UNIVERSITE DE TECHNOLOGIE DE COMPIÈGNE
Département de Génie Informatique

MOSS 7 - User Interface (Mac)

Jean-Paul Barthès

BP 349 COMPIÈGNE
Tel +33 3 44 23 44 66
Fax +33 3 44 23 44 77

Email: barthes@utc.fr

N206
July 2008

Warning

This document describes the Macintosh user interface to the **MOSS v7.xx** programming environment that uses the PDM4 model for representing knowledge. It includes a description of the browser and of the MOSS Editor. It is an upgrade of the first part of the previous document

`N206 -MOSS 6-Editor`

A current research version of MOSS 7 runs in a MacIntosh Common Lisp environment (MCL 5.1 or 5.2 for OSX) and in an Allegro Common Lisp environment (ACL 8.1 running under Windows XP).

Keywords

Object representation, object-oriented programming environment, ontology formalism, knowledge base, semantic queries.

Revisions

Version	Date	Author	Remarks
1.0	Jul 08	Barthès	Initial issue

MOSS documents

Related documents

- UTC/GI/DI/N196L - PDM4
- UTC/GI/DI/N206L - MOSS 7 : User Interface (Macintosh)
- UTC/GI/DI/N218L - MOSS 7 : User Interface (Windows)
- UTC/GI/DI/N219L - MOSS 7 : Primer
- UTC/GI/DI/N220L - MOSS 7 : Syntax
- UTC/GI/DI/N221L - MOSS 7 : Advanced Programming
- UTC/GI/DI/N222L - MOSS 7 : Query System
- UTC/GI/DI/N223L - MOSS 7 : Kernel Methods
- UTC/GI/DI/N224L - MOSS 7 : Low Level Functions
- UTC/GI/DI/N225L - MOSS 7 : Dialogs
- UTC/GI/DI/N228L - MOSS 7 : Paths to Queries

Readers unfamiliar with MOSS should read first N196 and N219 (Primer), then N220 (Syntax), N218 (User Interface). N223 (Kernel) gives a list of available methods of general use when programming. N222 (Query) presents the query system and gives its syntax. For advanced programming N224 (Low level Functions) describes some useful MOSS functions. N209 (Dialogs) describes the natural language dialog mechanism.

Contents

Warning MOSS has a *versioning mechanism*, i.e. all objects can have different versions in the same environment. The current document is intended for beginning users who want to explore a particular knowledge base that is assumed to be in *version 0* (also referred to as *context 0*).

In this document we use the terms *concept* or *class* to designate a MOSS concept/class object, and *individual* or *instance* to designate a MOSS individual/instance object.

1 Introduction

MOSS applications are usually constructed by using macros or functions. However, MOSS objects can be edited using the MOSS interactive editor. The editing process may be somehow complex due to the rich features of the PDM object model.

The editor features are illustrated on a family example that is both complex enough and easy to understand.

After having launched the Lisp environment, MOSS can be started as a stand alone application by loading the moss-load.lisp file in the MOSS directory. MOSS opens with an interface window shown Fig.1. Throughout the document we refer to this user's interface as the **MOSS window**.

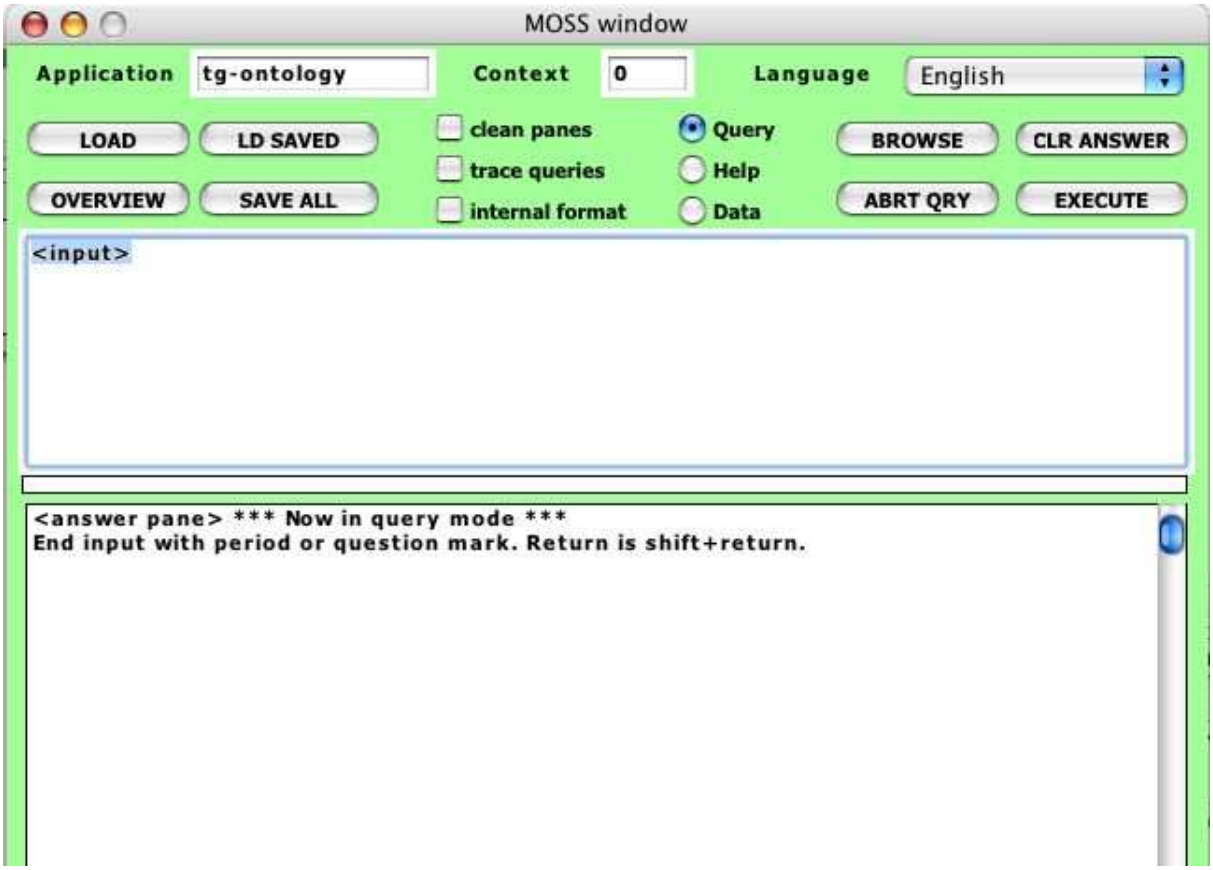

Figure 1: User Interface - MOSS window

In addition to the MOSS window, other more specialized windows can be activated for locating objects, editing objects, or browsing through the knowledge base. Section 2 describes the MOSS command window, Section 3 introduces the concept display, Section 5 the browser, Section **??** the experimental help dalogs. Section **??** the editor.

2 User's Interface - MOSS Window

The MOSS Window has several areas: A top line, a row of buttons underneath, a first display area labeled <input>, a second display area labeled <answer pane>, and in between the two display areas a progress line (that shows in white). This section introduces the various features of the MOSS window taking the view of a naïve user discovering MOSS.

2.1 Loading an Application

One must first initialize the environment by loading an application, i.e. a knowledge base containing a set of concepts, individuals and methods implementing an application. The format of an application file and its location must obey some rules and examples are given in the Appendix. We assume here that we have an application ready to be loaded that models a family and is named "Family."

Loading an application consists in typing the name of the application in the APPLICATION slot and clicking the LOAD button. Here the name of the file is family-m.lisp. If every thing goes well, then some obscure messages are printed in the ANSWER PANE (Fig.2). When typing the file name the case is not important.

Figure 2: Loading the Family knowledge base

2.2 Exploring the Family World

Note The input pane is used to give commands. Ending the input depends on the mode we are in:

- in the query mode, inputs are queries and must obey the query format (refer to the document on MOSS queries). An important point is that there must be as many closing parentheses as opening ones. To execute the command, close enough parentheses and hit the return key.
- in the help mode, a command is a text that ends with a period or a question mark. Hitting the return key simply allows entering more text in continuation lines.

- in the data mode the user may type any Lisp expression, and hit the enter key.

2.2.1 Simple Queries

Now, let us explore the family world. If we do not know what are the concepts of the FAMILY knowledge base, we can ask for them by using the formal request

```
(concept)
```

followed by enter. MOSS displays the available concepts in the ANSWER PANE (Fig.3). If we want to display results on a clean page each time, we check the clean panes box.

In the example, we obtain a list of four concepts: PERSON, COURSE, STUDENT, and ORGANISM. Note that during the access to the object base a progress line (dark blue in between the two large panes) shows the progression (this however is mostly for complex queries that take some time to execute).

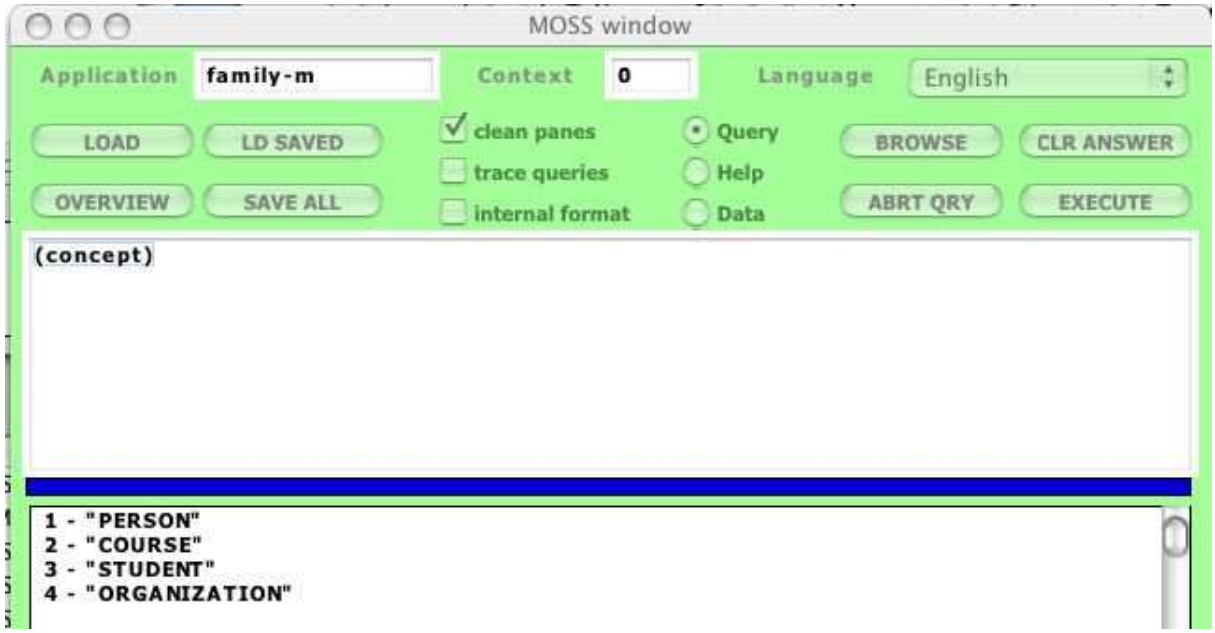

Figure 3: Concepts of the Family knowledge base

If we want now to obtain the list of individuals instances of person, we type

```
(person)
```

into the input pane. The result appears in the ANSWER PANE (Fig.4).

The objects are displayed using a =summary method attached to the concept of PERSON. In the example it prints the first name followed by the family name of a person.

2.2.2 More Complex Queries

More complex queries can be asked, reducing the list of results:

- Fig.5 for example gives the students whose name is "Barthès". The corresponding query is:

  ```
  ("student" ("name" :is "barthes"))
  ```

- Fig.6 gives the list of persons whose name is "Barthès" by using the entry point:

  ```
  "barthes"
  ```

Figure 4: Person individuals in the Family knowledge base. Query: ("person")

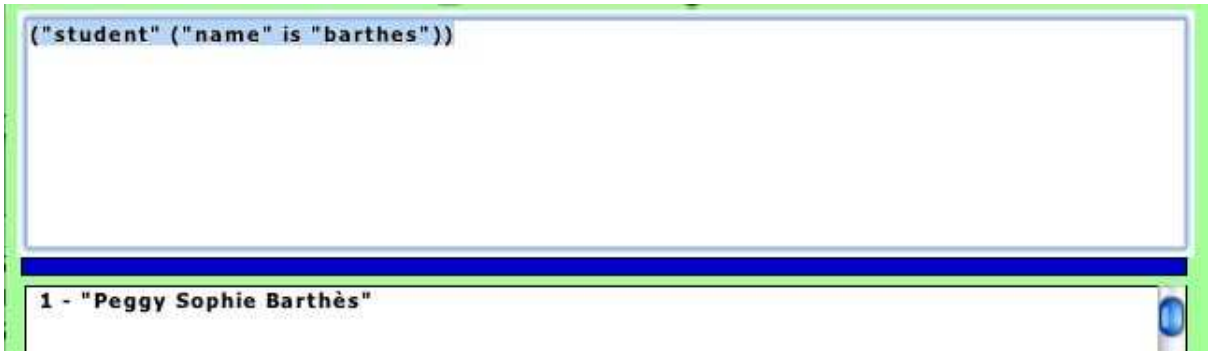

Figure 5: Query: ("student" ("name" is "barthes"))

Note that the result could be obtained by using the query:

```
("person"("name" :is "barthès"))
```

Using the entry point (index) directly returns the list of all objects in the knowledge base for which "barthes" is en entry point. It may include objects that are not persons.

- Fig.7 gives the list of persons whose name is "Barthès" and who have a sister. The corresponding query is:

```
("person" ("name" :is "barthes")
                ("sister" ("person")))
```

- Fig.8 gives the list of persons whose name is "Barthès" and who have at least two brothers. The corresponding query is:

Figure 6: Query: ”barthes”

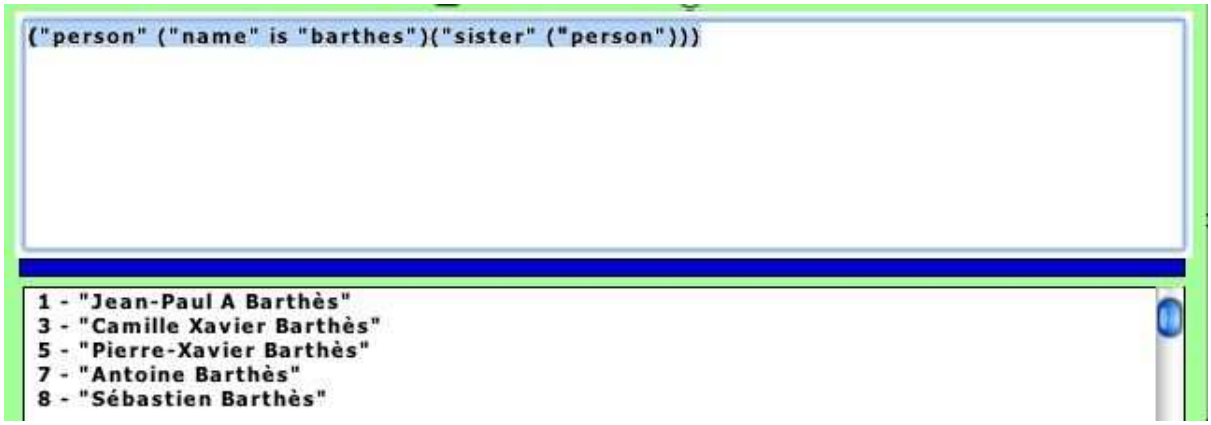

Figure 7: Query: (”person” (”name” is ”barthes”) (”sister” (”person”)))

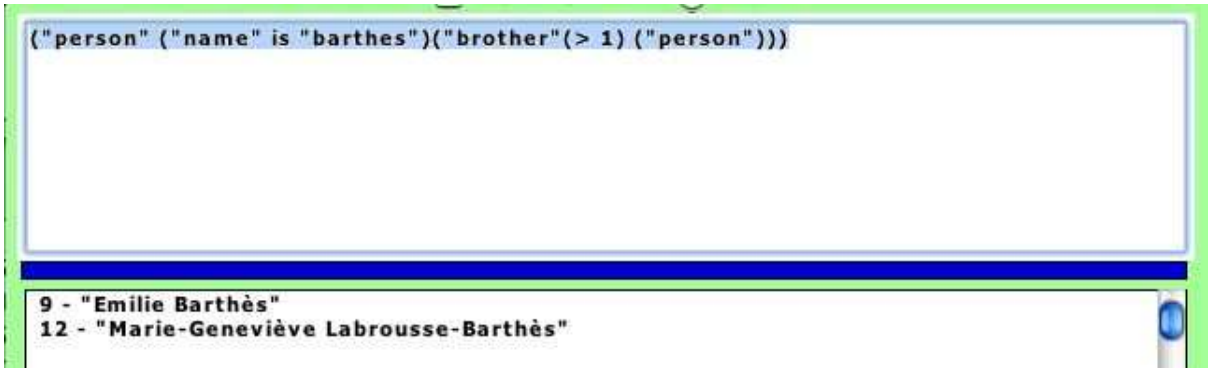

Figure 8: Query: (”person” (”name” is ”barthes”) (”brother” (> 1) (”person”)))

```
("person" ("name" is "barthes")("brother" (> 1) ("person")))
```

The syntax for queries can be quite complex and is detailed in the document ”MOSS Query System.”

Note If a query takes too long, it can be aborted using the ABORT QUERY button. The message "Query aborted at user request" appears in the ANSWER PANE.

2.3 Obtaining More Details about an Object

Objects listed in the answer area can be selected. For example, selecting the Claire Labrousse entry in the list of answers and clicking the BROWSE button makes a browsing window appear as shown Fig.9.

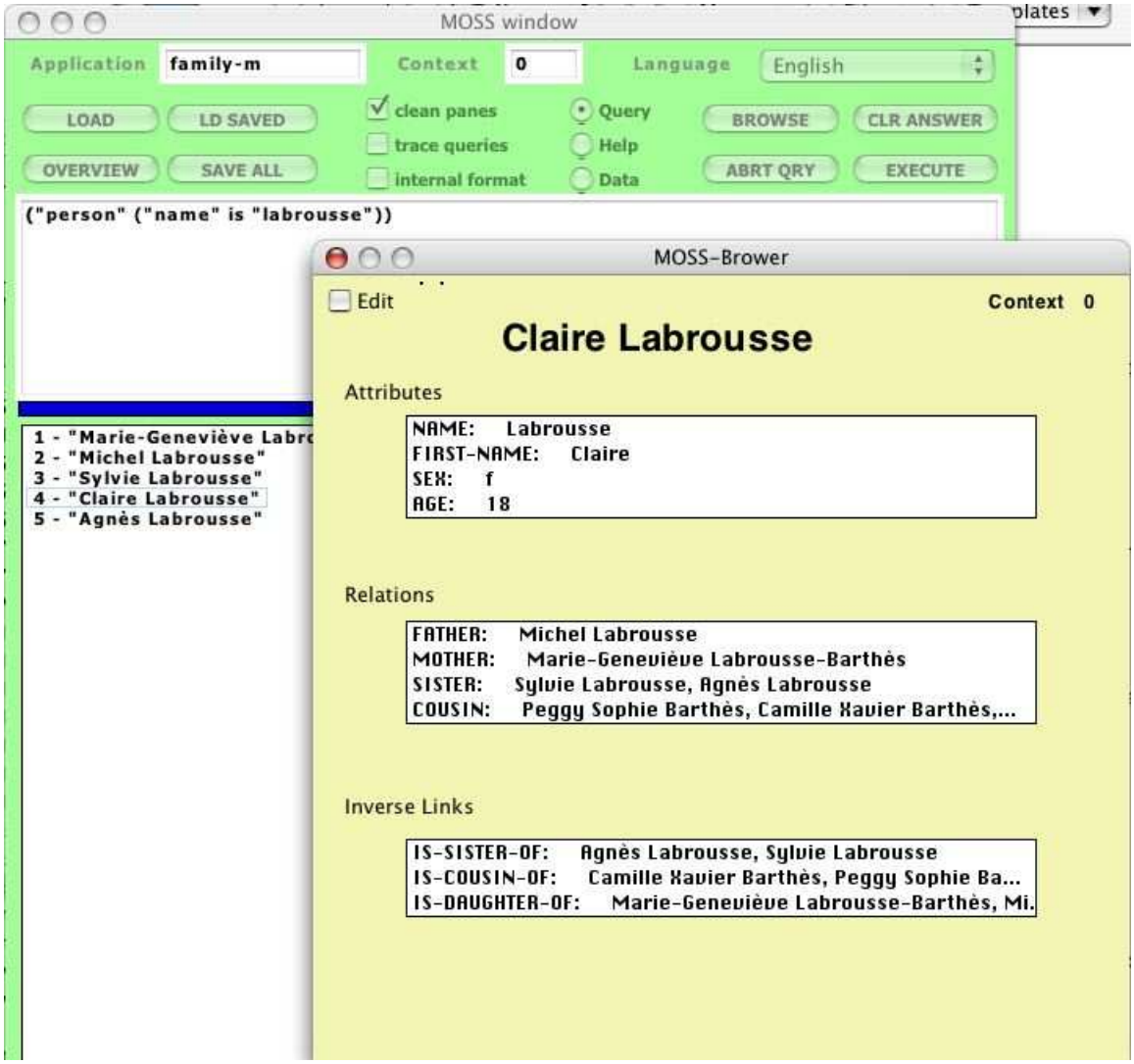

Figure 9: Detailing an object

Then it is possible to browse the database from there (see Section 5), or to edit the object [1] by pushing the Edit button (Fig.10), or to discard the browser window by closing it.

2.4 Browsing Classes and Instances "Ontology" Style

Clicking the OVERVIEW button triggers the CONCEPT DISPLAY, which is described in Section 3.

2.5 Other Buttons

The meaning of the other buttons of the MOSS window is the following:

[1] Note that the object will be edited in memory and not saved back into the original file

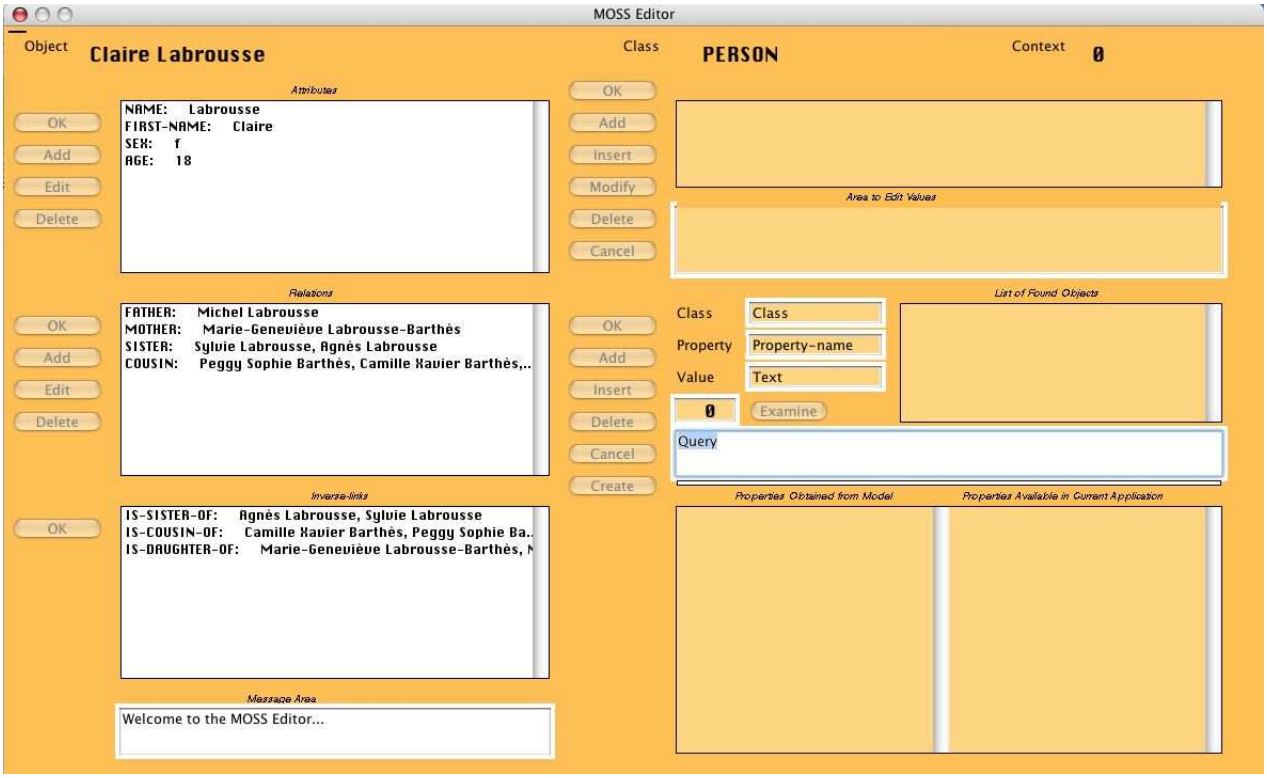

Figure 10: MOSS Editor

- CLEAR ANSWER obvious role is to clear the ANSWER PANE.
- EXECUTE is mostly intended for debugging and executes any Lisp expression typed into the INPUT PANE, displaying the result into the ANSWER PANE.
- SAVE ALL builds a flat file of the application containing all classes, instances, methods, special variables, and functions of the application, stripping the environment of the MOSS system data (so that the system can be updated independently from applications). Using this feature allow saving the objects that have been edited with the MOSS editor, however the link to the original text file is lost. Thus, in practice the changes must be done to the original text file rather than to the structured objects directly in memory, which reduces the role of the editor near to nothing.
- LOAD SAVED loads an application from its saved file.
- BROWSE displays a selected object among the objects returned as the result of a query.
- CONTEXT indicates the context in which objects are displayed (a context is similar to a version). Here the test example only uses context 0. Other contexts are illegal.
- TRACE QUERYING is used mainly for debugging and can be safely ignored.
- LANGUAGE specifies the language to be chosen for displaying a knowledge base (provided it is a multilingual knowledge base).

3 MOSS Concept Display

The Concept Display as shown Fig.11 is used to browse application concepts and individuals. It has three main areas: a left concept area, a central instance area, and a right detail area.

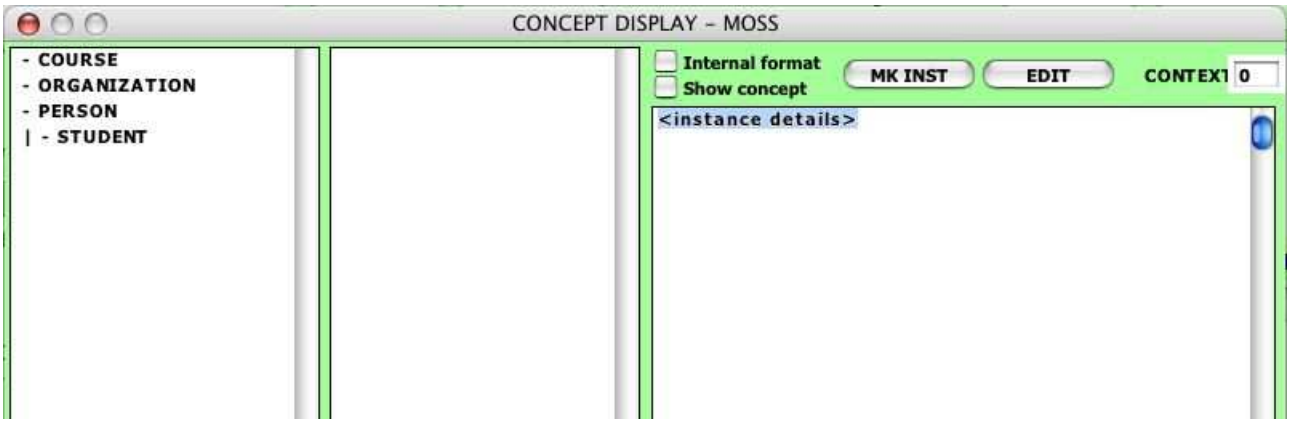

Figure 11: MOSS concept display

3.1 Concept Display Concept Area

The left part contains a tree representation of the application concepts/subconcepts, actually it is a forest. Top level concepts are represented as files, and concepts with subconcepts as folders.

3.2 Concept Display Central Area

When one selects a concept, the list of corresponding individuals is displayed in the central area (Fig.12).

Figure 12: The PERSON concept and corresponding individuals

Individuals corresponding to a concept include individuals corresponding to subconcepts. Individuals are displayed using the =summary method attached to the concept, or else as a list of internal identifiers when the method has not been defined (Fig.13).

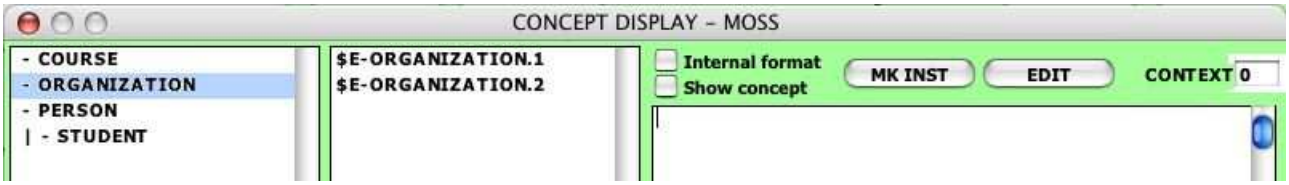

Figure 13: The ORGANISM concept and corresponding individuals (internal identifiers)

3.3 Concept Display Detail Area

When an individual is selected the details are shown on the right hand side detail area (Fig.14).

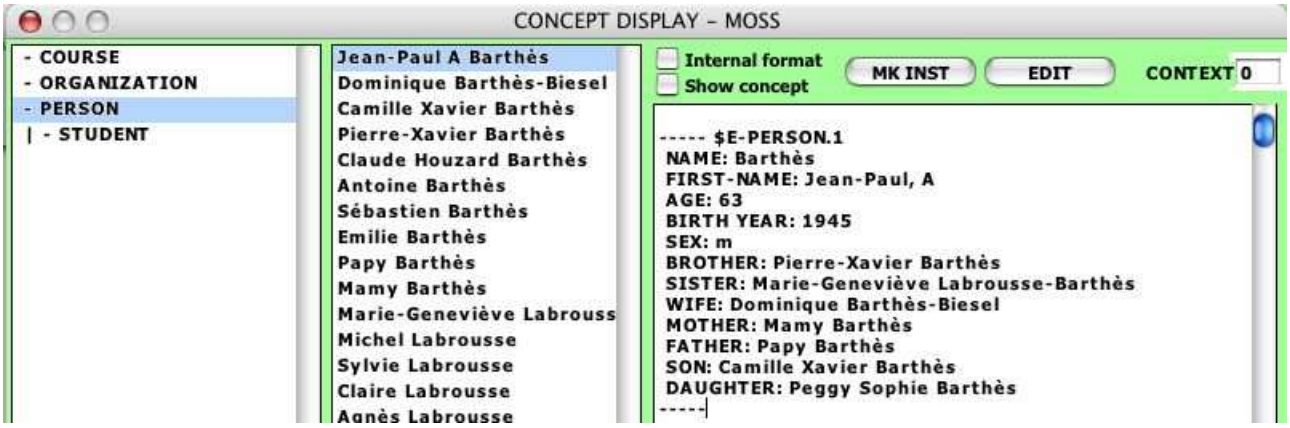

Figure 14: Details of the $E-PERSON.1 object

It is possible to view the internal format of an object by checking the INTERNAL FORMAT checkbox. However, this is not intended for the casual user (Fig.15).

It is also possible to view the concept itself rather than the selected individual by checking the show concept check box (Fig.16).

It is also possible by checking both check boxes to view the internal format of the concept PERSON.

3.4 Editing or Creating Objects

This can be done by clicking the EDIT button. The EDITOR window is then called.

It is possible to create a new individual by clicking the "MK INST" button. The EDITOR opens with a new empty individual of the selected concept.

However, in both cases the changes will occur in memory and will not be reported back to the original text file defining the knowledge base.

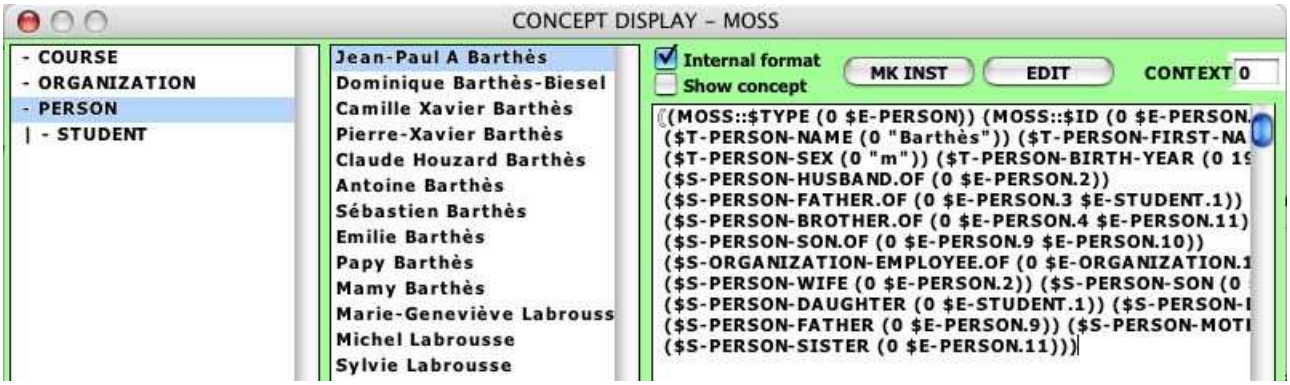

Figure 15: Internal representation of the $E-PERSON.1 object

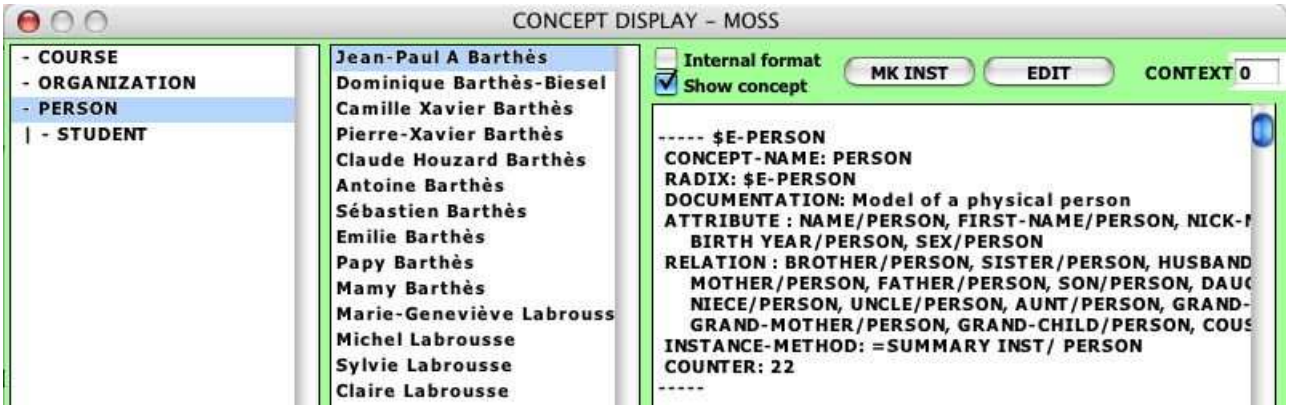

Figure 16: Detail of the PERSON concept

4 The MOSS Browser

The MOSS browser is called when clicking the BROWSE button in the MOSS window (Fig.17).

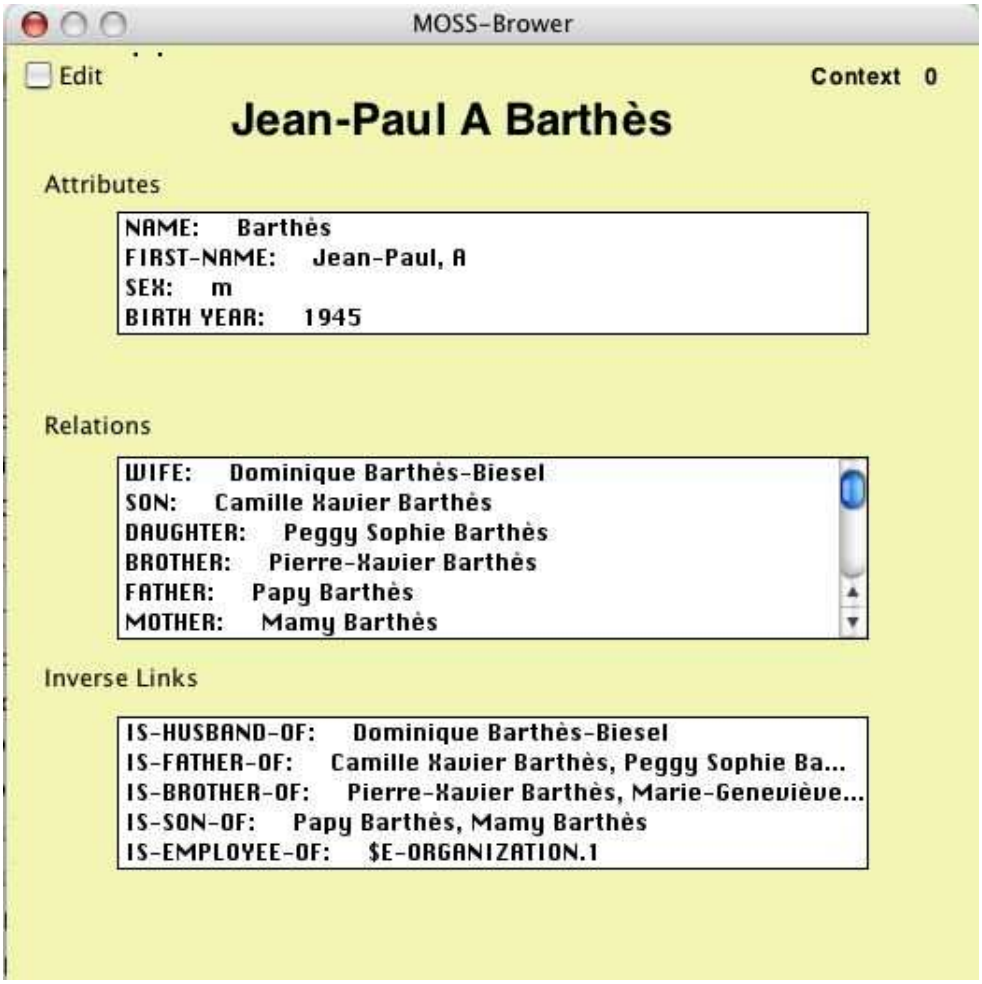

Figure 17: The MOSS browser window

The MOSS browser window displays the title of the object (concept or individual) as the result of applying the =summary method to the object. It contains three areas: one for attributes, one for relations, and one for inverse relations (inverse links). The context in which the object is displayed is shown at the top right corner, and an Edit button is available at the top left corner.

4.1 Browsing the Knowledge Base

MOSS allows navigation as follows:

Clicking on a relation or on an inverse link opens a small temporary window containing information about the linked objects (Fig.18). The first value is the internal key of the object (here $E-PERSON.9).

Double clicking on one of the objects of the temporary window opens a new window with the details of the new object. Windows are tiled.

Coming back to the previous window is done by closing the browser window.

Thus the user can navigate from object to object using direct or inverse links and stop for editing some of the objects.

4.2 Launching the Editor

Opening an Editor for the current browsed object is done by clicking the "Edit" button (Fig.19).

Using the browser is very intuitive.

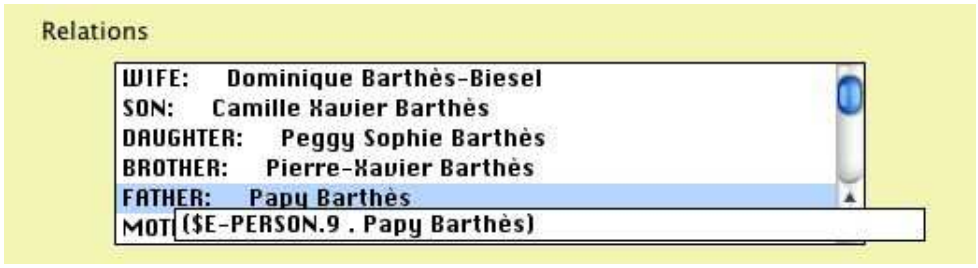

Figure 18: Objects linked to the current object

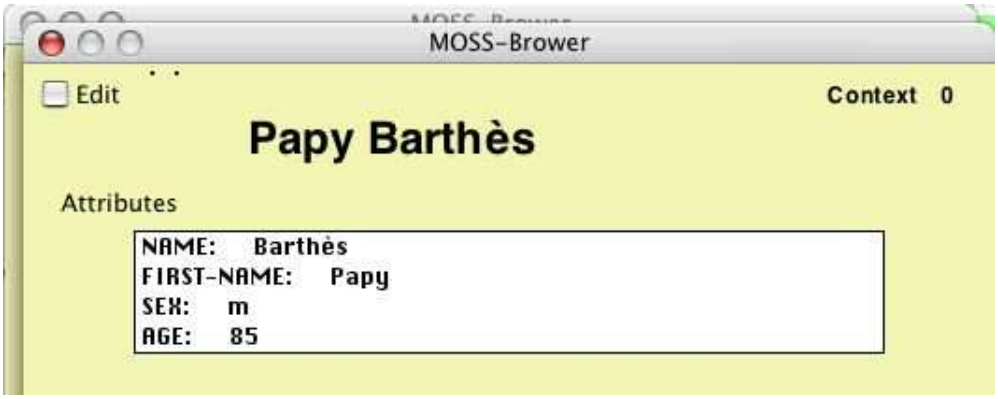

Figure 19: Edit and Exit button

5 The MOSS Editor

(Fig.20) displays the MOSS editor window. It is a fairly complex interface allowing to edit any kind of object. However, the objects being edited are kept in memory and the original text file remains unchanged. This limits the scope of the editor when the objects are loaded from an original text file. It would be useful only when the objects are kept in a persistent object base.

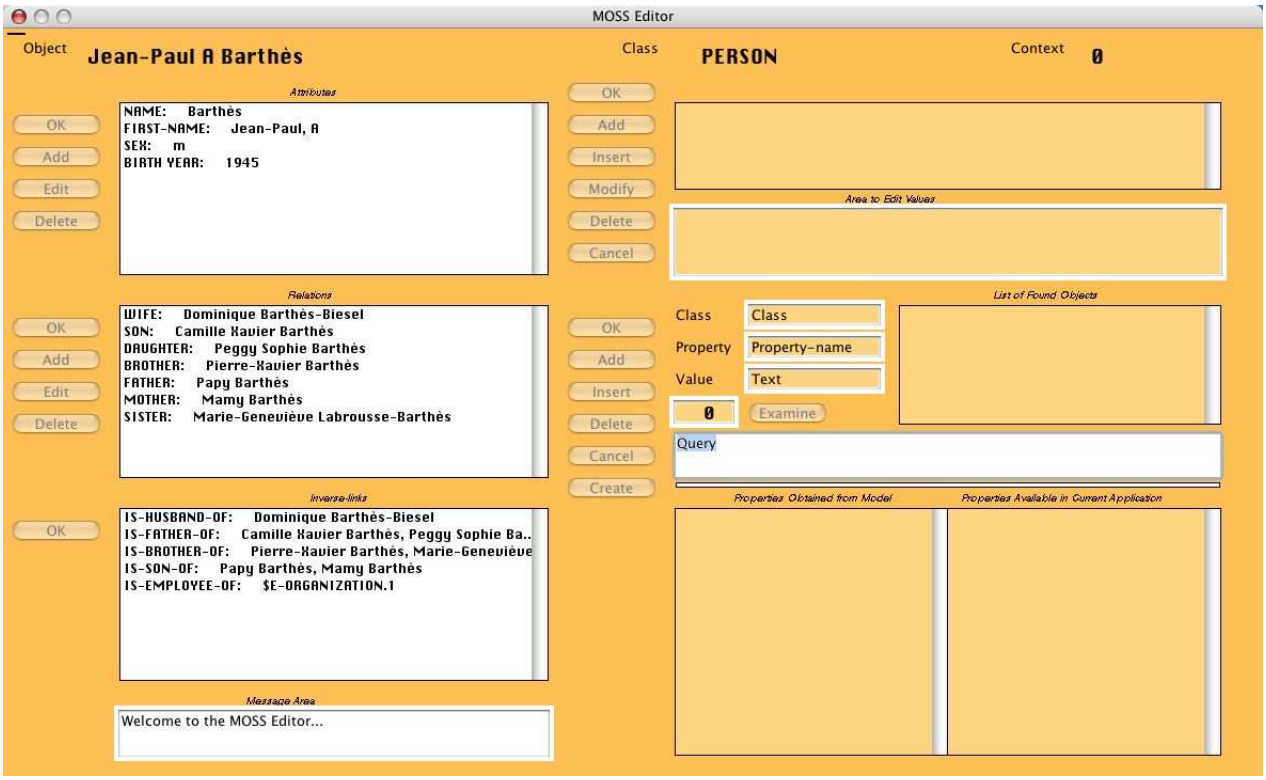

Figure 20: The MOSS editor window

The left part of the window is devoted to the display of the selected object. The top part shows the attributes, the middle part the relations, the bottom part the inverse relations. At the very bottom a special area is used to display (error) messages. The right part of the window is used to edit the object, i.e., add values, delete values, add properties, or link the objects to other objects.

When the window first appears all buttons are dimmed. I.e., nothing can be done. Some buttons are activated when some part of the object is selected in one of the left windows. For example the left top buttons become active when we select the first-name property (Fig.21). Note that the other left windows become dimmed, telling us that we are focusing on editing an attribute

To reset the display to the initial state, we can click the OK button.

The following paragraphs explain what happens for each of the different areas.

5.1 Editing an Attribute

The various possibilities for editing an attribute are:

- Display (by clicking or double-clicking the attribute)
- Add (by clicking the ADD button)
- Edit (by clicking the EDIT button)
- Delete (by clicking the DELETE button)

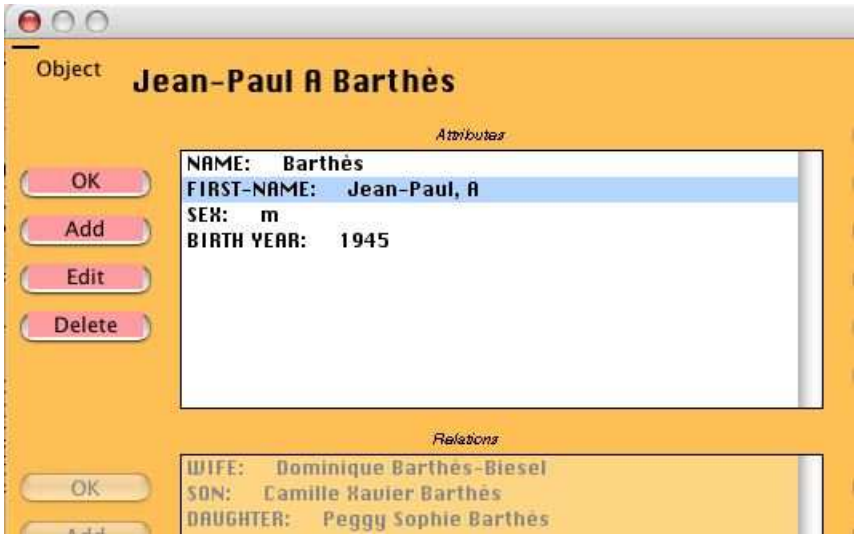

Figure 21: Selecting an attribute

5.1.1 Displaying an Attribute

When an attribute is selected in the attribute window (top left), then the corresponding line is marked as selected and the left buttons are activated.

When the data is multiple, then double-clicking on the data displays a small table with one entry per line (Fig.22). If the data is a single value, nothing happens.

Figure 22: Displaying attribute values

When the function key (Apple key) is pressed while double-clicking on a selected object, then entry-points (e.g. when there are synonyms) are shown for attributes having entry points. For example the entry points associated with the value ?Barthès? of property NAME is the list (BARTHES)(Fig.23).

Figure 23: Displaying entry points

When the attribute has no entry-point, which is the case for attributes FIRST-NAME, SEX, AGE, then double-clicking while holding the command key yields a beep sound and the message

```
No entry point method for this property
```

appears in the message window.

5.1.2 Adding a New Attribute

Whenever the left Add button is active, clicking it will ask MOSS to add a new attribute to the object. Two cases occur, depending on whether the edited object is an instance of a class or an orphan (classless object):

- When the edited object is an instance, the list of attributes recorded at the class level appears in the "*Properties Obtained from Model*" window (left side of the bottom right area - Fig.24), and the list of other attributes present in the system appear in the right window, labeled "*Properties Available in Current Application.*
- When the edited object is an orphan, then only the right list appears.

Thus the user can add an attribute from the model list, e.g. BIRTH-YEAR or NICK-NAME, or decide to add another attribute present in the system from the general attribute list, e.g., LABEL. This means that *MOSS objects can have properties that are not part of their class description*, which is somewhat different from standard object systems.

Note that each property is followed by the class (concept) for which it has been defined.

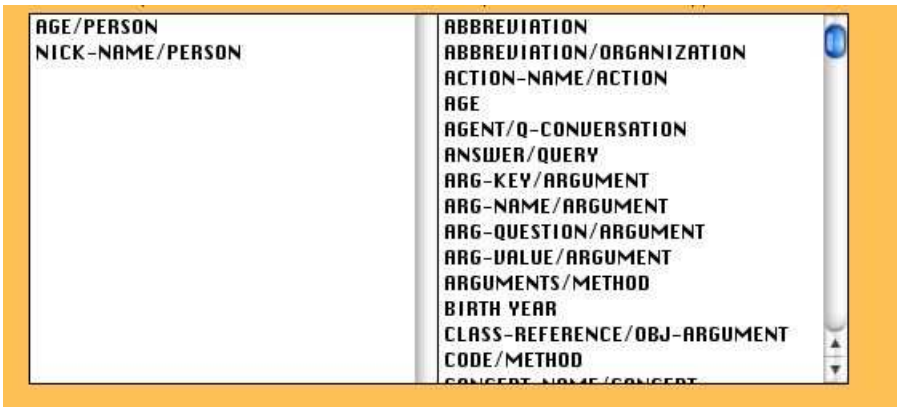

Figure 24: Attributes that can be added

Once the attribute has been selected by clicking on it, then the attribute selection area dims and the attribute editing area (top right part) is activated. The name of the selected attribute shows up on top of the first window, and the cursor is positioned in the second window. The OK button is activated (Fig.25). The following message appears in the message area

```
You can now add a value for the attribute.
```

Once the value attached to the property has been added, then we click the OK button and the entered value appears in the top right window as the first value attached to the NICK-NAME attribute (Fig.26). We are now ready to enter more values. Remember MOSS attributes are multi-valued and the values are ordered for the attributes.

Note that at the same time the left window displaying the object attributes is updated as well.

When no more values are needed, then we can return to doing other things by clicking into the top left window displaying the object attributes. All areas but the attribute area are dimmed.

Figure 25: Ready to add an attribute

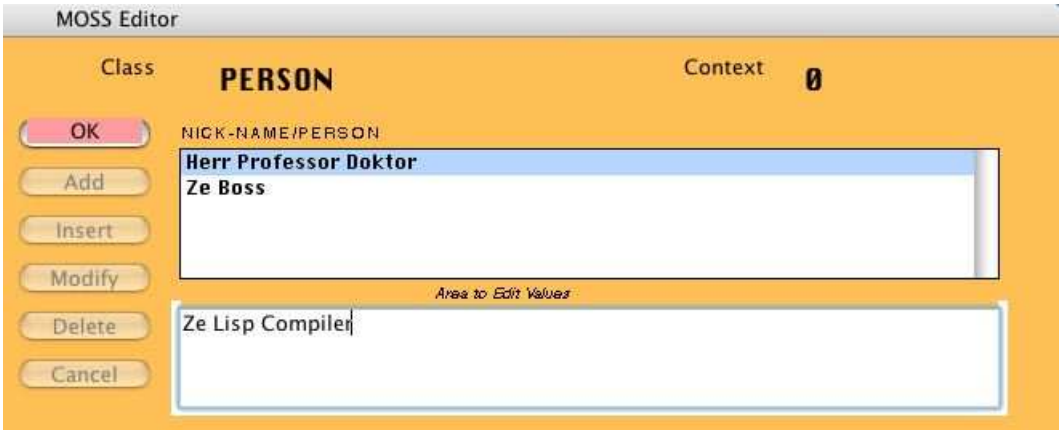

Figure 26: Adding a value for an attribute

Note Clicking the OK button while no value has been typed in adds an empty value. A warning message appears in the bottom right window entitled *Message Area*:

```
You just added an empty value.
```

An indication of an empty value is given by the presence of a comma in the attribute list in the *Attributes* area and as a blank line in the list of values (top right window). An empty value can be selected in the list of values in the right top window, and deleted.

5.1.3 Modifying the Values Attached to an Attribute (Right Top Area)

Modifying the values attached to an attribute is done readily. For example, if we want to modify the recently created NICK-NAME attribute, then we select the corresponding line in the left attribute window and the same situation as in Fig.26 occurs with the exception that the OK button is not active. However when we click into a blank area of the detailed window then the OK button becomes active and the following message appears in the message area:

```
No selected value. You can only add more values.
```

We can then add values as done previously.

When we select an existing value however, then the Add Insert Delete Modify buttons become active, with the following effects:

- **Add**. Clicking the Add button allows us to enter a value that will appear right *after* the selected value or at the end of the list of nothing is selected.

- **Insert**. Clicking the Insert button allows us to enter a value that will appear right *before* the selected value.
- **Delete**. Clicking the Delete button deletes the selected attribute. No confirmation is asked.
- **Modify**. Clicking the Modify button allows to edit the selected value, and then replace the old one at the same position.

All the changes are also posted in the attribute display area in the top left hand-side of the edit window. At any time, clicking in this window terminates the editing.

5.1.4 Deleting a Given Attribute

Clicking the Delete button remove the selected value and all attached values from the object. No confirmation is asked.

5.2 Re-enabling the Other Areas

After the attribute edition is finished, we are back to a situation like Fig.21 with all areas dimmed except for the attribute area. Clicking the OK button re-enables the other areas.

5.3 Editing a Relation

The various possibilities for editing an attribute are:

- Display (by clicking or double-clicking on the attribute)
- Add (by clicking the ADD button)
- Edit (by clicking the EDIT button)
- Delete (by clicking the DELETE button)

5.3.1 Displaying a Relation

When a relation is selected in the relation window (center left), then the corresponding line is marked as selected and the left buttons are activated.

Double-clicking on the data displays a local table with one entry per line (Fig.27). Clicking on one of the entries of the table opens a new editor window with the corresponding object to edit.

Figure 27: Displaying details of the COUSIN relation

5.3.2 Adding a New Relation

Whenever the left Add button is active, clicking it will ask MOSS to add a new relation to the object. Two cases occur, depending on whether the edited object is an instance of a class or an orphan (classless object):

- When the edited object is an instance, the list of relations recorded at the class level appears in the *Properties Obtained from Model* window (left side of the bottom right area - Fig.28), and the list of other relations present in the system appear in the right window, labeled *Properties Available in Current Application.*
- When the edited object is an orphan, then only the right list appears.

Thus, the user can add a relation from the model list, e.g. COUSIN or NEPHEW, or decide to add another relation present in the system from the general attribute list, e.g., COUNTER. This means that *MOSS objects can have properties that are not part of their class description*, which is somewhat different from standard object systems.

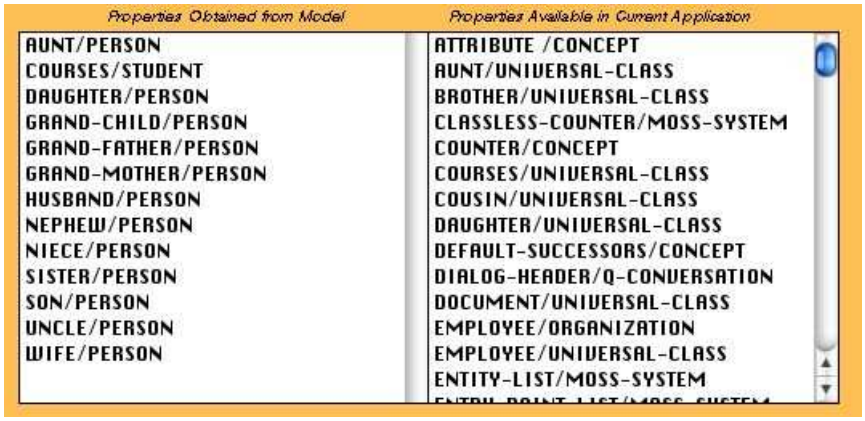

Figure 28: List of relations that can be added to a person

Once the relation has been selected by clicking on it, then the relation selection area dims and the relation editing area (middle right part) is activated. The name of the selected relation shows up on top of the first window, and the cursor is positioned in the second window. The Cancel button is activated (Fig.29). The following message appears in the message area:

```
You can now search for an object to add, or create a new one.
```

In the top right hand-side part of the editor window the name of the concerned relation appears (here UNCLE). The center part of the window is used to locate objects to connect to the edited object. When objects exist they must be found. When they do not exist they can be created.

The bottom part of the area can be used to input queries for locating objects.

The right center part of the editor window helps to locate objects.

For example, if we know an entry point (here the family name "canac"), we can use it to locate objects, as shown Fig.30.

We can add an object by selecting it (Fig.30) and clicking the Add button. The object appears in the top right area and the relation and new object appear in the left list of relations.

Clicking the Cancel button deselects all selected objects.

If we look for a person, we can specify the class name followed by Return or Enter, as shown Fig.31. The count figure gives the name of available persons. The Examine button allows to list them as shown Fig.32.

Objects can then be selected from the list and added using the add button. Multiple selections are permitted. Clicking Cancel deselects all selected objects.

Double-clicking an object of the list call the browser. It is a convenient way to have more information on the object.

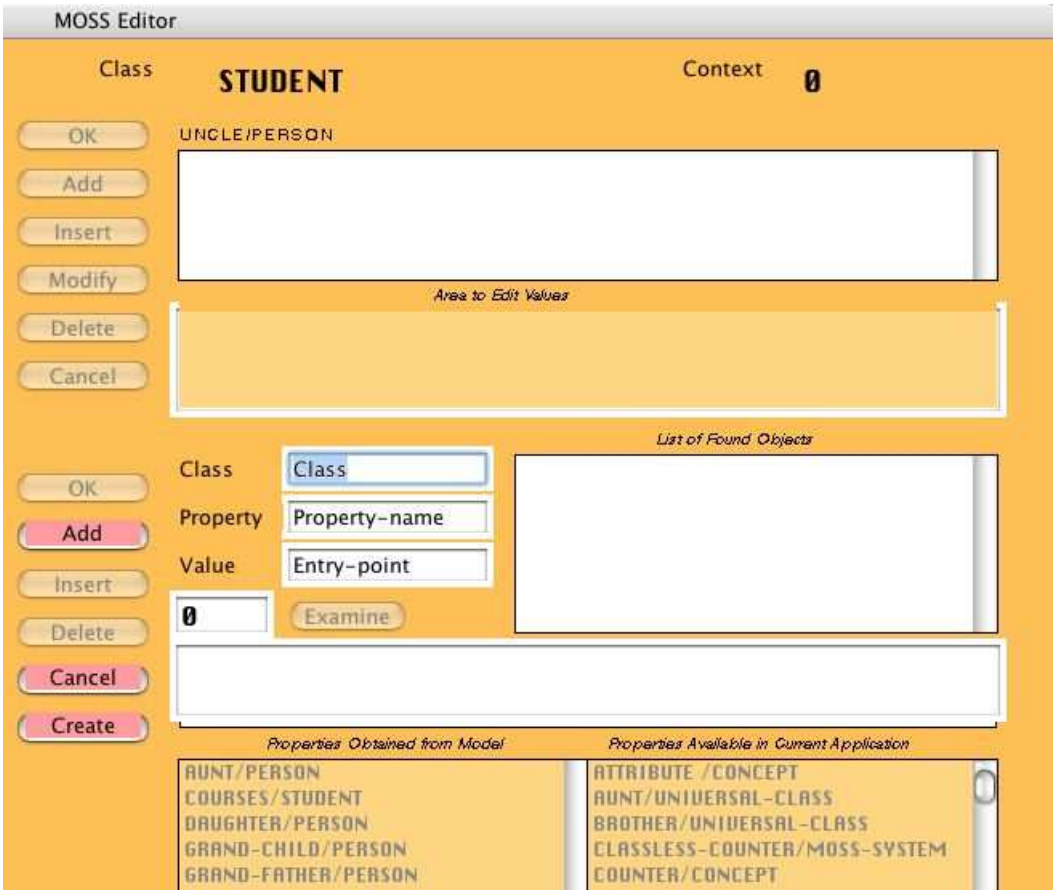

Figure 29: Adding a new relation

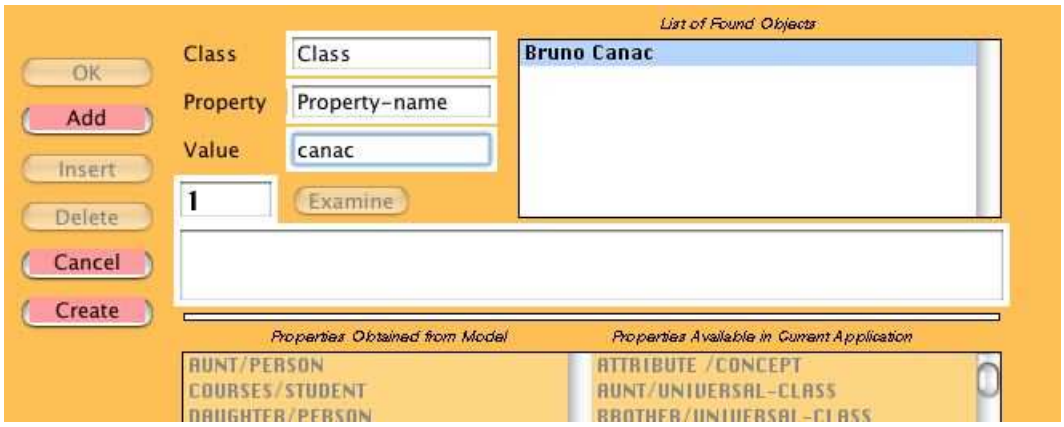

Figure 30: Locating an object to link from its entry point (here CANAC in the value slot)

Objects can also be located by using the query area, which works like in the window area. Executing the query is done by clicking the enter key. The number of candidates to be examined in the query is given in the counter. Viewing the objects is done by pushing the Examine button (Fig.33).

Creating a Missing Object to Add

It is possible that the object that needs to be added does not exist. It is then possible to create it by choosing a class, or selecting an object in the *List of Found Objects* and clicking the Create button. If the selected object is an instance or an orphan, then a message appears, and if confirmed (OK button), then the object is cloned

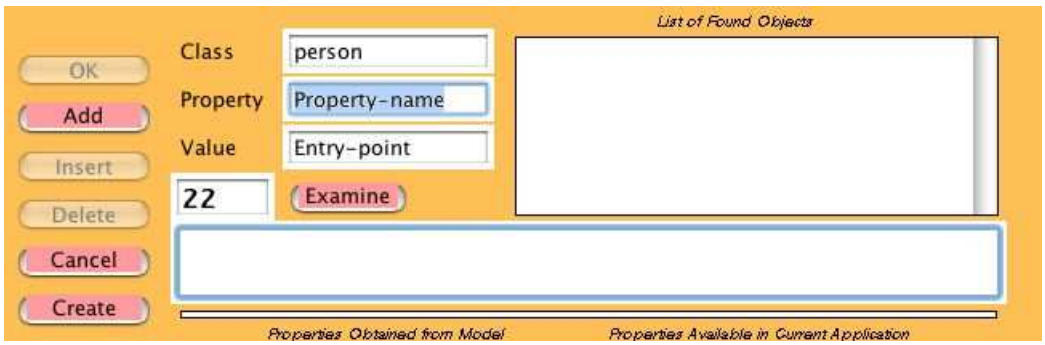

Figure 31: Locating objects of a given class (here PERSON)

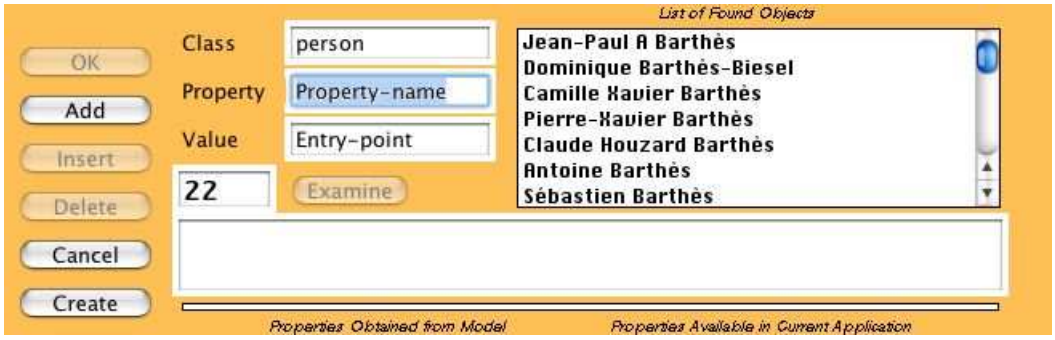

Figure 32: Displaying the objects of a given class (here PERSON)

Figure 33: Result of executing a query for locating objects

```
Selected object is not a class, do you want to clone it?
Yes, push OK; no, push CANCEL.
```

5.3.3 Editing a Relation

When the proper line of the left display window is selected and the Edit left button is clicked, then the linked objects appear in the top right window and the name of the relation is given to this window. The editing process proceeds as described in the previous paragraph.

For example clicking on the COUSIN line of the relations brings up the window shown Fig.34. The linked objects appear in the top right editing pane.

Selecting a cousin on the top right pane activates the editing buttons whose behavior is similar than in the case of attributes, except for the CANCEL and CREATE buttons already explained.

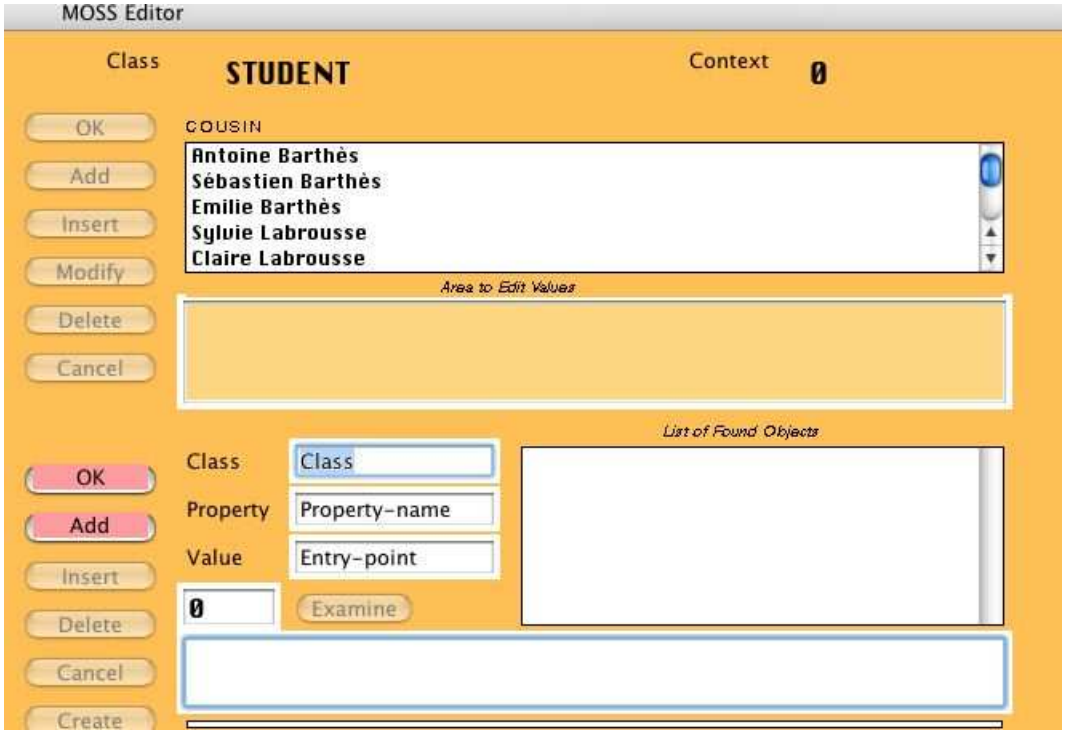

Figure 34: Editing the COUSIN relation

5.3.4 Deleting a Relation

To delete a relation, one must first select it on the left pane displaying relations and click the Delete button. No confirmation is asked.

5.4 Following Inverse Links

When the inverse link area is activated it is possible to display the details of the objects related to the object being edited by double-clicking on it (Fig.35).

Figure 35: Viewing objects related by inverse relations

Clicking on one of the displayed objects calls a new editing window with the selected object to be edited.