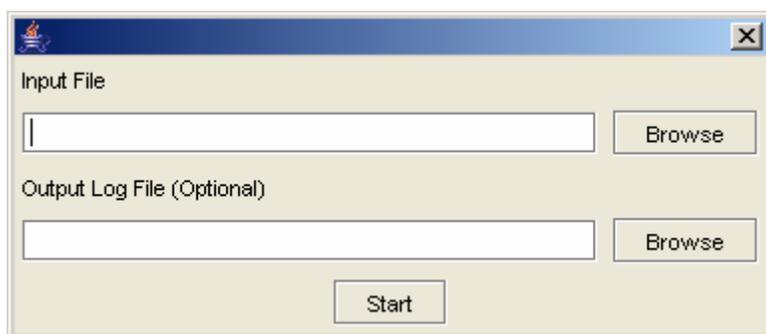


Figure 31. Batch Loader dialog box

Creating Olfactory_Cistern_sub_1
Olfactory_Cistern_sub_1 created.

Creating Olfactory_Cistern_sub_2
Olfactory_Cistern_sub_2 created.

Creating Olfactory_Cistern_sub_3
Olfactory_Cistern_sub_3 created.

Creating Olfactory_Cistern_sub_4
Olfactory_Cistern_sub_4 created.

Creating Olfactory_Cistern_sub_5
Olfactory_Cistern_sub_5 created.

Creating Olfactory_Cistern_sub_6
Olfactory_Cistern_sub_6 created.

Creating Olfactory_Cistern_sub_7
Olfactory_Cistern_sub_7 created.

Creating Olfactory_Cistern_sub_8
Olfactory_Cistern_sub_8 created.

Figure 32. Sample batch loader log file

2.9 About the Batch Editor Tab

The Batch Editor tab enables users to edit a batch of classes. Figure 33 shows the design of this tab.

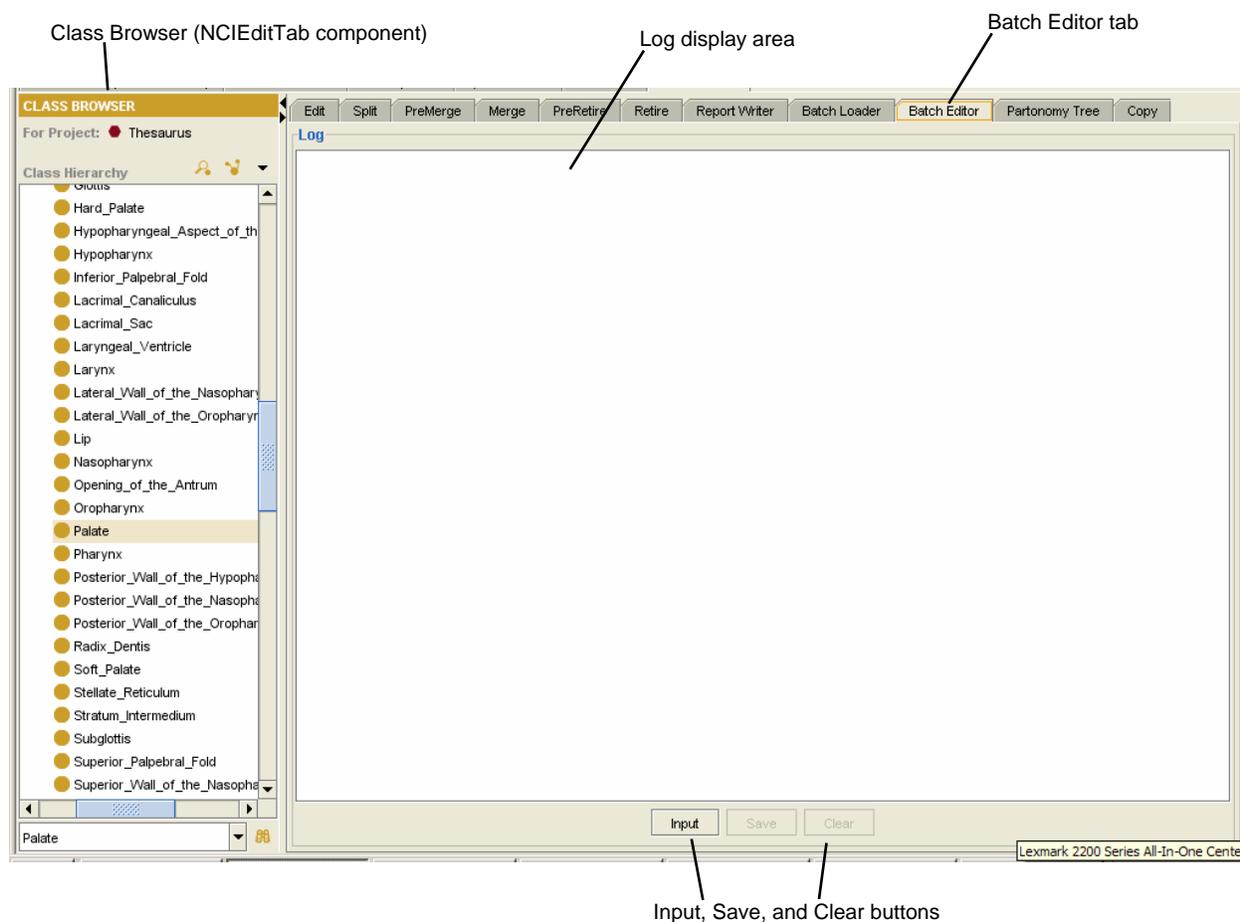


Figure 33. Batch Editor tab

Table 12 describes each UI component and references relevant sections in this specification.

Table 12. UI Components of the Batch Editor tab

UI Component	Description
Log display area (right side)	Displays the log status (that is, the status of the batch edit process).
Input button	Accepts a tab-delimited input file and starts the batch edit process.
Save button	Saves the log to an ASCII file.
Clear button	Clears the text display area.

2.9.1 Using the Batch Editor Tab

2.9.1.1 Preparing a Batch Input File

To load a batch of classes into the knowledge base, users must first prepare a tab-delimited input file as shown in Figure 34.

```
Olfactory_Cistern      new      property      Synonym Olfactory_Cistern_Synonym_1
Olfactory_Cistern      new      property      Synonym Olfactory_Cistern_Synonym_2
Olfactory_Cistern      new      property      Synonym Olfactory_Cistern_Synonym_3
Olfactory_Cistern      new      property      Synonym Olfactory_Cistern_Synonym_4
Olfactory_Cistern      new      property      Synonym Olfactory_Cistern_Synonym_5
Olfactory_Cistern      new      property      Synonym Olfactory_Cistern_Synonym_6
Olfactory_Cistern      new      Role        rAnatomic_Structure_is_Physical_Part_of some|Chromosome_Band
Olfactory_Cistern      new      Role        rAnatomic_Structure_is_Physical_Part_of all|Blood_Vessel
Olfactory_Cistern      edit     property      Synonym Olfactory_Cistern_Synonym_6 Olfactory_Cistern_Synonym_6_modified
```

Figure 34. Batch Input file

Each input line of the batch file should contain the data elements specified in Table 13.

Table 13. Data elements for a batch file

Data Field Number	Description
1	Class Name
2	New/Edit
3	Attribute Type (Property/Role)
4	Attribute Name (Property Name/Role Name)
5	Old Value (and modifier if applicable)
6	New Value (and modifier if applicable)

2.9.2 Loading a Batch File

To load the prepared file, users follow these steps:

1. Click the **Input** button.

The dialog box shown in Figure 31 on page 32 appears.

2. Click the **Browse** buttons to locate the input file and an output log file.
3. Click the **Start** button.

A progress bar shows that the load has begun. The batch load process produces the following results:

- The classes appearing in the first field of each input line are created and placed under a superclass as specified in the third field of each input line.
 - Each class is automatically assigned some annotation properties.
 - When the process is completed, a log similar to the one shown Figure 35 (on the following page) appears in the text area.
4. (Optional) Click the **Save** button to save the log to an ASCII file.

Olfactory_Cistern Done.	new	property	Synonym	Olfactory_Cistern_Synonym_1	
Olfactory_Cistern Done.	new	property	Synonym	Olfactory_Cistern_Synonym_2	
Olfactory_Cistern Done.	new	property	Synonym	Olfactory_Cistern_Synonym_3	
Olfactory_Cistern Done.	new	property	Synonym	Olfactory_Cistern_Synonym_4	
Olfactory_Cistern Done.	new	property	Synonym	Olfactory_Cistern_Synonym_5	
Olfactory_Cistern Done.	new	property	Synonym	Olfactory_Cistern_Synonym_6	
Olfactory_Cistern Done.	new	Role	rAnatomic_Structure_is_Physical_Part_of		some Chromosome_Band
Olfactory_Cistern Done.	new	Role	rAnatomic_Structure_is_Physical_Part_of		all Blood_Vessel
Olfactory_Cistern Done.	edit	property	Synonym	Olfactory_Cistern_Synonym_6	Olfactory_Cistern_Synonym_6_modified

Figure 35. Sample Batch Editor log file

2.10 About the Partonomy Tree Tab

The Partonomy tree tab enables users to select a root class and display it as a partonomy tree. Figure 36 shows the design of this tab.

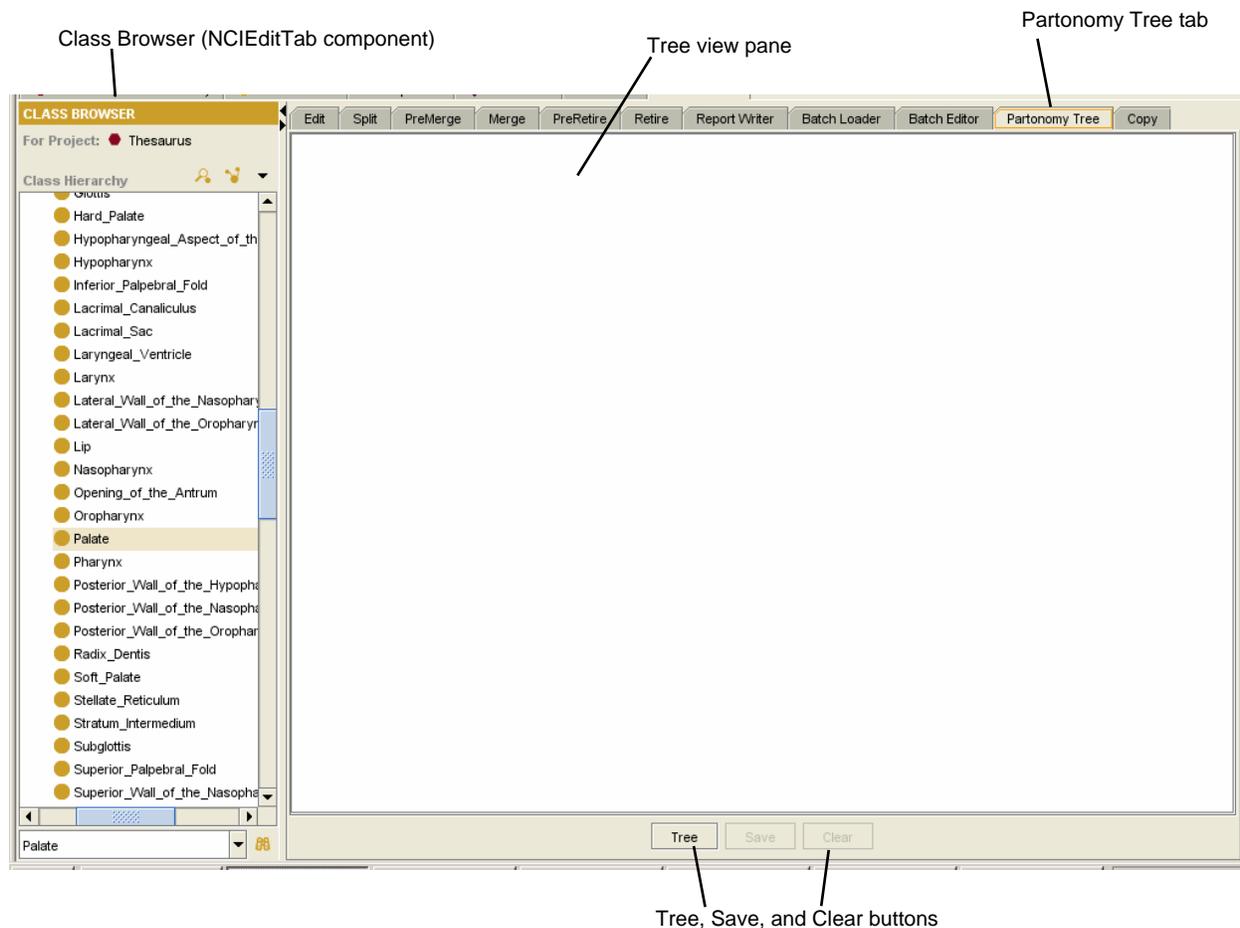


Figure 36. Partonomy Tree tab

Table 14 describes each UI component and references relevant sections in this specification.

Table 14. UI components of the Partonomy Tree tab

UI Component	Description
Tree view pane (right side)	Displays a partonomy tree generated from a selected root class.
Tree button	Triggers the generation of a partonomy tree from a selected root class.
Save button	Saves the partonomy tree as an ASCII file.
Clear button	Clears the tree view pane.

2.10.1 Partonomy Tree Defined

A partonomy tree shows a class to which other classes are connected by *part_of* relations.

2.10.2 Using the Partonomy Tree Tab

To generate a partonomy tree, users follow these steps:

1. Select a root class from the Class Hierarchy.
2. Click the **Tree** button.

This action opens the Select Transitive Properties dialog box, shown in Figure 37.

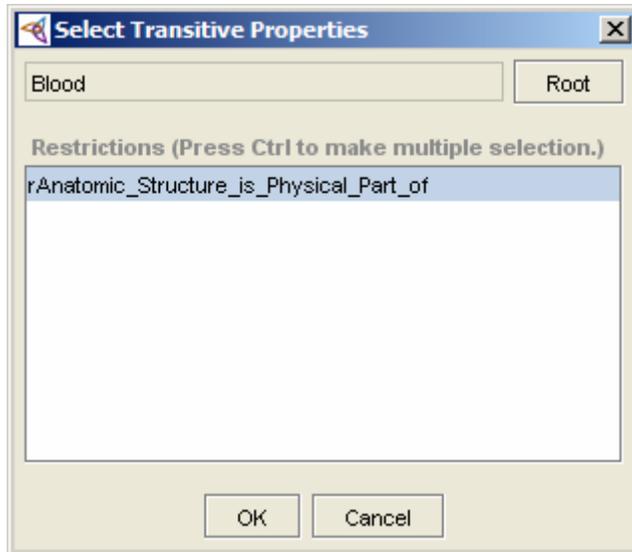


Figure 37. Select Transitive Properties dialog box

3. Select one or more restriction names from the list box.
4. Click **OK** to generate the partonomy tree.
5. (Optional) Click the **Save** button to save the tree to an ASCII file.

Figure 38 shows a sample partonomy tree stemming from the root class *Blood*.

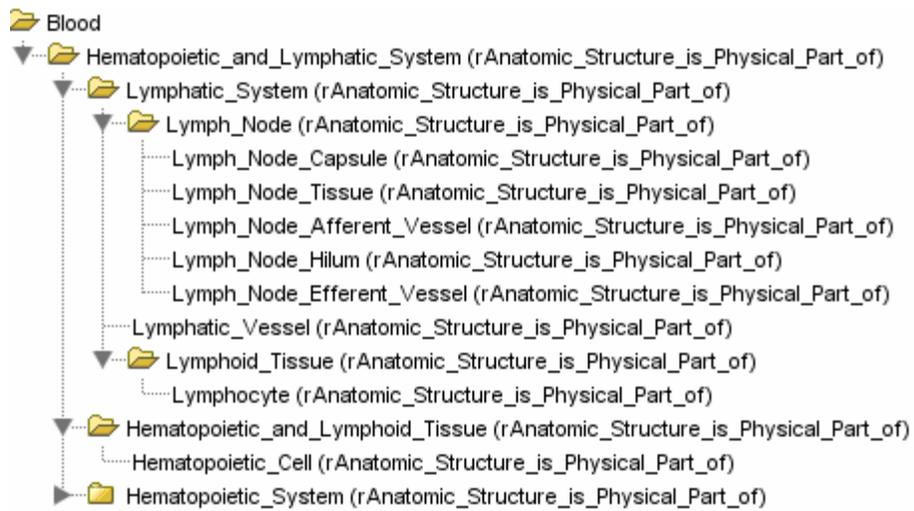


Figure 38. Sample partonomy tree

2.11 About the Copy Tab

The Copy tab enables users to edit two classes simultaneously and clone new classes from existing classes. Figure 39 shows the design of this tab.

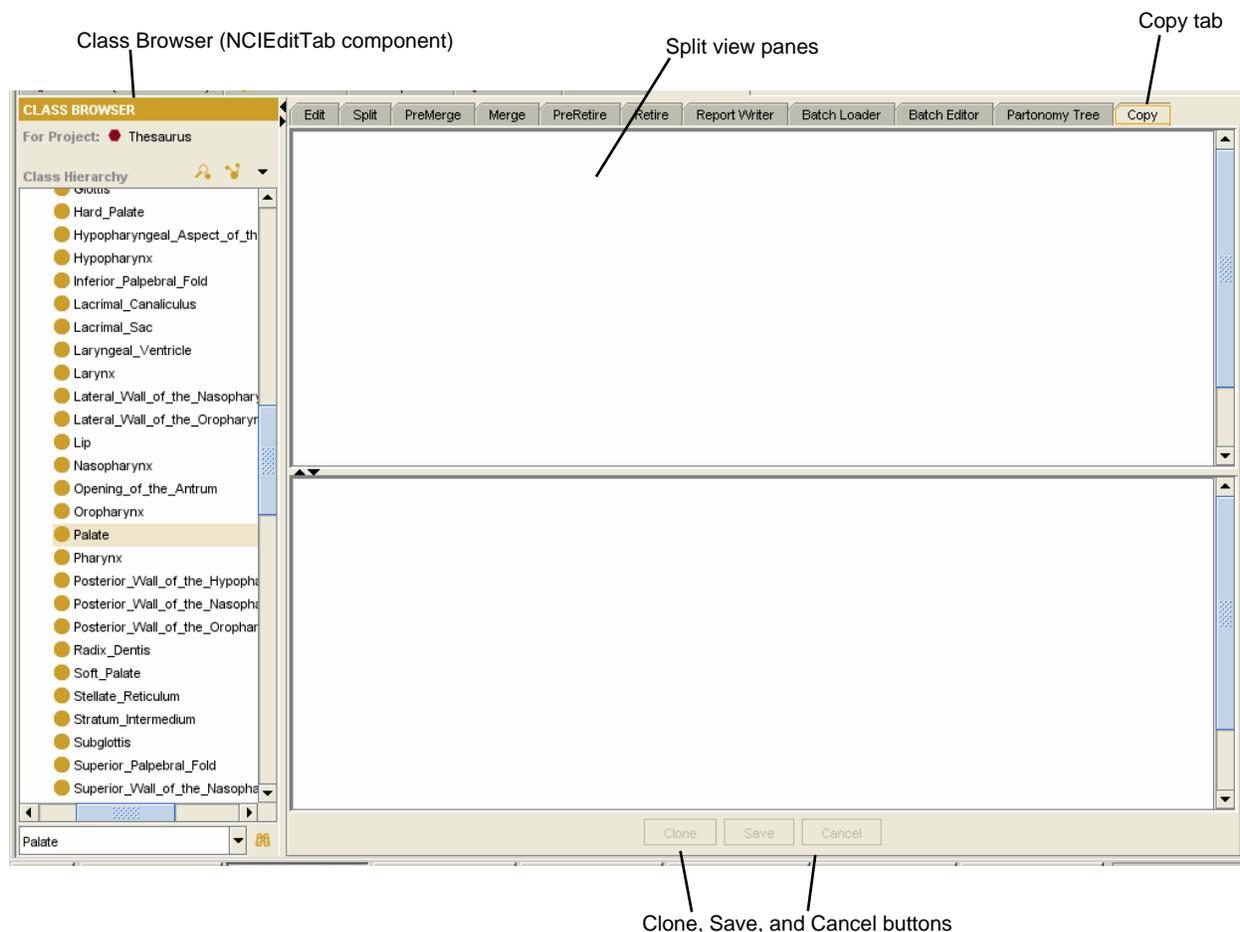


Figure 39. Copy tab

Table 15 describes each UI component and references relevant sections in this specification.

Table 15. UI components of the Copy tab

UI Component	Description
Split view panes (right side)	Display selected classes from the Class Hierarchy.
Clone button	Creates a clone of the selected class.
Save button	Saves changes to the knowledge base.
Clear button	Clears the split view panes.

2.11.1 Using the Copy Tab

To copy a class, users follow these steps:

1. Drag an existing class from the Class Hierarchy and drop it into the upper tree pane.
2. Click the **Clone** button to clone the selected class.

The Class Identifiers dialog box shown in Figure 21 on page 22 appears.

3. Specify a label and a preferred name for the new class, then click **OK**.

This action generates a new class. The tree representation of this new class is shown in the lower pane.

4. (Optional) Use the **Copy** and **Paste** commands to copy attributes from one class to another.
5. Click the **Save** button to save changes to the knowledge base.

2.11.2 Editing a Selected Class

Users can edit a selected class by right-clicking the pane in which the class is displayed. The right-click menu offers the following commands:

- | | |
|----------------------|-------------------|
| - Expand | - Add Property |
| - Delete | - Add Restriction |
| - Modify Property | - Add Parent |
| - Modify Restriction | - Add Association |

2.12 About the NCIEdit Tab Plug-In Utility Classes

The initialization of the NCIEdit Tab Plug-In instantiates the following classes:

- OWLWrapper
- NCICodeGenerator
- EventListener
- NCIEditFilter
- DataHandler

The utility of each class is described in Table 16.

Table 16. Utility classes

Class	Description
OWLWrapper	Contains common methods that are called by edit tabs.
NCICodeGenerator	Retrieves a code from the centralized code generator server.
EventListener	Detects events dispatched by the Protégé server when a class has been modified or a subclass has been added or removed by other users.
NCIEditFilter	Checks to determine whether a class meets all NCI-specific editing business rules.
DataHandler	Performs the Save action.

Table 17. NCIEditTab architecture/standards usage

NCIEditTab component	Level of use	Notes
Edit Tab	Fully uses	
Split	Fully uses	
Pre-merge	Fully uses	
Merge	Fully uses	Authorized user only
Pre-retire	Fully uses	
Retire	Fully uses	Authorized user only
Report Writer	Fully uses	
Batch Loader	Fully uses	
Batch Editor	Fully uses	
Partonomy Tree	Fully uses	
Copy	Fully uses	

Level of use

Fully uses/complies

Partially uses/complies

Does not use/comply

3. Performance Considerations

In this section, describe the product's degree of scalability. Note any known limits to the product's performance and scalability.

4. Third-party Tool Dependencies and Requirements

Table 18 describes the third-party tools on which this product depends. It includes the version of each product, as well as any database drivers that are supported.

Table 18. Third-party tool requirements/dependencies

Tool	Version	Used for
Base Protégé		Dependency
Protégé OWL Plug-In		Dependency
MySQL		Repository of knowledge base
Tomcat Servlet Engine		Publish ProtegeStartUp.xml
History Plug-Ins		Log history data
Other jar files		
Jdom.jar		
Xmlrpc.jar		
Xerces.jar		

5. Document Revision History

This section is used to record revisions to this document. Content changes to this document are indicated by revision bars (|) in the left margin unless a complete rewrite is indicated.

In the *Change Description and Explanation* column of the following change log, indicate the inspection date on which each document version is based. Entries in this log must be maintained for at least three years.

In the *Status* column, use one of the following designations:

- D = Draft
- A = For Review and Approval
- R = Re-approval, Plan of Record Change

To add a row to the end of the Change Log, click in the last cell of the table and press **TAB**.

Table 19. <Document Title> Document Change Log

Version/Date	Author	Change Description and Explanation	Status
mm/dd/yy	name	Text	D/A/R

6. Document Approval

6.1 Approvers List List names from Product Launch Process Responsibility Matrix.

The individuals whose names are listed in this section are the approvers for the <document title>. All approvers must provide written approval before the next steps in the product launch process can begin.

Table 20. Approvers for this document

Title	Name
Technical Director (for the product)	name
Senior Development Manager (for the product)	name
Solutions Manager (for the product)	name

6.2 Reviewers List List names from Product Launch Process Responsibility Matrix.

The individuals whose names are listed in this section are the reviewers for the <document title>. Though reviewers do not have to formally approve this document, all reviewers are encouraged to review and comment on the document. If necessary, reviewers should review only those sections that are relevant to their areas of responsibility, rather than reviewing the entire document.

Table 21. Reviewers for this document

Title	Name
Development Manager(s) for the product	name
caCORE architecture steering committee representative	name
Development team	name

Note to Authors: The End of Document indicator must remain in the document if single sequential page numbering is used. If page numbering style X of Y is used (e.g., 1 of 50), then the End Of Document indicator may be removed.

When using the page numbering style X of Y, Microsoft Word does not automatically update the total number of pages (the Y value) when viewing the document in Page Layout format. To update the total number of pages in Page Layout format, the user must select the View Normal option and then the View Page Layout option. The page numbers are always printed correctly.

END OF DOCUMENT