The Ministry of Education and Science of the Russian Federation
Samara State Aerospace University
(National Research University)

DESIGN PROBLEM SOLVING USING ONTOLOGICAL SYSTEMS

Electronic Laboratory Guideline

SAMARA

2011

Compilers: **Borgest Nikolay M.,**
Simonova Elena V.,
Shustova Dina V.

Translated by: **Borgest Diana N.**

Design Problem Solving Using Ontological Systems = Решение проектных задач с помощью онтологических систем [Electronic resource] : Electronic Laboratory Guideline / The Ministry of Education and Science of the Russian Federation, Samara State Aerospace University; Compilers N. M. Borgest, E. V. Simonova, D. V. Shustova. - Electronic text and graphic data (4 Mb). - Samara, 2011. - 1 CD-ROM.

The guidelines for laboratory works on the subject “Ontology of Designing” are a part of postgraduate programmes which were developed based on using new educational technologies, resources and distance-learning systems for the Masters programme «Designing, construction and CALS-technologies in Aeronautical Engineering » for education direction 160100.68 «Aeronautical Engineering».

Prepared by the Department of Aeronautical Engineering SSAU.

Contents

INTRODUCTION

The “Ontology of Designing” laboratory works consist of two parts. In the first part standard design problems of choosing and taking decisions are solved (Laboratory works 1, 2), and in the second part an ontological analysis of domain on the example of an aircraft is done (Laboratory works 3). Magenta (Magenta Corporation Limited) and Protégé (Stanford University, USA) are used as ontology editors.

The first and third labs are done using the ontology editor Protégé. It supports two main ontology modelling methods via Protégé – Frames (Protégé 3.3 (Laboratory works 1)) and Protégé – OWL (Protégé 4.1. beta (Laboratory works 3)). Ontologies developed in Protégé can be exported in many different formats including RDF, OWL and XML (http://protege.stanford.edu/)

Technical System Design is a complex knowledge-intensive process that is preceded by the need analysis stage. The need, which the transport systems realise in the laboratory works, is considered as goods transportation. One of the main tasks of designing is to provide the conditions of high quality transportation.

Ontological analysis of the domain is an important aspect of designing which aims to organise and classify entities, set connections, parameters and characteristics, and design a thesaurus. In addition to organising domain knowledge the ontological analysis also helps to improve the quality of design works.

The first laboratory work sets a task of goods transportation from one place to another. Samara and Moscow are the points of departure and destination. There are several ways to solve this task according to the following criteria:

- Loading capacity of the vehicle;
- Load volume;
- Cost and time of transportation, and others.

In accordance with the initial conditions and the given parameters, we look for the most efficient way to transport the goods via:

- Air transport;
- Motor transport;
- Rail transport;
- River transport.

Ontological analysis of the entities, attributes and relations between them allows finding an optimal way to deliver the goods.

When an efficient way of transportation is chosen new details are considered. Ontology is complemented with real brands of aircrafts. In this case the performance specification becomes more detailed with reference to the specifics of a chosen transport.

In the second laboratory work, based on the characteristics of aircraft prototypes (this information is stored in the database), wing loading and thrust-to-weight ratio plane parameters are chosen. The multi-agent system Magenta (www.magenta-technology.com) is used as a working tool in this lab.

Lab 3 is one of the most important laboratory works for the postgraduates. In the methodological guidelines there is an example of the ontological analysis of the domain

“Aircraft” using the editor **Protégé 4.1. beta**. This version allows building OWL ontologies for the Semantic Web. The formal semantics of OWL makes it possible to obtain logical consequences, i.e. facts that are not presented directly in the ontology but can be concluded by the means of semantics.

The **aim** of the laboratory works is to learn the method of ontological analysis based on the design problem solving examples of choosing transport systems, their characteristics and organising the domain Aircraft.

The **tasks** of the practical part:

- Become familiar with the contemporary ontology editors Protégé (3.3 and 4.1 beta versions) and Magenta;
- Gain skills in design problem solving using ontological systems.

The laboratory works are done simultaneously with the assimilation of the textbook material [1, 2] and reading the scientific articles [3] prepared for the course.

LABORATORY WORK 1.

1.1. Part 1. Choosing a transport

INTRODUCTION

The artificial intelligence literature has many different definitions of "ontology" (including [1-3]). Generally, ontology as a way of knowledge representation of the outside world is a precise specification of a domain. Ontology includes a vocabulary for representing and knowledge exchange about the domain as well as a set of relations between the terms in this vocabulary. The use of ontology as one of the ways of organising terms and concepts is a means of describing the knowledge that combines with the other well-known knowledge representation models.

Ontology forms a knowledge base together with a set of individual instances of classes. Ontology always reflects an analyst's view, i.e. it is always subjective; therefore, the only correct way to create it does not exist. Despite this, it is possible to identify some basic principles that underpin the creation of an ontology.

- Clarity: ontology should effectively transmit the meaning of the terms; its definitions should be objective, and for this reason a clearly fixed formalism should be used.
- Coherence: all the definitions should be logically consistent.
- Extendibility: it is necessary to design an ontology ensuring that its vocabularies of terms can be extended by using already existing concepts.
- Encoding bias: the conceptualisation of the ontology must be specified at the level of representation and be independent of symbol-level encoding;
- Minimal ontological commitment: an ontology should require the minimal ontological commitment sufficient to support the intended knowledge-sharing activities.

Protégé is one of the most popular ontology editors created at Stanford University (USA). In Protégé all the domain notions are divided into classes, subclasses, instances.

The process of ontology building using Protégé includes:

1. Defining ontology classes;
2. Organising the classes into a hierarchy;
3. Defining slots and their values;
4. Filling the slot values for instances of classes.

1.1.1. GOALS AND OBJECTIVES OF LABORATORY WORK 1

Goals:

1. Get familiar with the ontology editor Protégé;
2. Learn how to develop an ontology.

Objectives:

1. Develop a transportation ontology;
2. Fill the created ontology with instances;
3. Choose a vehicle for goods transportation that meets the given requirements.

The following requirements (vehicle performance) are considered: loading capacity, cargo space, time and cost of transportation.

1.1.2. Building an ontology

1.1.2.1. Project development

In Protégé the work starts with creating a new project or selecting an existing one.

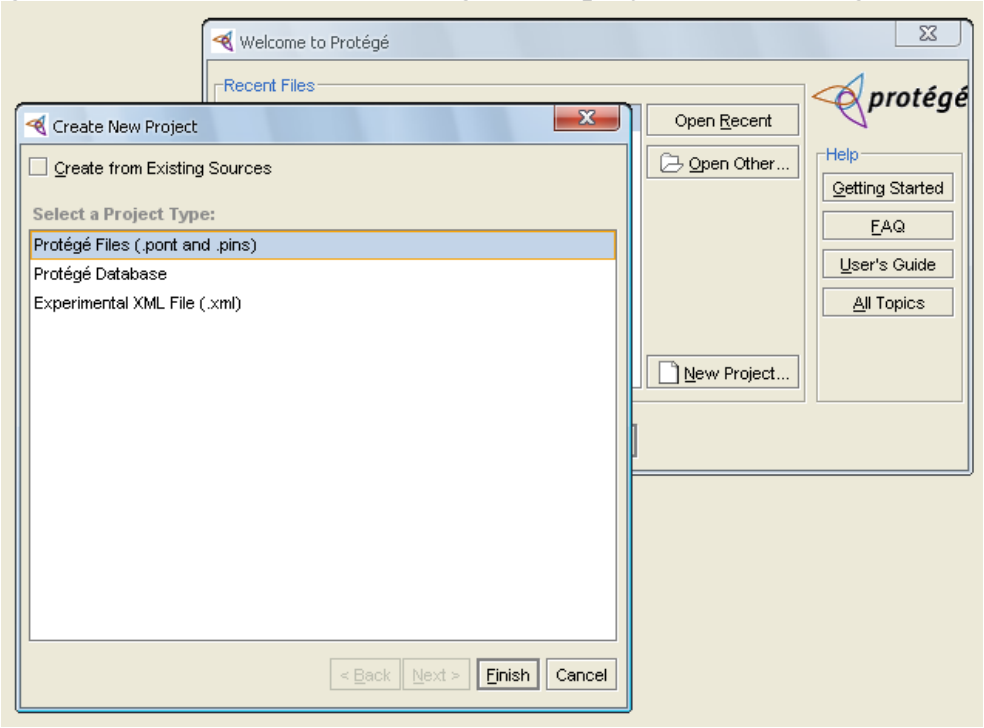

Figure 1.1. The Welcome Window

To create a new project, press the button "New Project…" on the welcome window (figure 1.1). The dialogue box "Create New Project" will appear. It allows selecting a project type. If it is not required to create files in a special format, then click "Finish", the Protégé Files default format will be selected (.pont and .pins). Note, that it is not necessary to mark the checkbox "Create from Existing Sources". This opens a window allowing viewing the classes (figure 1.2). At this stage there are just default Protégé system classes THING and SYSTEM-CLASS.

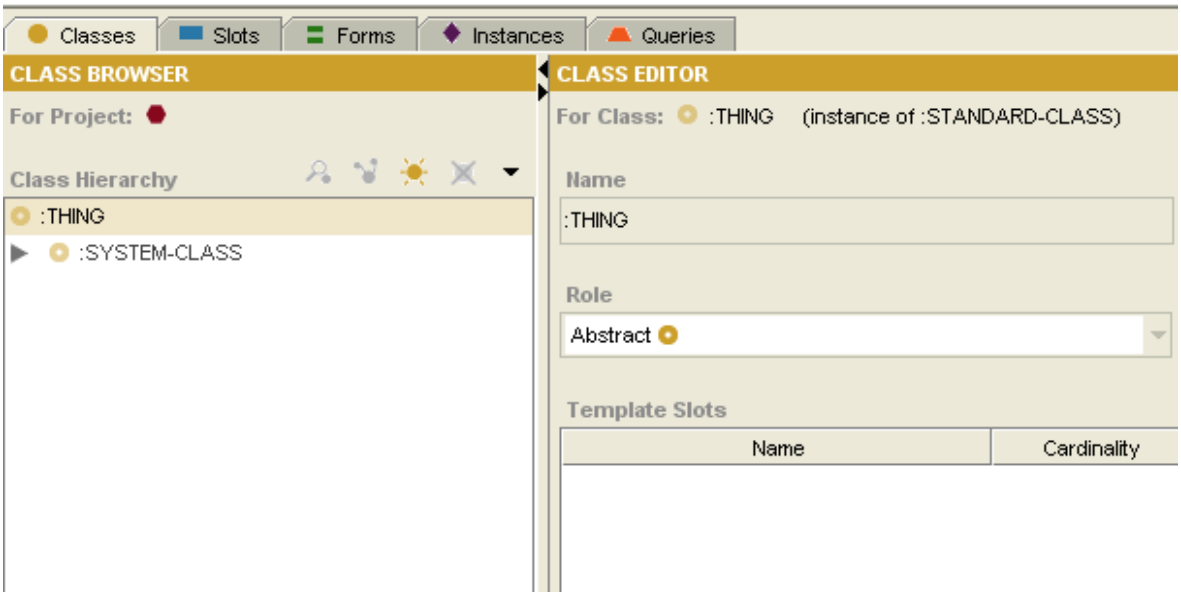

Figure 1.2 Class window

1.1.2.2. Saving a project

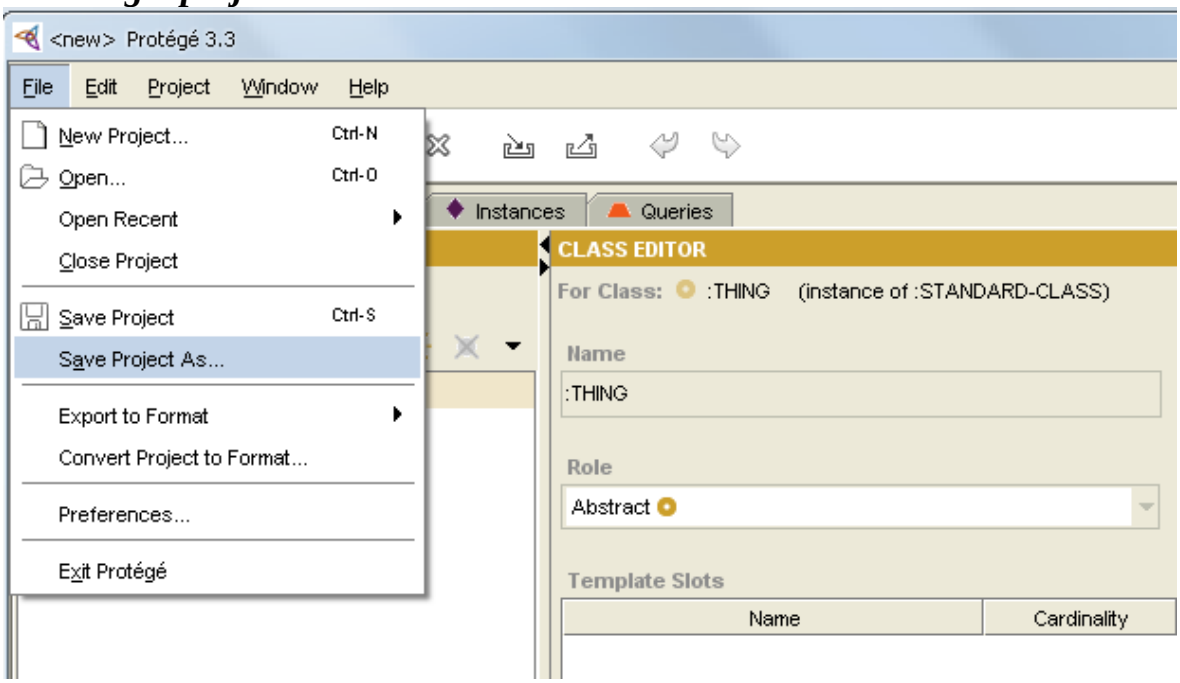

Figure 1.3 Save Project window

A project can be saved in several ways:

1. By selecting "Save Project As…" from the File menu (figure 1.3)
2. By pressing the saving button on the toolbar;
3. With the key sequence Ctrl+S

This will bring up a window for selecting a file name of the project (figure 1.4)

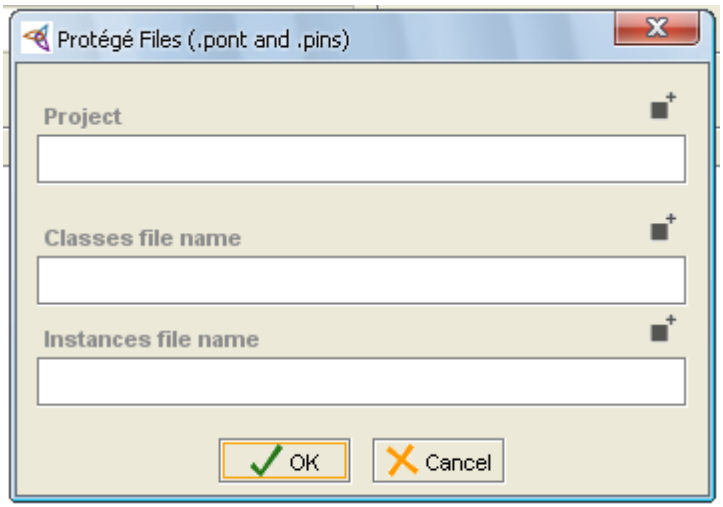

Figure 1.4. Selecting file name window

To specify a location where the project will be saved, click the button which is on the left from the Project sign. In the new dialogue box navigate to the correct path in the file system or create a catalogue where the project data will be stored. Open the new folder. Type a name to the project "LR1_varX", where X is your variant. Clicking the Select button returns you to Save project window. Click OK, and the project will be saved.

1.1.2.3. Creating classes

The Protégé main window consists of tabs that represent different aspects of knowledge models. When creating a new project the tab Classes is the most important one. The classes correspond to objects or type of objects in a certain domain. In the lab, classes will include the types of vehicles viz. air, rail, motor and river transport.

In Protégé classes are presented in a class hierarchy that is located in the Class browser, on the left of the Classes tab. The properties of the classes that are currently selected in the browser will be shown on the right in the fields Name, Template Slots (figure 1.2).

1.1.2.4. Creating the "Air transport" class

To create a new class select the class Thing in the Class browser. All newly created classes should be at a lower level of this system class. The System class is used to determine structures of Protégé various forms.

Press the button to create a class in the upper corner of the Class browser. A new class with a standard name based on the name of the project will be created. In the Name box, type "Авиатранспорт" (Air transport), the name of the class, using the keyboard and press Enter. You can see that after creating, the name of the new class in the Class browser will be highlighted; it is to indicate that this class is currently selected (figure 1.5).

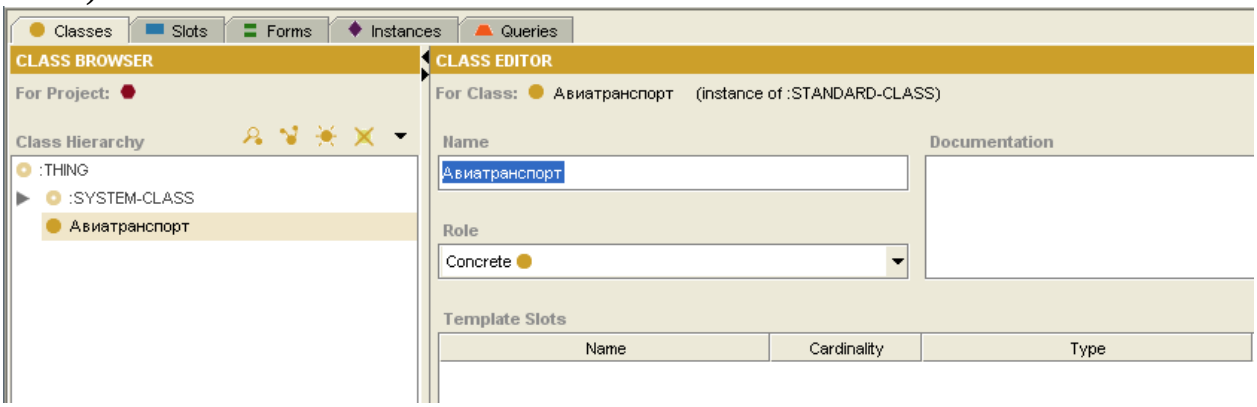

Figure 1.5. Class editor window

In Protégé the names of classes are set to write with capital letters, and the names of slots with lowercase letters. This rule helps to distinguish the classes from the slots in the newly created ontology.

1.1.2.5. Creating the "Rail transport", "Motor transport", and "River transport" classes

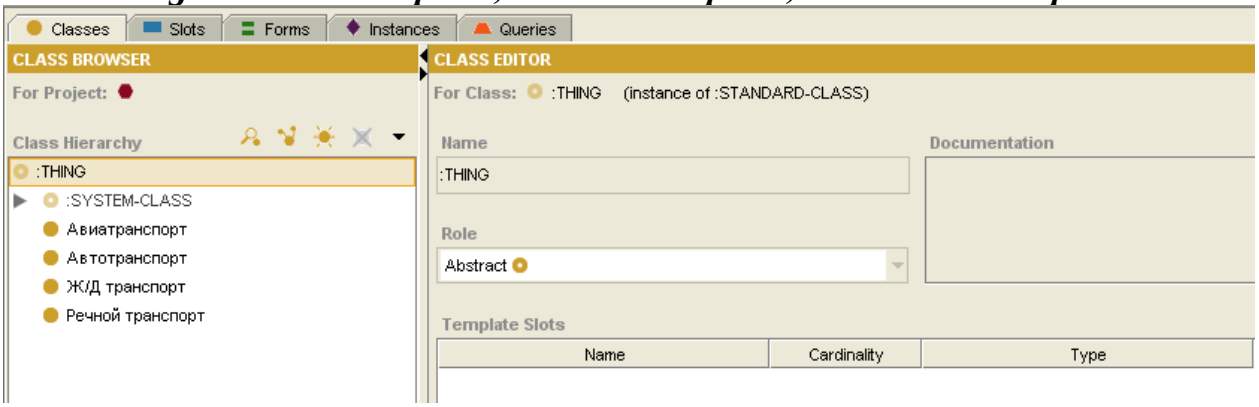

Figure 1.6. Created classes browser window

When new classes are created it is required to select the class Thing in the Class browser, so that the created classes are on the same level. Similarly to the previous class, create the “Железнодорожный транспорт” (Rail transport), “Автомобильный транспорт” (Motor transport), and “Речной транспорт” (River transport) classes (figure 1.6).

1.1.2.6. Modifying the tree hierarchy

Together air transport, motor transport, rail transport and river transport are the transport system. To represent it in the ontology structure, a new class, “Транспортная система” (Transport system), is to be created. This class will be on the same level as the previous classes (figure 1.7).

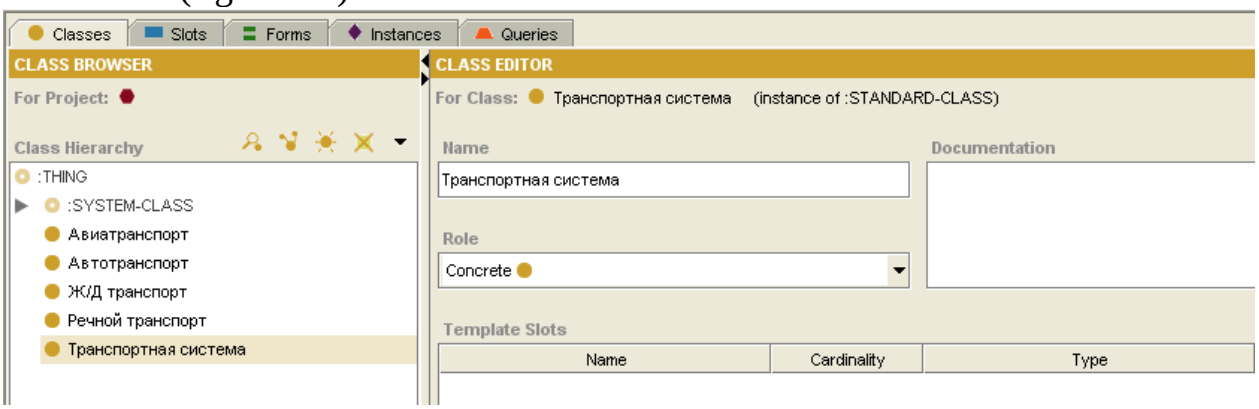

Figure 1.7. Creating the “Транспортная система” (Transport system) class

In fact, the “Авиатранспорт” (Air transport), “Автомобильный транспорт” (Motor transport), “Ж/Д транспорт” (Rail transport), and “Речной транспорт” (River transport) classes should be at a lower level than the “Транспортная система” (Transport system) class. This example is given to learn how to edit the structure of the tree hierarchy.

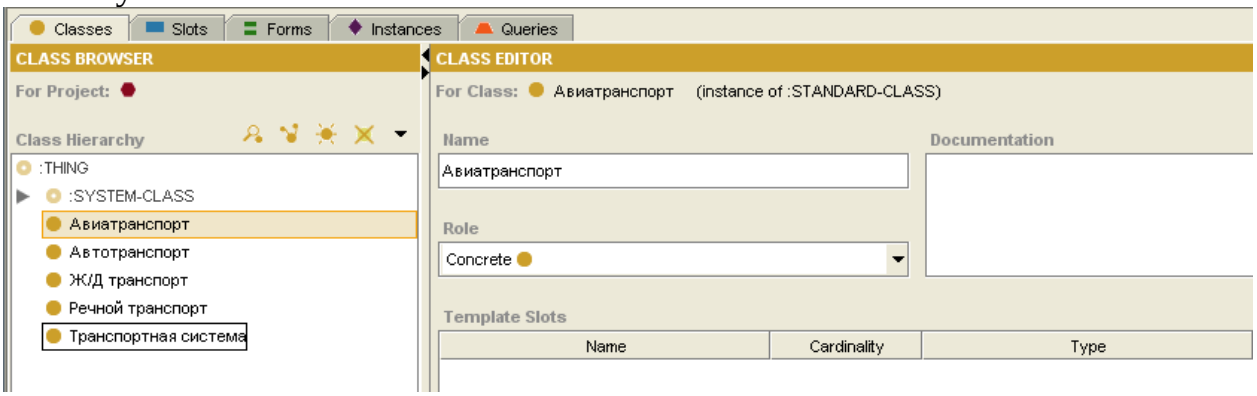

Figure 1.8. Adding the “Авиатранспорт” (Air transport) class to the “Транспортная система” (Transport system) class

To change the inheritance in the tree hierarchy, select the “Авиатранспорт” (Air transport) class and drag it to the “Транспортная система” (Transport system) class holding the left mouse button (figure 1.8). Do the same for the other classes. Thus, the “Транспортная система” (Transport system) class will include the “Авиатранспорт” (Air transport), “Автомобильный транспорт” (Motor transport), “Ж/Д транспорт” (Rail transport), and “Речной транспорт” (River transport) subclasses (figure 1.9).

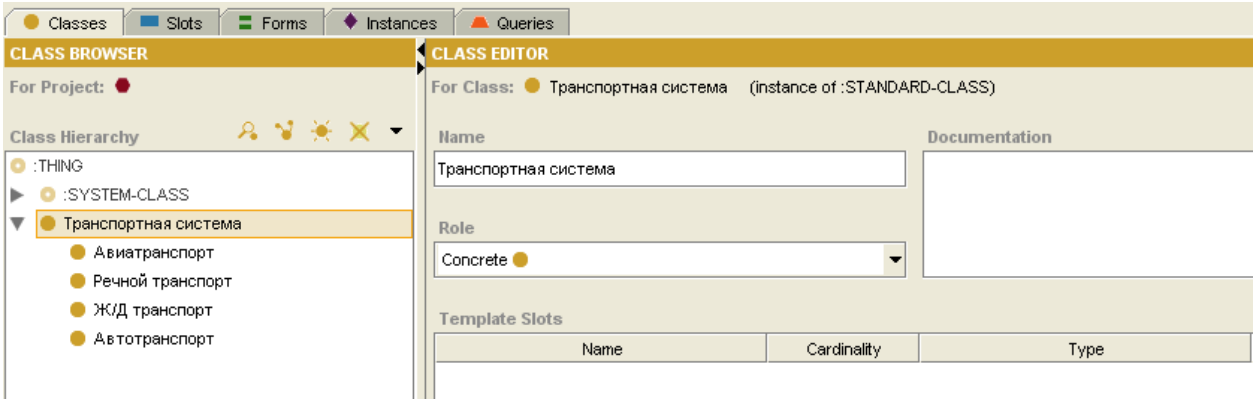

Figure 1.9. Edited Tree Hierarchy

For further graphical representation, create the “Семантическая сеть” (Semantic Web) class at the same level with the “Транспортная система” (Transport system) class (figure 1.10).

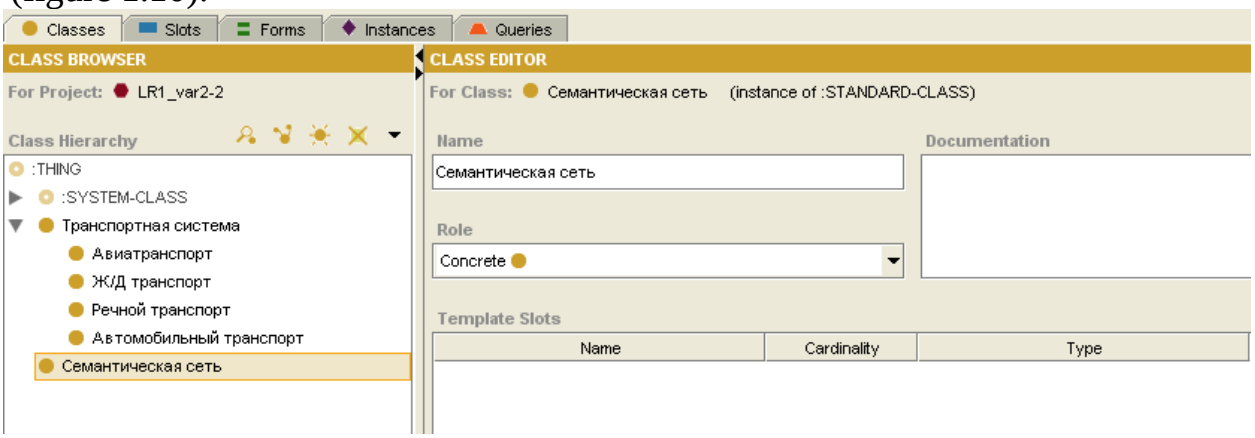

Figure 1.10. Creating the class “Семантическая сеть” (Semantic Web)

1.1.2.7. Creating slots

In Protégé classes are specific domain concepts such as Air transport, or Motor transport. At the same time classes are larger than objects, combined in a hierarchy. They also can have attributes (properties) and relations between them such as capacity, cost and time of transportation, cargo space, etc.

Attributes and relations are described by slots. In this section it will be shown how to create slots, attach them to classes, and describe relations among classes. One can also find a mechanism of slot inheritance.

Managing the Slots tab is similar to the Classes tab. Created slots are displayed on the left of the browser, and editing is possible with the editor on the right.

There are several ways to create a slot. Consider one of them.

In the main window select the tab Slots, then press the Create Slot button in the upper right corner of the Slot Hierarchy (figure 1.11).

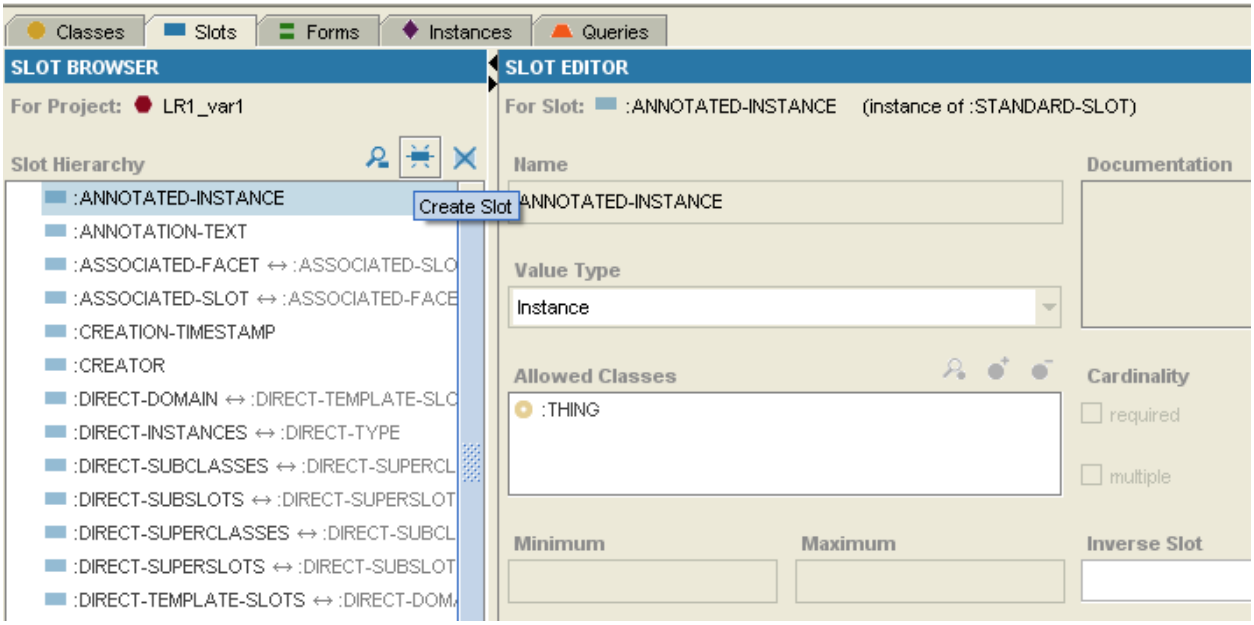

Figure 1.11. Creating the Slot window

This will create a new slot. As with creating a class it will be given a default name. Rename it to “название” (name). Slot has the default value type String. The type imposes restrictions on what values a slot can take on. For example, the string slot can take on as values alphanumeric strings (including spaces). For this simple slot it is just needed to tick the Multiple box. In order to attach it to several classes, it is not necessary to change facets in slot editor (figure 1.12).

Similarly, create the slots “грузоподъемность(кг)” (loading capacity(kg)), “объем отсека(м3)” (cargo space(m3)), “время(ч)” (time(h)), “стоимость(руб/кг)” (cost(rub/kg)). These slots will not be strings. For this reason in the Value Type box select the Float type, and in the Minimum box put “0” to be sure that any value of these fields will not be negative.

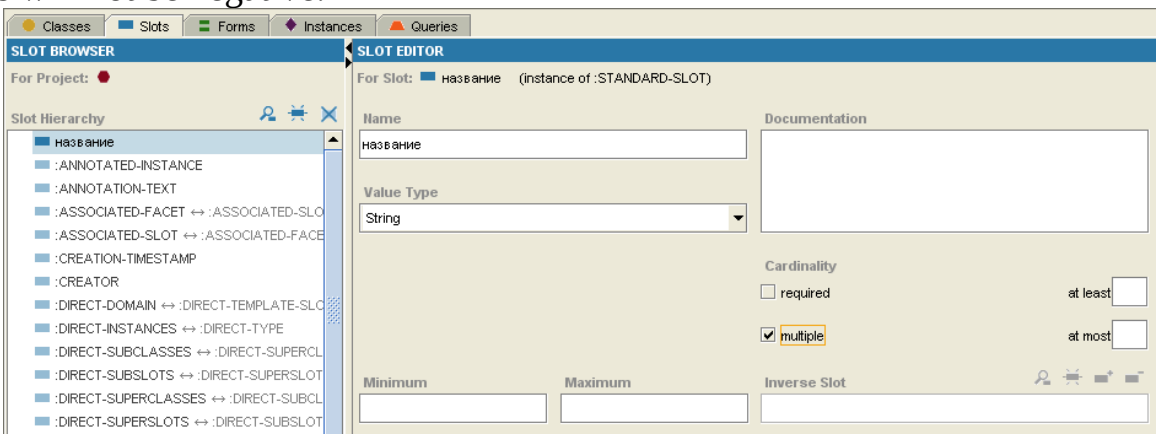

Figure 1.12. Creating the “название” (name) slot

Also create the slot “круглогодичность” (year-round); select the String type. For all created slots tick the Multiple box (figure 1.13).

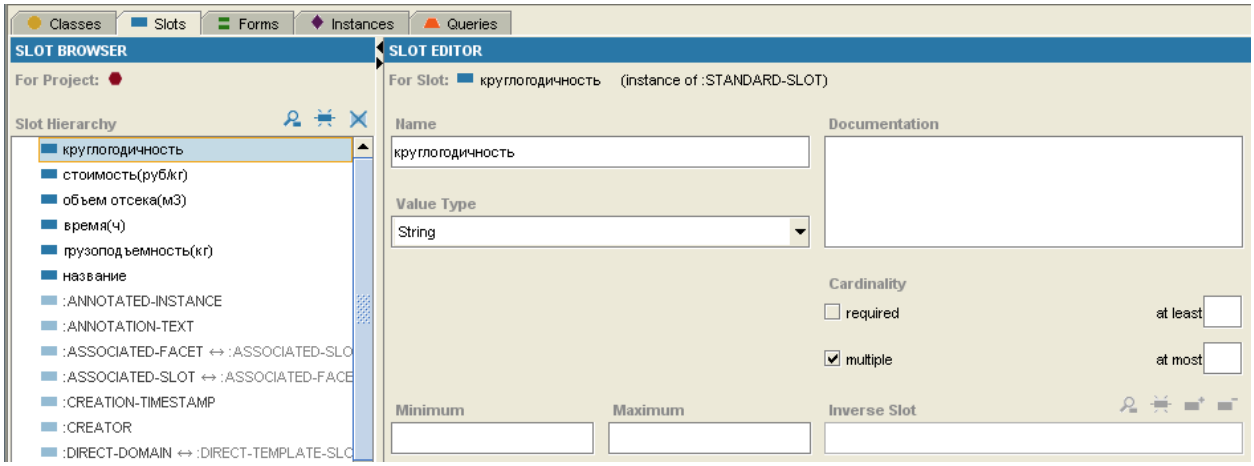

Figure 1.13. Adding slots

1.1.2.8. Adding the slots to classes

At this point you have identified the general attribute, “название” (name). In order to use it in the developing ontology it is needed to attach it to a class.

Navigate to Classes and open the “Транспортная система” (Transport system) class for editing. Any attribute created or associated with a class, is displayed in the Class editor, on the right of the Class browser, while there are no records. Select the “Транспортная система” (Transport system) class to attach it to the “название” (name) slot. In the Class editor in the Template box slots click the Add Slot button. A list of available slots in your project will appear in alphabetical order except the Protégé system classes which are located in the bottom of the list. Select the “название” (name) slot (figure 1.14).

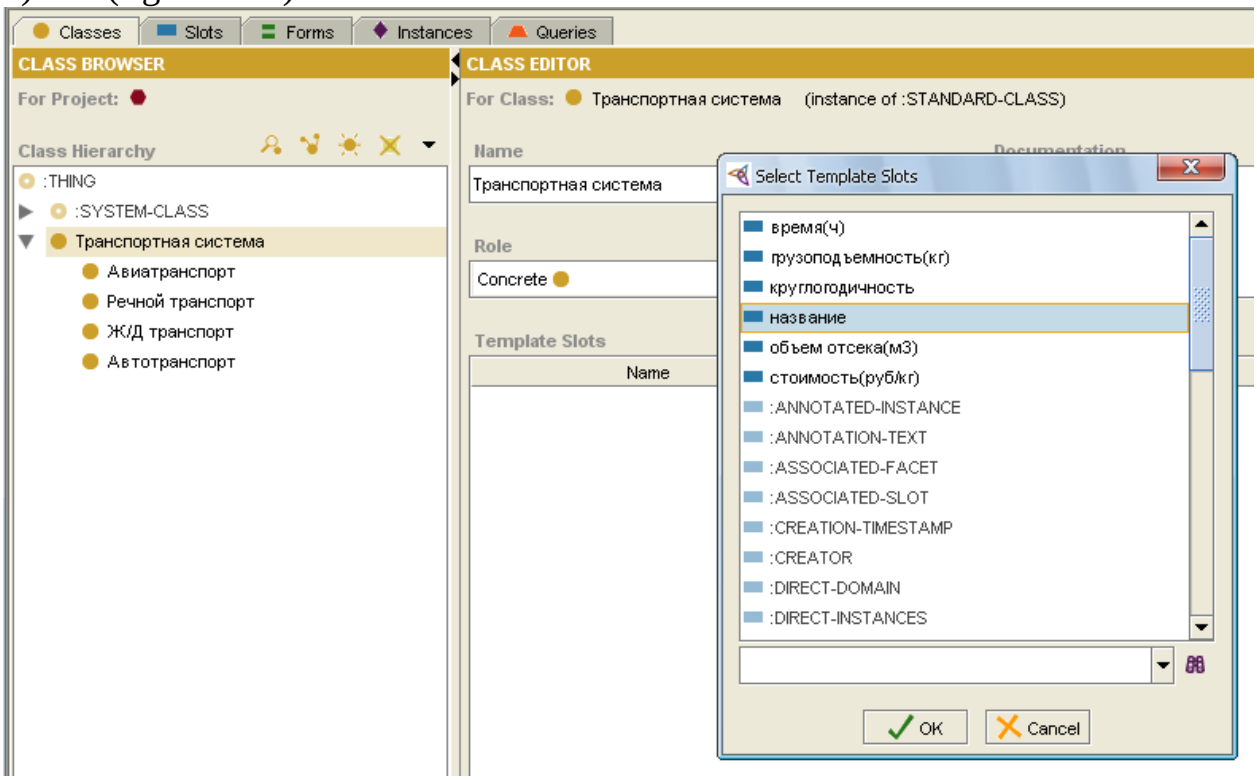

Figure 1.14. Adding a slot to a class

To add several slots at once to the Select Template Slot box select the slots holding Ctrl and click OK.

Now in the Template slots pane one can see all the slots included in the “Транспортная система” (Transport system) class as well as their properties: the capacity (the number of elements of type), the type itself (String, Float) and the minimum value (Minimum) (figure 1.15).

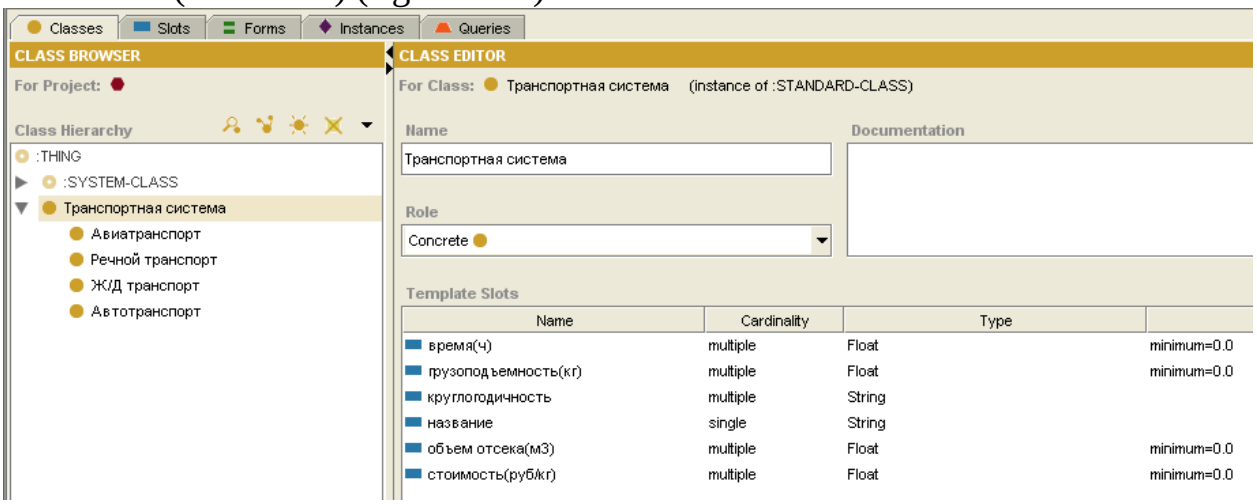

Figure 1.15. Slots of the “Транспортная система” (Transport system) class

Since “Авиатранспорт” (Air transport), “Автомобильный транспорт” (Motor transport), “Ж/Д транспорт” (Rail transport), and “Речной транспорт” (River transport) are the subclasses of the “Транспортная система” (Transport system) class and have the same slots, they are displayed in the same Template Slots pane in parentheses (figure 1.16).

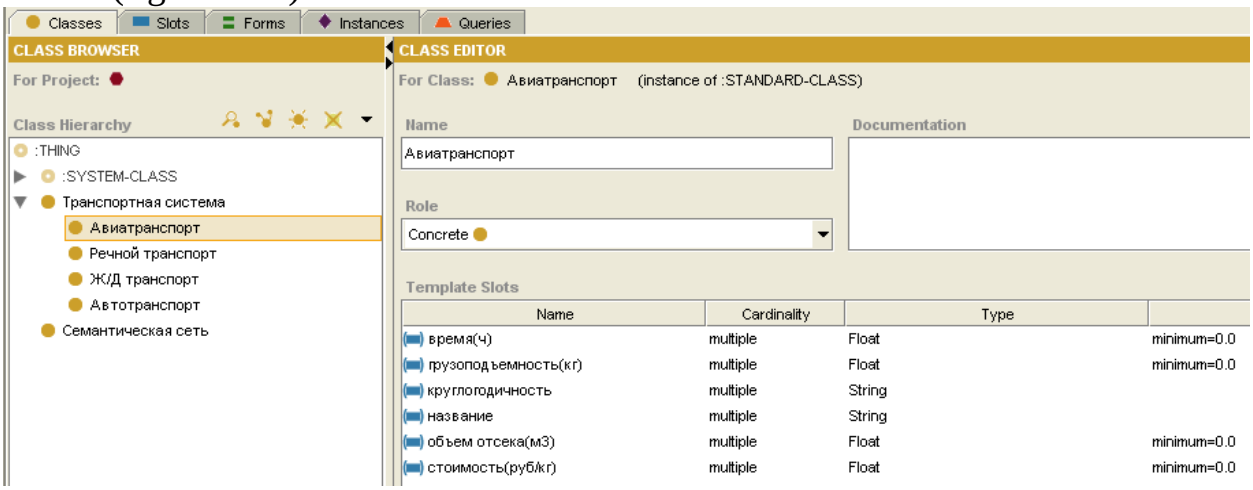

Figure 1.16. The slots of the “Авиатранспорт” (Air transport) class

1.1.2.9. Assigning a slot to a class

In paragraph 1.1.2.8 it was described how to select necessary slots for a class. To assign a slot to a class can be done in a different way.

Open the Slots tab and select the “название” (name) slot. At this stage in the Slot editor in the Domain pane there is just one class, “Транспортная система” (transport system) (figure 1.17).

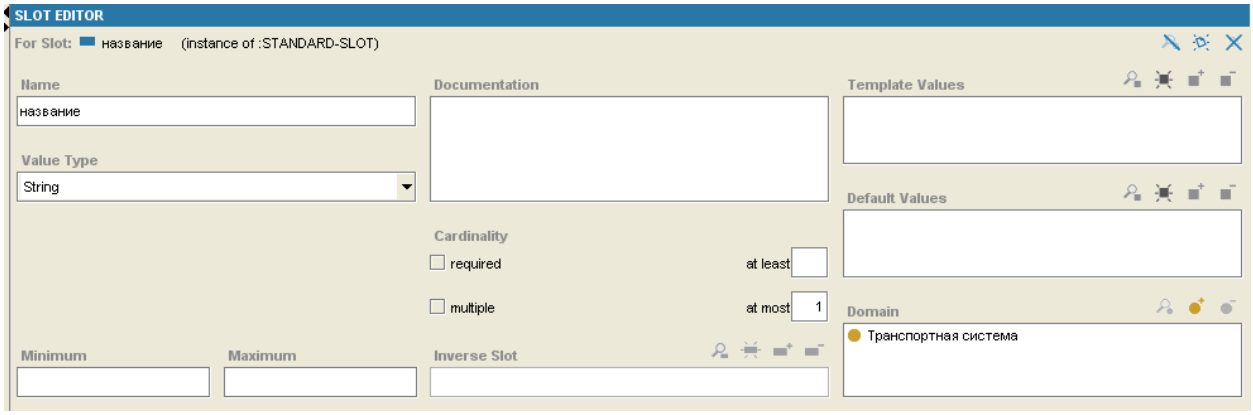

Figure 1.17 The Slot editor pane for the “название” (name) slot

To add a class click the Add Class button and select the “Семантическая сеть” (Semantic Web) class from the list, then click OK (figure 1.18).

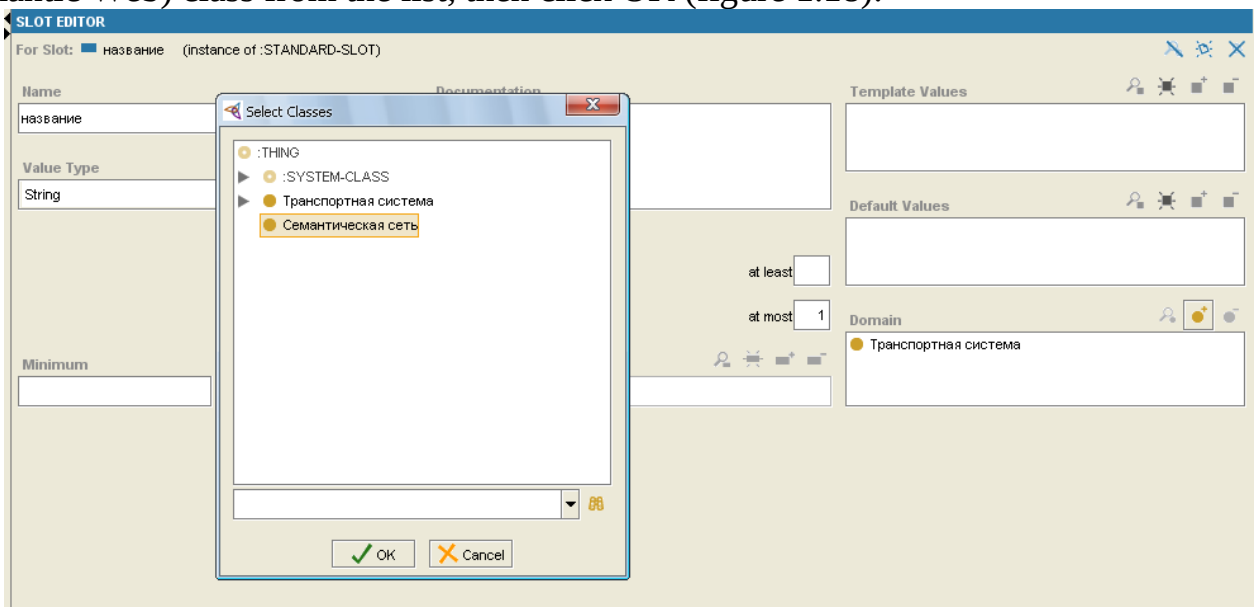

Figure 1.18. Assigning a slot to a class

Now in the properties of the “название” (name) slot in the “Domain” box there are two classes, “Транспортная система” (Transport system) and “Семантическая сеть” (Semantic Web) (figure 1.19).

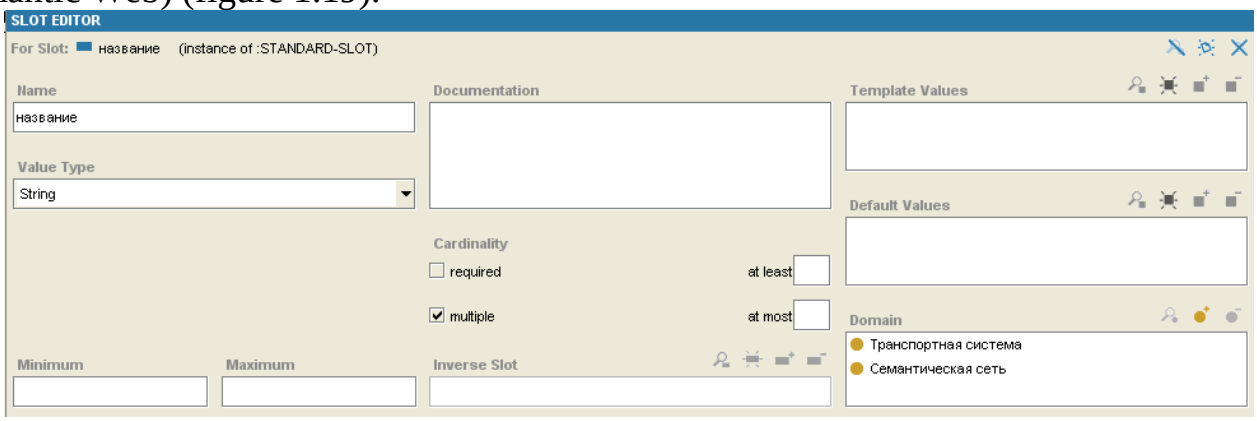

Figure 1.19. Edited properties of the “название” (name) slot

If you navigate to the Classes tab, you will see that in the properties of class “Семантическая сеть” (Semantic Web) the there is the “название” (name) slot (figure 1.20).

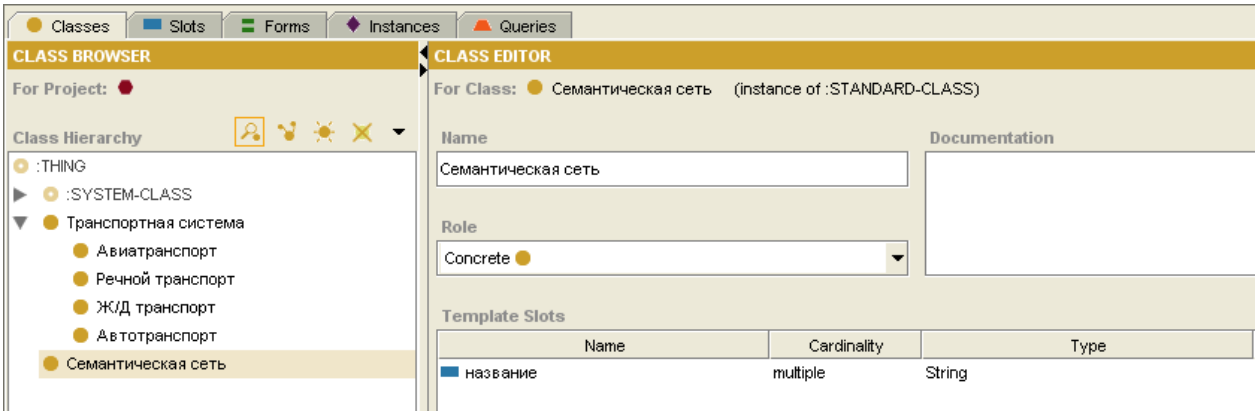

Figure 1.20. The properties of the “Семантическая сеть” (Semantic Web) class

1.1.2.10. Creating relations among classes

The Protégé system allows creating slots that can be used to describe relations among classes that are not defined in the class hierarchy. To do this, there are the Instance and Class slots. For example, the “Семантическая сеть” (Semantic Web) class was created for ontology graphical representation. Now you need to assign it to the “Транспортная система” (Transport System) class in a way that it would not contradict the developing ontology. You can create a new slot that will describe the relations among classes.

In the Class browser select the “Семантическая сеть” (Semantic Web) class. Click the Create Slot button to create and assign the new slot to this class. In the opened window for editing, type “организационная структура” (organisational structure) in the Name box, select Instance in the Value Type box, and select “Транспортная система” (Transport system) from the list using the Add Class button in the Allowed Classes box. In order for the semantic web to have more than one connection tick the Multiple box on the Cardinality pane (figure 1.21). Close the window when the properties of the slot are created. The changes will be saved automatically.

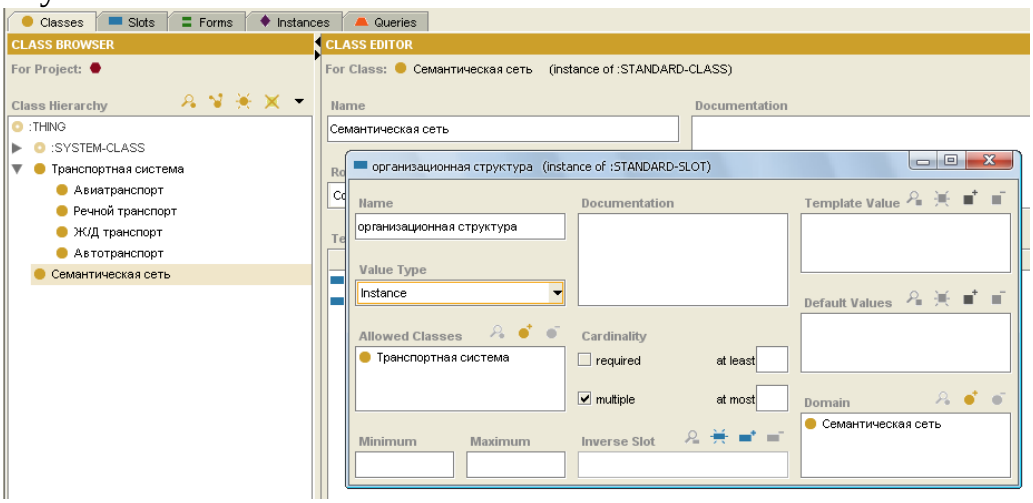

Figure 1.21. Creating the “организационная структура” (organisational structure) slot

In a similar way create the “есть” (is) slot for the “Транспортная система” (Transport system) class and set the same properties as in the previous slot.

Now, the ontology can be represented as a semantic web using the created slots, where instances of the “Транспортная система” (Transport system) class and the relations among them will be displayed.

1.1.2.11. Setting up instance forms

For each class of the ontology Protégé generates a default form which can be used to enter instances. The forms contain different fields of data entry called widgets, for each slot connected with a class. For different data types of slots there are different widgets, for example:

- TextFieldWidget, for the slots with the data type String;
- IntegerFieldWidget, for the fields which values are presented as an integer;
- InstanceListWidget, for the slots which have instances as a type and the number of elements is more than one, etc.

In Protégé it is possible to change the standard form of instances representation. The name, size, type, location and displaying of the widget can be modified.

Open the Forms tab and select the “Транспортная система” (Transport system) class. You will see the standard form of instance representation (figure 1.22).

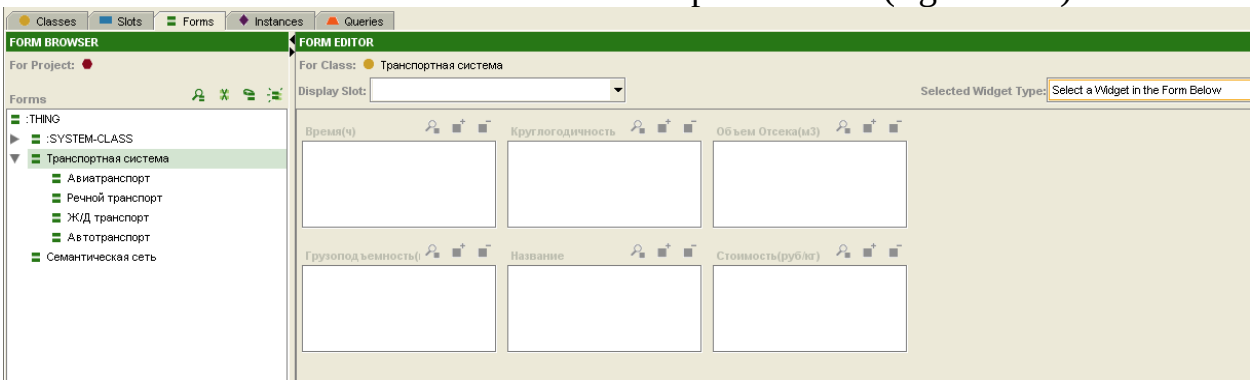

Figure 1.22. The standard form of widgets

To see how the changes will be presented in the Instance editor navigate to the Instance tab, select “Транспортная система” (Transport system) and click the Create Instances button in the Instances Browser. New widgets will appear in the same form as in the Forms tab in the Instance Editor pane that was empty until now (figure 1.23).

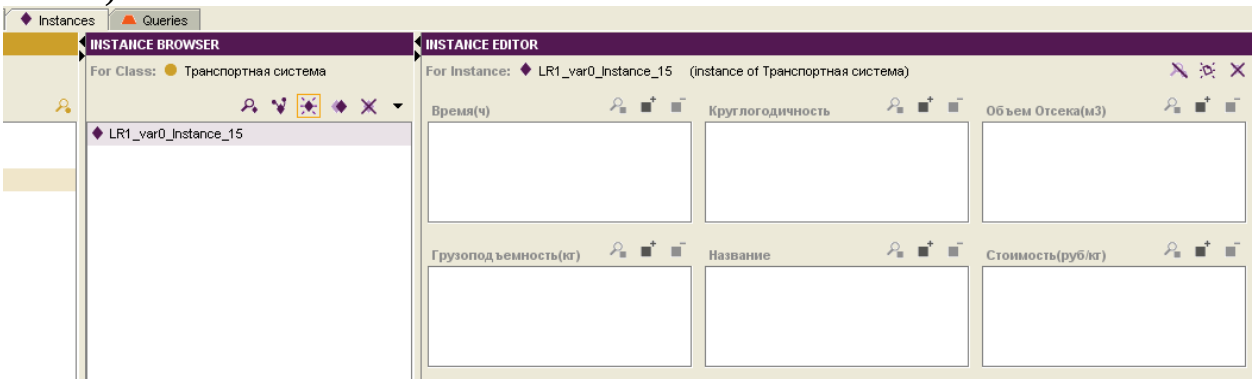

Figure 1.23. Instances window

1.1.2.12. Changing a widget's parameters

Resizing a widget

To resize a widget, click a mouse on it, and you will see a green frame (figure 1.24). It means that the widget is selected and can be now modified.

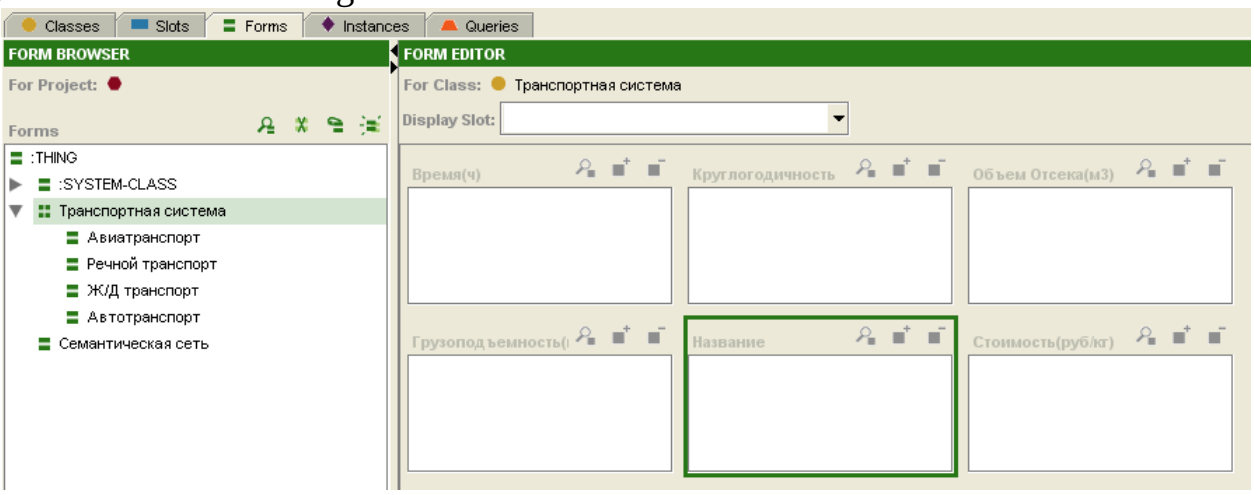

Figure 1.24. Selecting a widget for editing

Click on the widget and holding the mouse drag to resize it (figure 1.25).

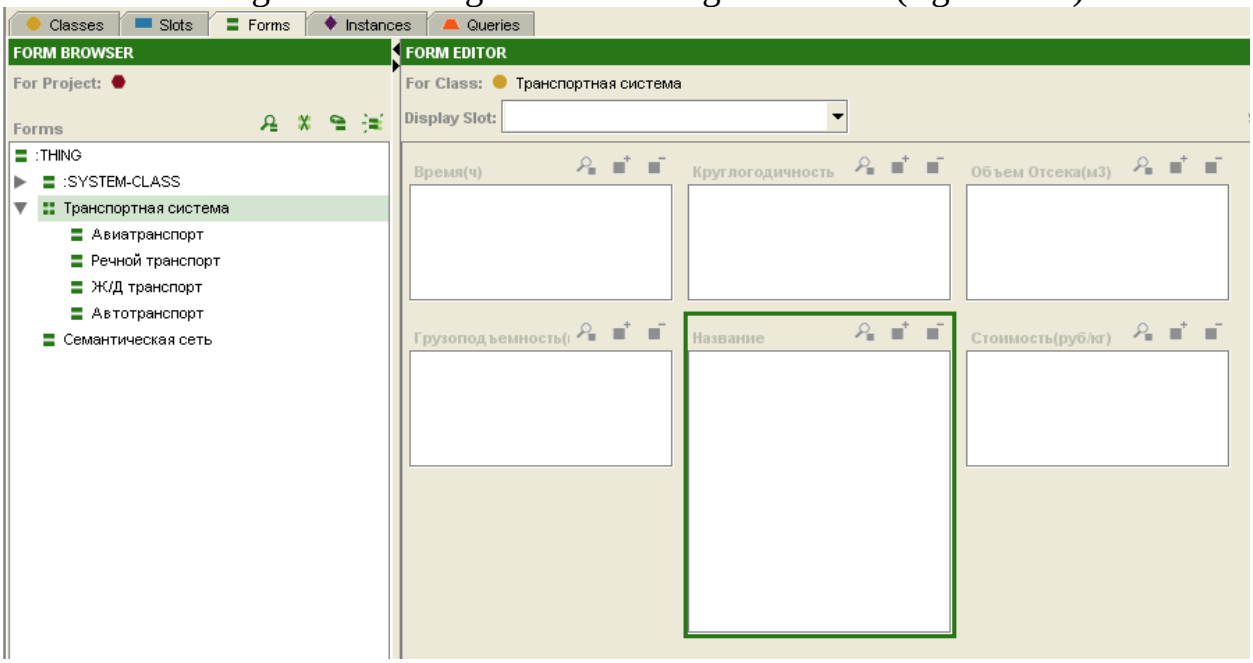

Figure 1.25. Resizing the "Название" (name) widget

Notice that the icon in front of the "Транспортная система" (Transport system) class in the Form Browser has changed from to . The new icon shows that the form of this class has been customised.

Moving a widget

To move a widget click it and holding the left mouse button drag it to another place in the Forms window (figure 1.26).

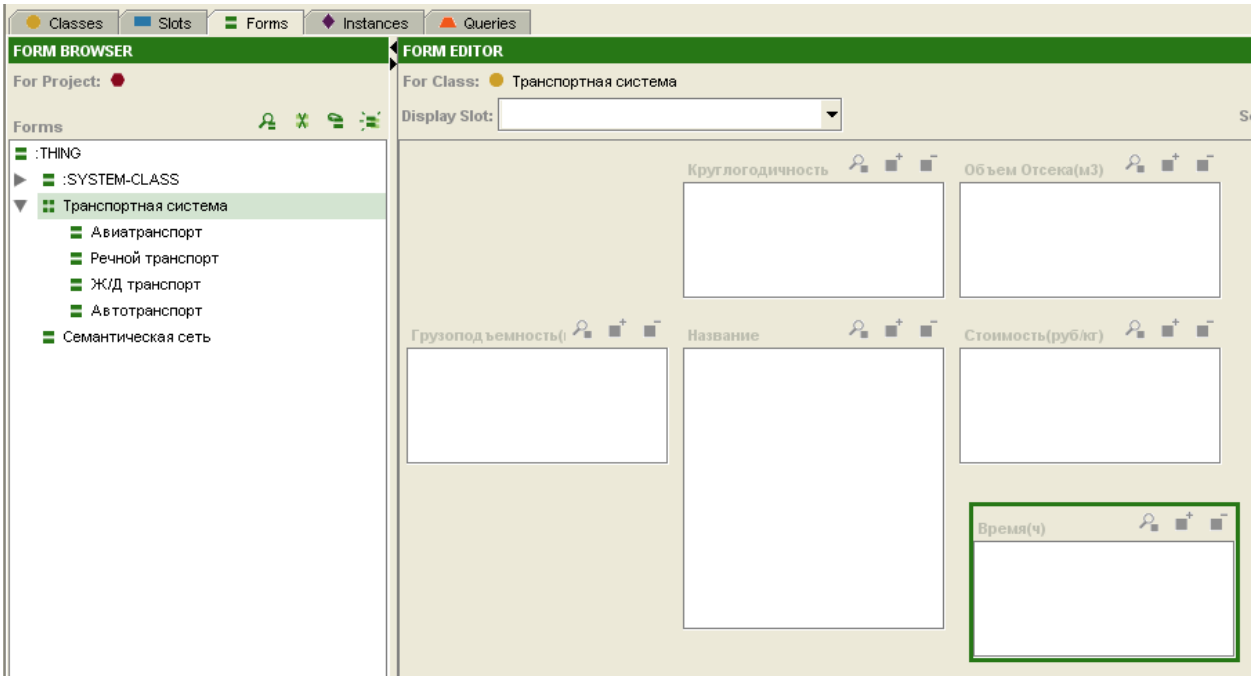

Figure 1.26. Moving the “Время” (time) widget

Change the position and the form of the widget in accordance with figure 1.27.

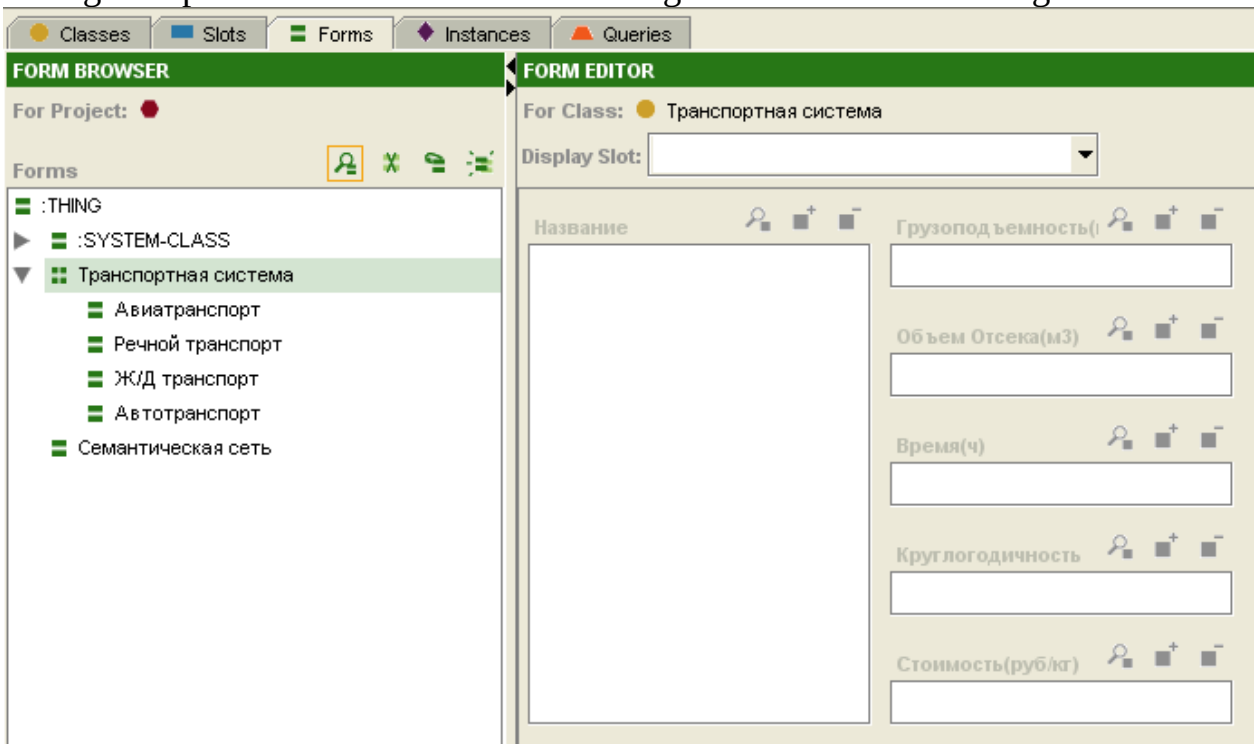

Figure 1.27. The edited Forms window

Customising widget buttons

You can customise widgets to display a different label or a different set of buttons than the default. For example, to delete the button double-click the widget. Click the Buttons tab, and click the box in front of Show View Value button (figure 1.28). Click OK.

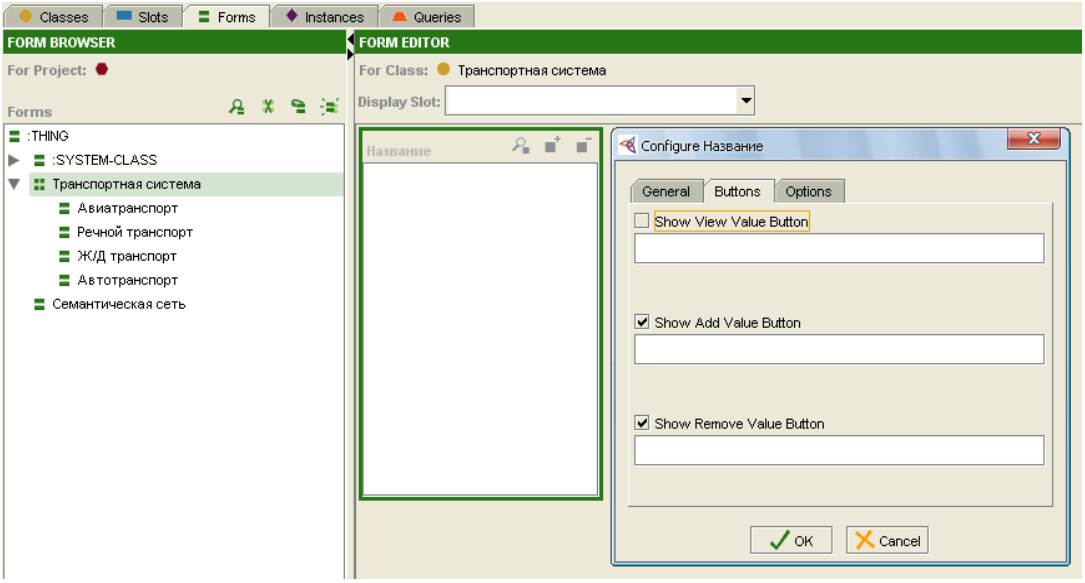

Figure 1.28. Customising the widget's buttons

After, the widget window will be changed (fig .1.29).

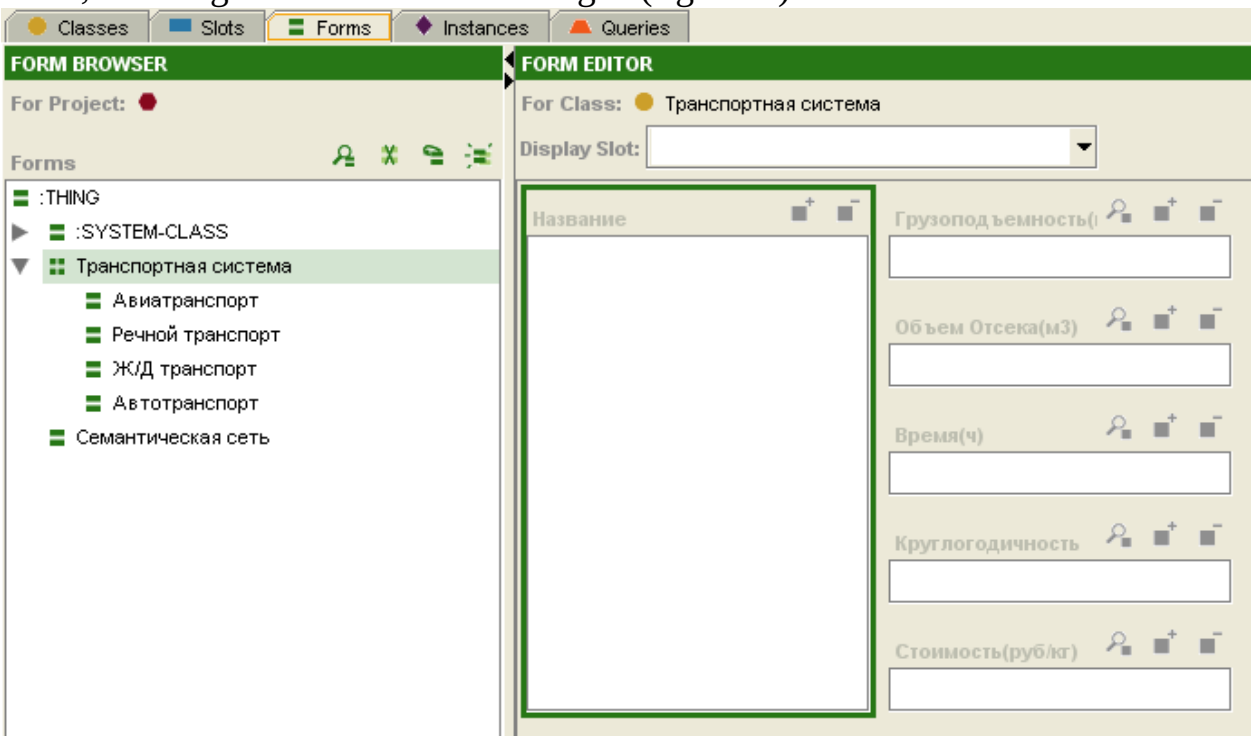

Figure 1.29. Customised window of the widget

Hiding a widget

You can hide a widget so it cannot be seen in the Form and Instance Editors (this does not remove any information from the ontology).

To hide the widget for the "название" (name) slot click it and select <none>in the upper right corner in the Selected Widget Type combo box (Figure 1.30).

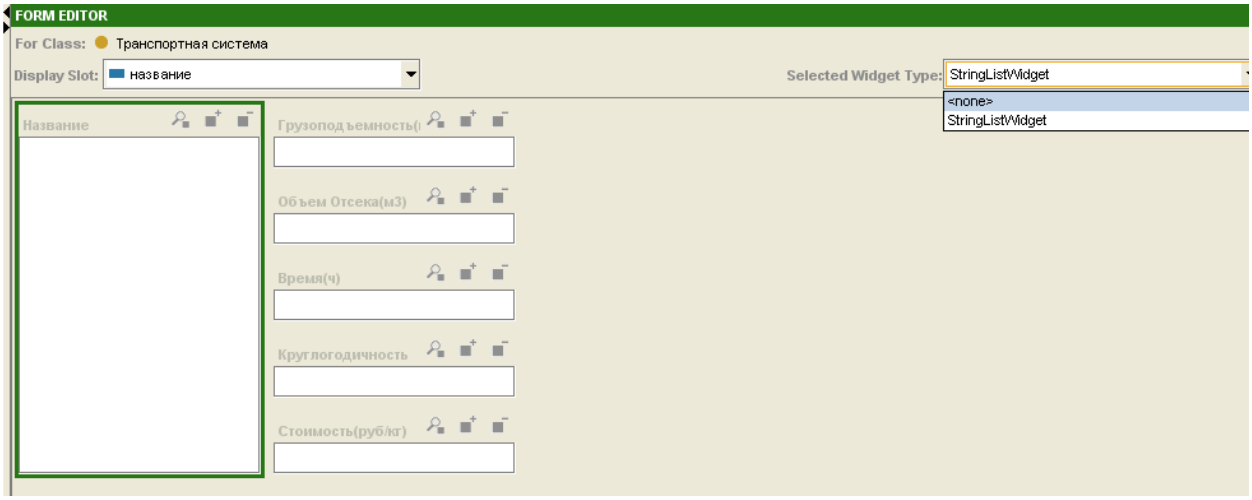

Figure 1.30. Hiding the widget

The widget is no longer visible on the Form Editor (figure 1.31).

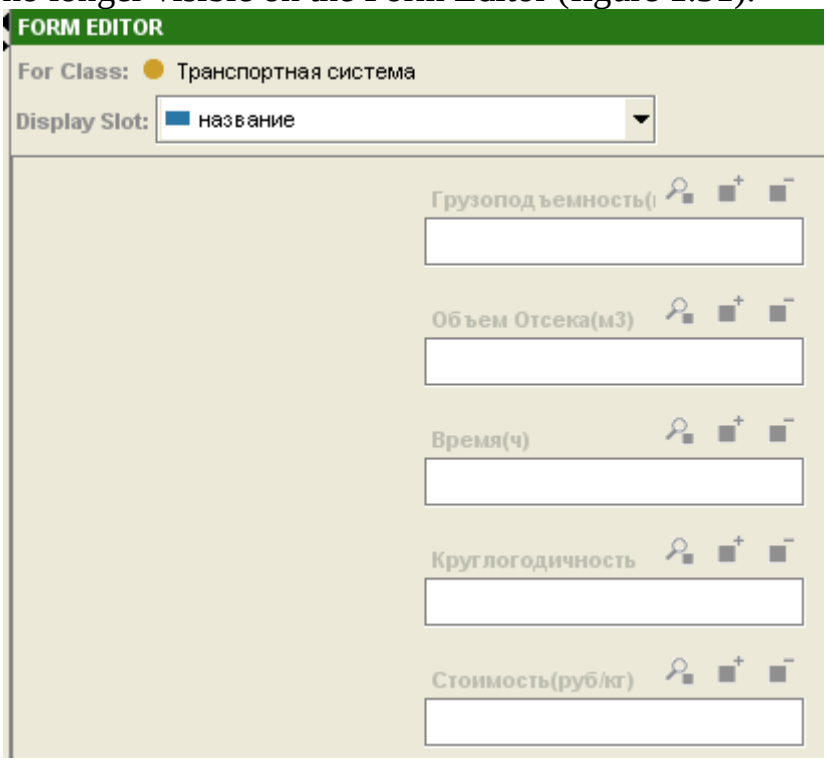

Figure 1.31. The modified Form Editor window

Displaying a hidden widget

To restore a hidden widget to view in the Form Browser click the View Form Customisations  button.

In the Configure form dialogue box you can see a list of all the slots and their corresponding widgets. Click on <none> and select String List Widget. Click OK (figure 1.32).

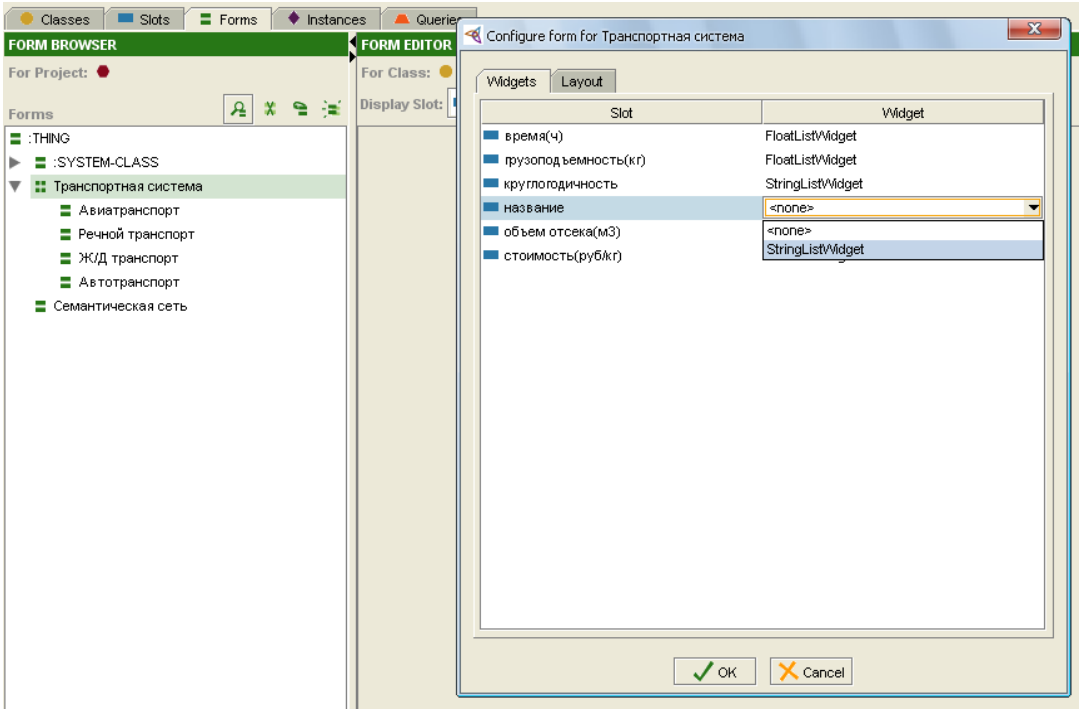

Figure 1.32. Modifying the hidden widget

Now the widget is again visible in standard form in the Form Browser (figure 1.33).

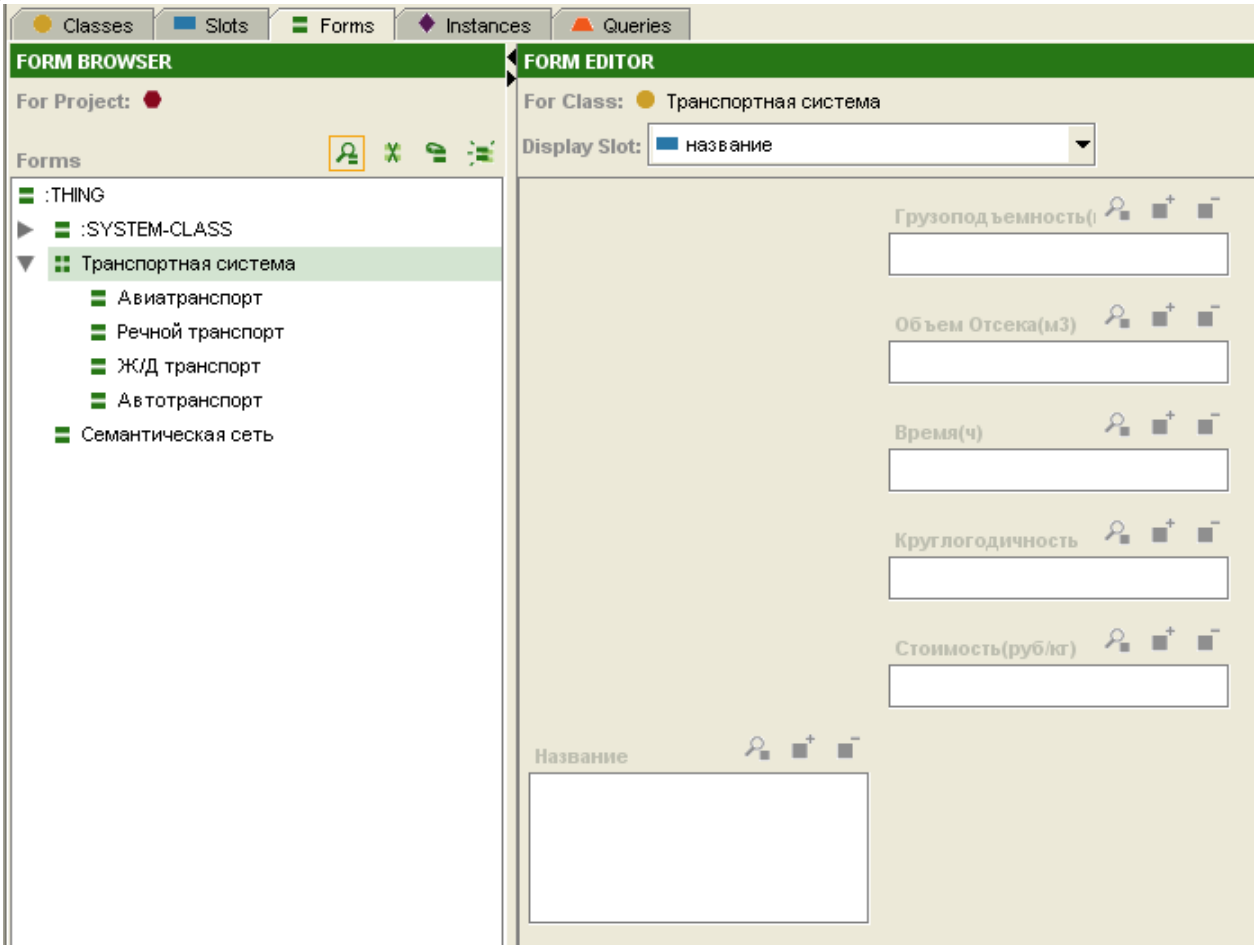

Figure 1.33. Displaying the hidden widget

Resize and move the widget for the subclasses of the “Транспортная система” (Transport system) class as it is shown on figure 1.27. Hide all the widgets except the “название” (name) widget for the “Транспортная система” (Transport system) class (figure 1.34).

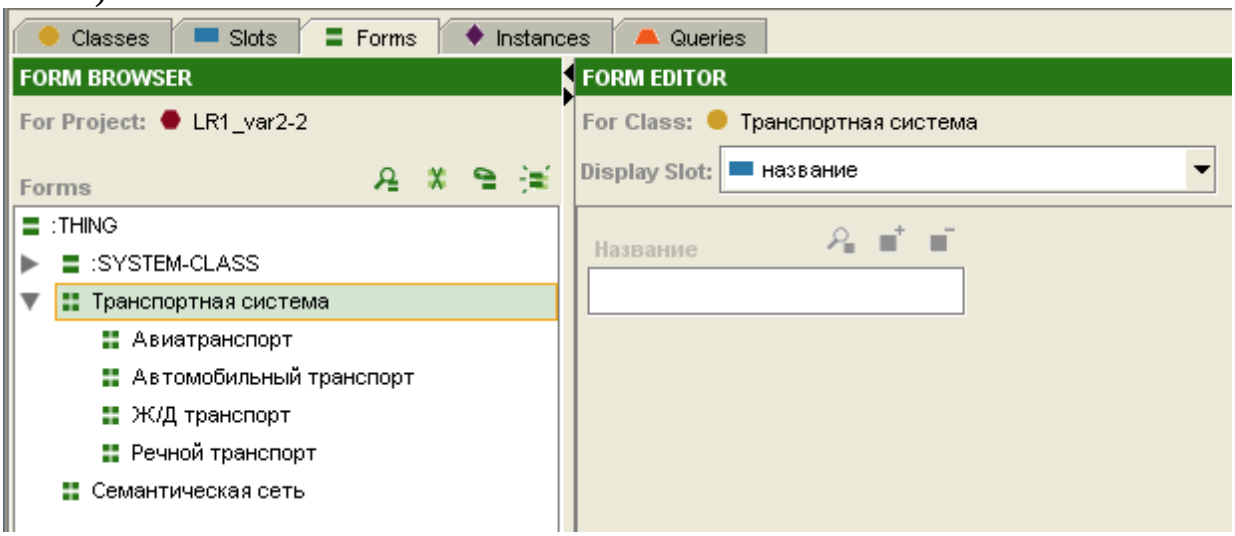

Figure 1.34. The “Транспортная система” (Transport system) class widget

1.1.2.13. Creating instances

Instances of classes are the actual data in the developing knowledge base. Before entering data, it is required to check the project structure, because after the data is entered, structure modifying can cause a loss of some information. In addition, if you add slots you will have to go back and fill in the slot values for all instances that were created previously.

To create an instance for the “Авиатранспорт” (Air transport) class, click on the Instances Tab and select “Авиатранспорт” (Air transport). Press the Create Instances button in the Instances Browser. In the Instances Editor pane slots will appear in the form you created. In the “Название” (Name) box in the Instances Editor pane click the Add Value button. In the Create String Value window put the cursor in the input line, and type the name of the aircraft as “Ан-12” (AN-12). Click OK (figure 1.35).

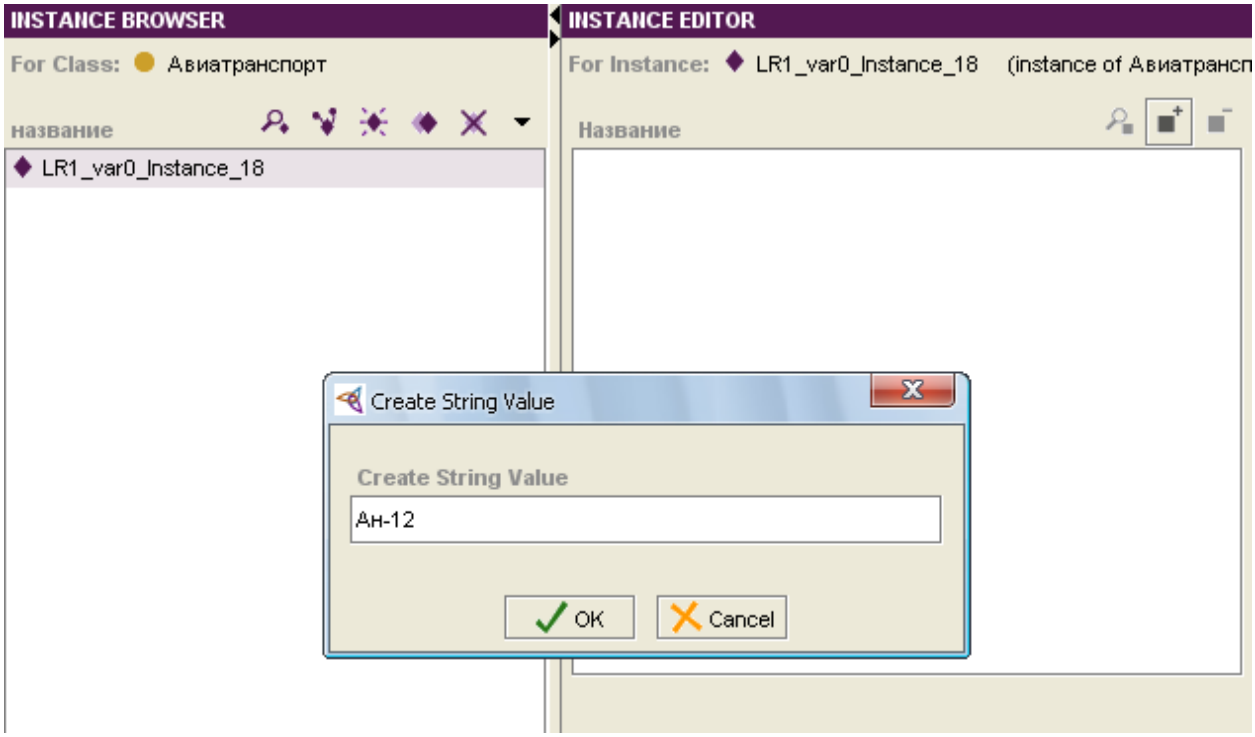

Figure 1.35. Creating instances

Similarly, pressing the button enter the data (see Table 1.1) for the instance “Ан-12” (An-12) in the strings “Грузоподъемность” (Loading Capacity), “Объем отсека” (Cargo space), “Время” (Time), etc. Notice, that the name in the Instances Browser will remain standard (figure 1.36).

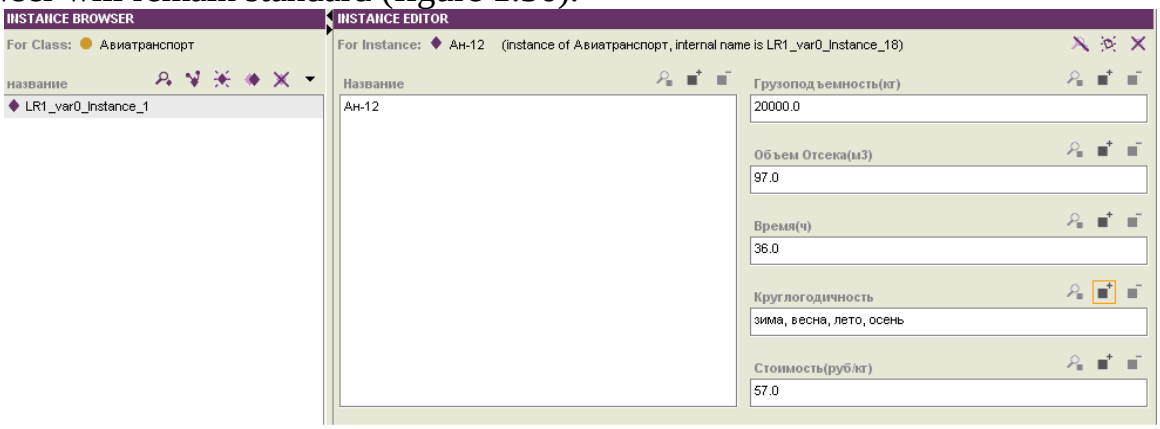

Figure 1.36. Created instance

In Protégé, if you attempt to enter slot values that do not satisfy the restrictions of the slot the value will appear in red, that is in the field with the floating point number format you cannot enter alphabetic data.

Table 1.1. Characteristics of vehicles

Vehicle	Loading Capacity (kg)	Cargo Space (m3)	Time (h)	Year-round	Maximum cost (rub/kg)
Air transport					
Ан-12 (An-12)	20 000	97	36	Winter, spring, summer, autumn	57
Ан-24 (An-24)	5 500	110	24		61
Ил-76 (Il-76)	48 000	180	48		55
Як-76 (Yak-76)	2 700	51	24		63
Rail transport					
Coach 11-066	66 000	86	72	Winter, spring, summer, autumn	6
Coach 11-217	68 000	104	72		5.5
Coach 11-К1	68 000	138	72		5
Motor transport					
Mersedes Actros	16 000	48	54	Winter, spring, summer, autumn	3.8
Gaz 3302	1 500	9	36		4.8
Zil 5301	3 000	18	48		4
River transport					
Volga - Don	5 300 000	4 200	72	Spring, summer, autumn	4
Amur	2 500 000	1 500	68		3.5

1.1.2.14. Setting the display slot

For each class in the ontology you can specify one of its slots to be a display slot. Protégé will display the value of this slot any time it displays instances of the class. If a display slot is not set, Protégé will display the underlying system generated name for the instances. You may choose to set the display slots for the classes before starting creating instances.

To set the display slot for the “Авиатранспорт” (Air transport) select the Set Display Slot from the Instances Browser. Select “название” (Name) from the Display Slot menu (figure 1.37).

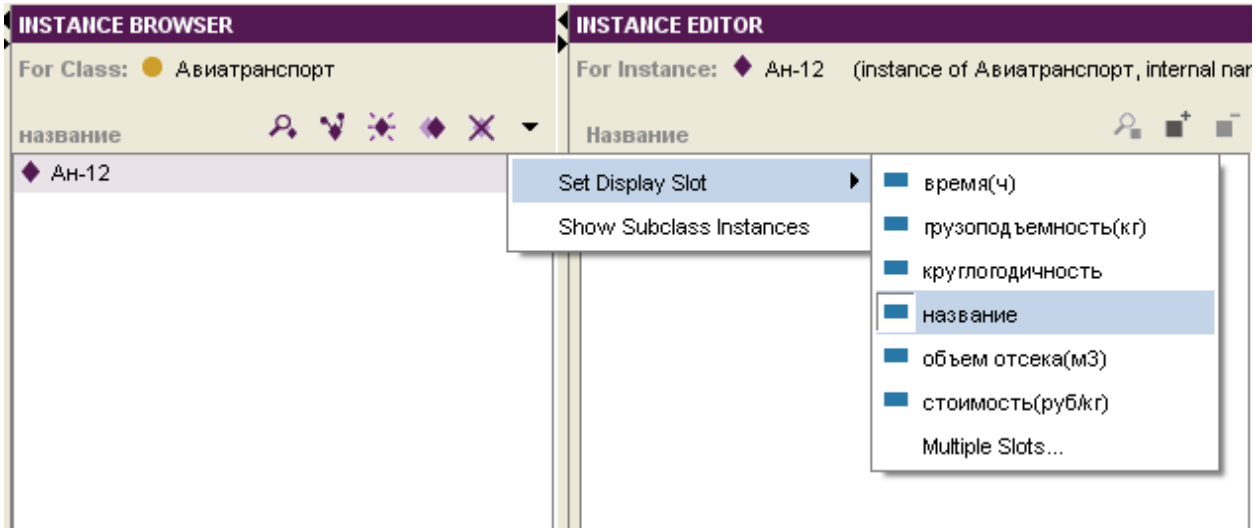

Figure 1.37. Modifying the standard name of the instance

Instances are now listed in the Instance Browser by alphabetical or numerical order.

Notice that the display of the “Авиатранспорт” (Air transport) class in the Class Hierarchy pane changed after a new instance was created. The (1) in the parentheses indicates that this class now has one instance.

Similarly, using the data of Table 1.1., create instances for the “Авиатранспорт” (Air transport), “Автомобильный транспорт” (Motor transport), “Ж/Д транспорт” (Rail transport), and “Речной транспорт” (River transport) classes. For these classes also select “название” (name) as a display slot.

1.1.2.15. Modifying widget parameters to create a semantic representation of an ontology

In section 1.1.2.10 the “Организационная структура” (Organisational structure) and “есть” (is) slots were created. In this section it will be shown how to create a semantic representation of an ontology using these slots.

Navigate to the Forms Tab and select “Семантическая сеть” (Semantic Web). In the Form Editor, widgets will be presented in a standard form (figure 1.38).

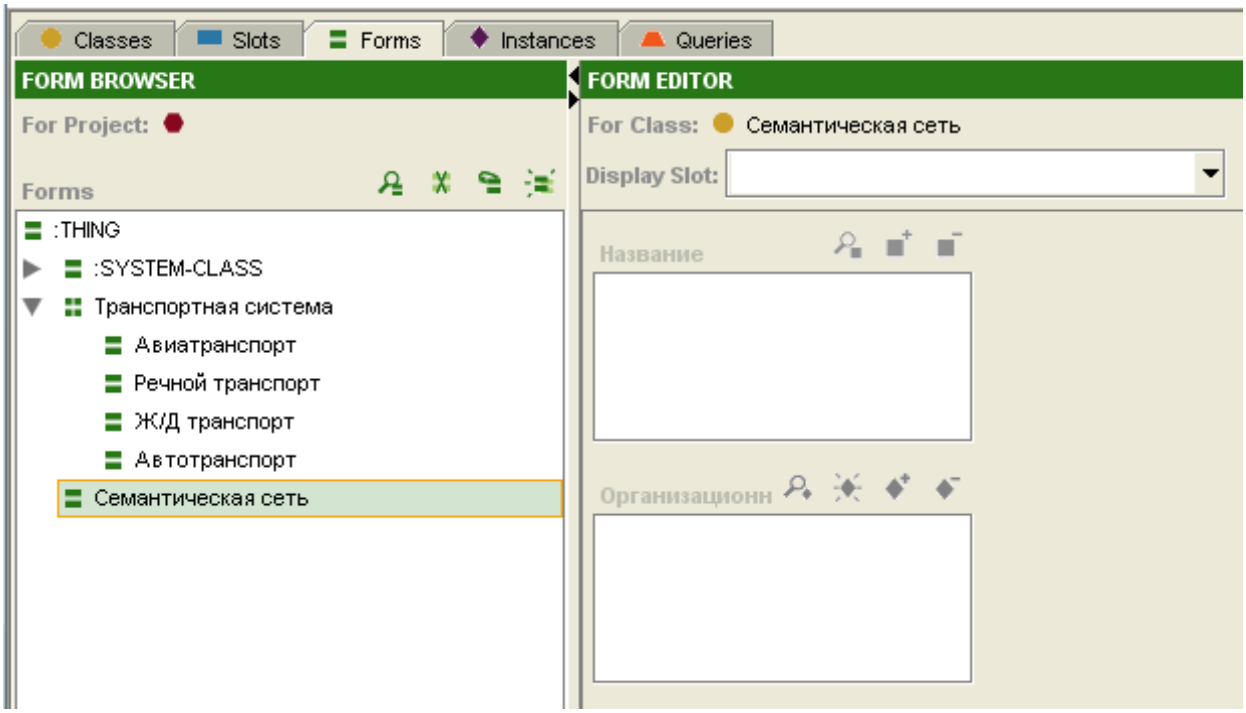

Figure 1.38. Standard form of widgets of the
“Семантическая сеть” (Semantic Web) class

Select the “Организационная структура” (Organisational structure) widget for modifying it.

Select Graph Widget from the Selected Widget Type combo box (figure 1.39).

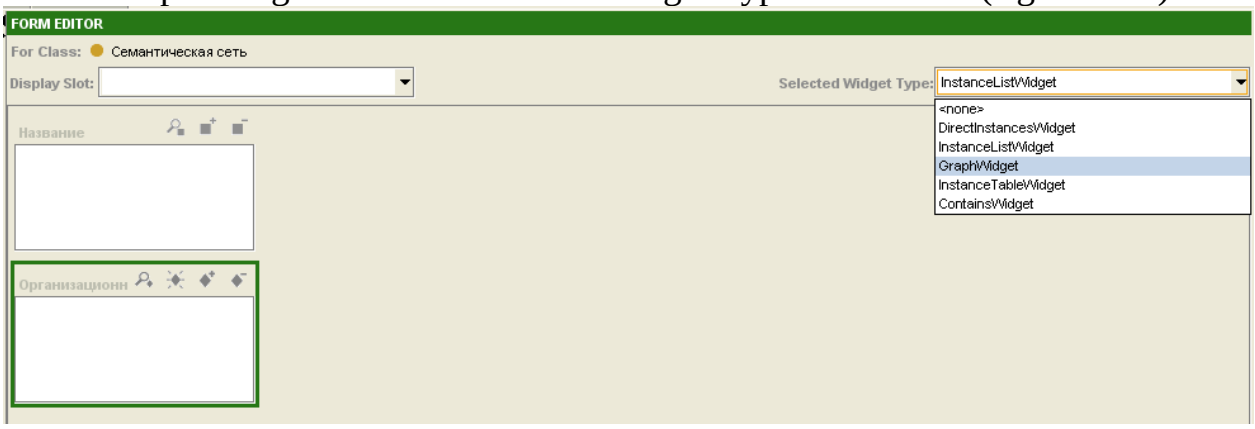

Figure 1.39. Selecting widget type

The widget pane will be presented as a graphic field with nodes (figure 1.40).

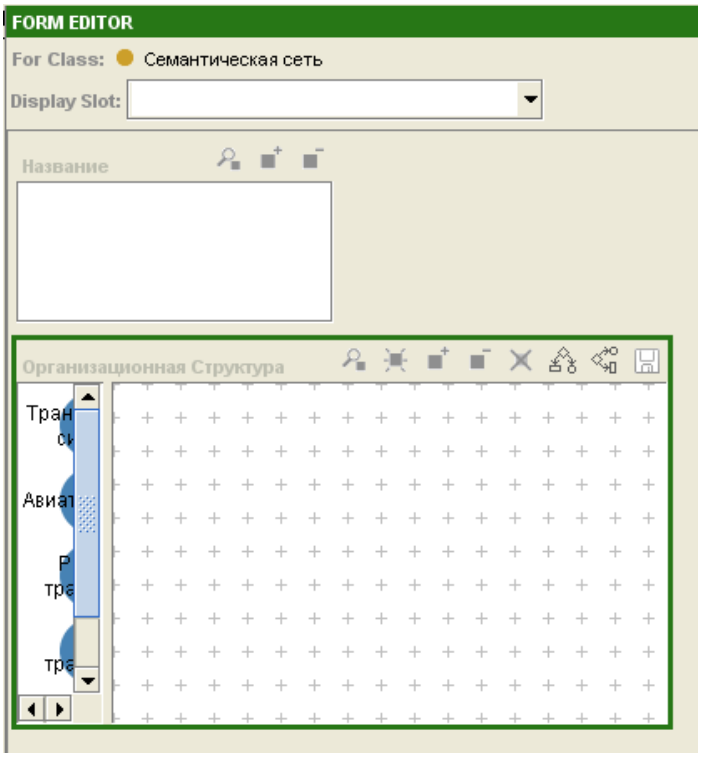

Figure 1.40. Graph Widget representation

To edit this widget double-click it and choose the Nodes tab in the Configure dialogue (figure 1.41).

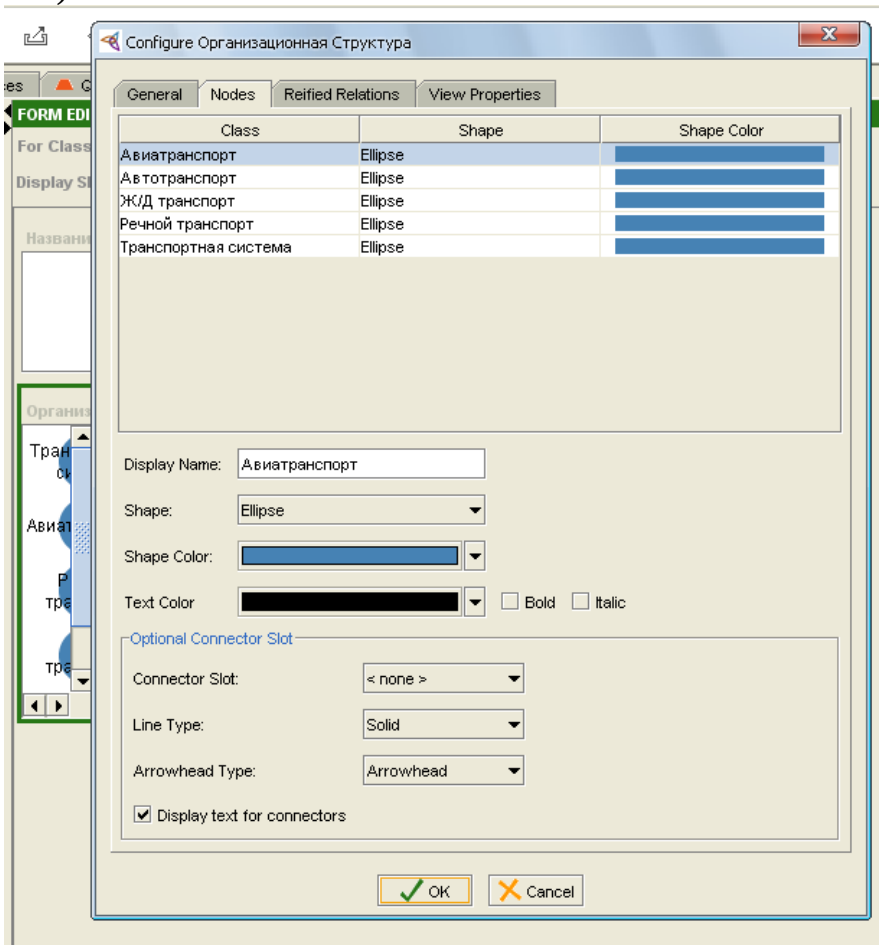

Figure 1.41. Graph Widget configuration dialogue

In this window you can change default settings, such as:

- Shape and shape colour which will be presented in the semantic web;
- Text colour and type;
- Line types.

Select the “Транспортная система” (Transport system) class. Set the Rectangle shape, choose a colour and the “есть” (is) slot in the Connector Slot string (figure 1.42).

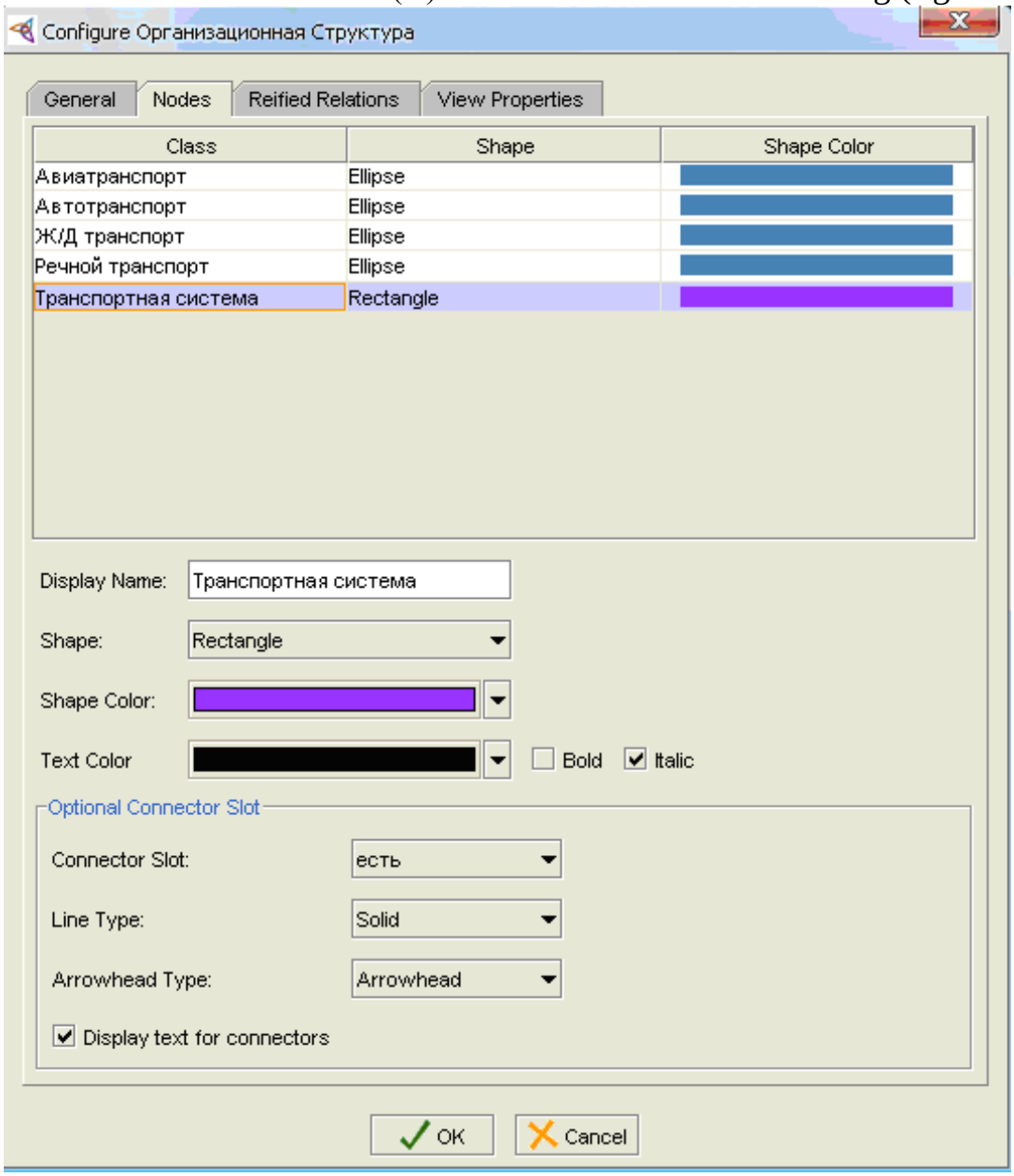

Figure 1.42. The “Транспортная система” (Transport system) Node editing

Set the same properties for the other classes (except the colour): the Octagon shape, and the “есть” (is) Connector Slot. Click OK. The final view is presented on figure 1.43.

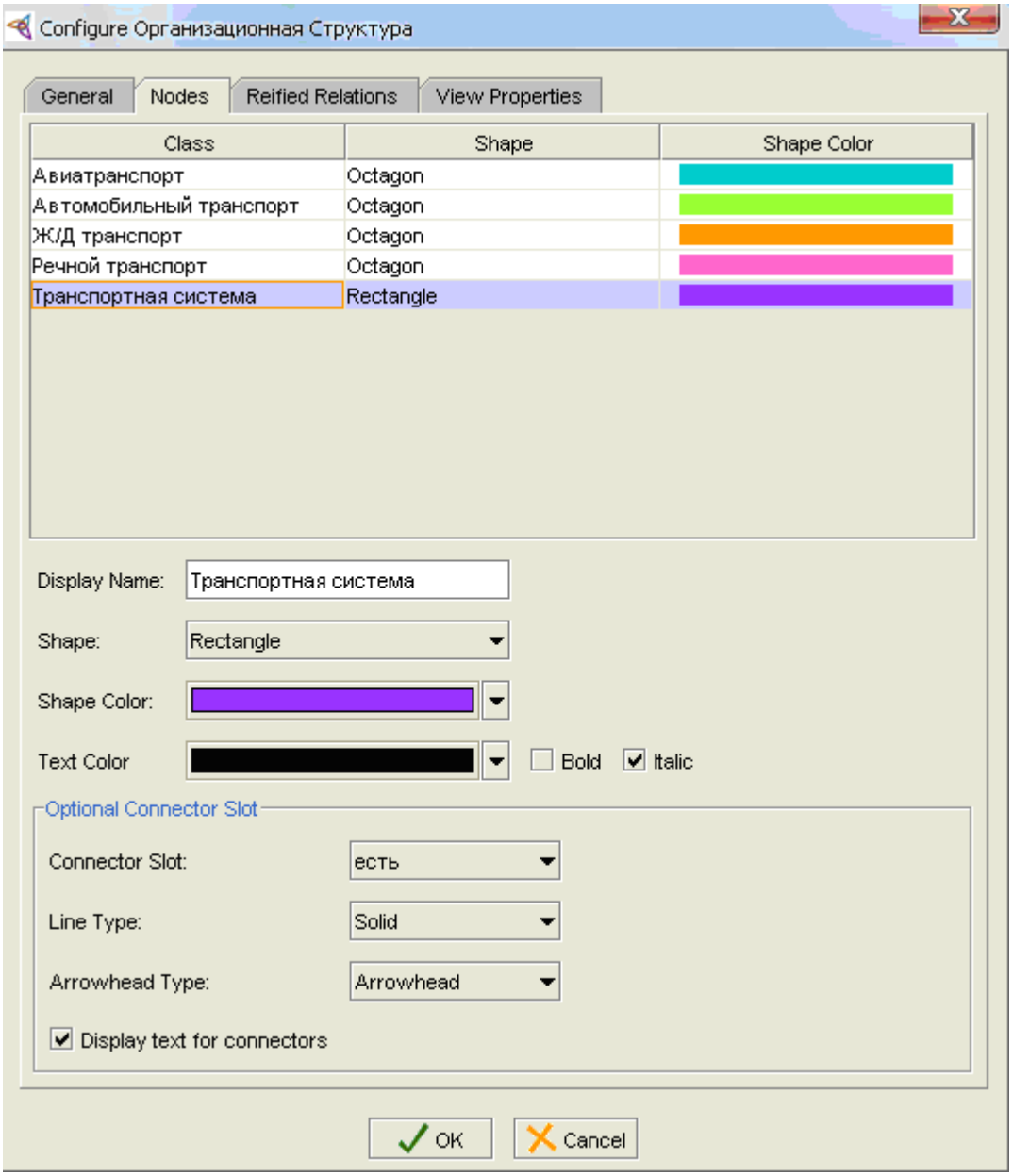

Figure 1.43. Modified properties of Nodes

For a better Semantic Web presentation, zoom Graph Widget (figure 1.44).

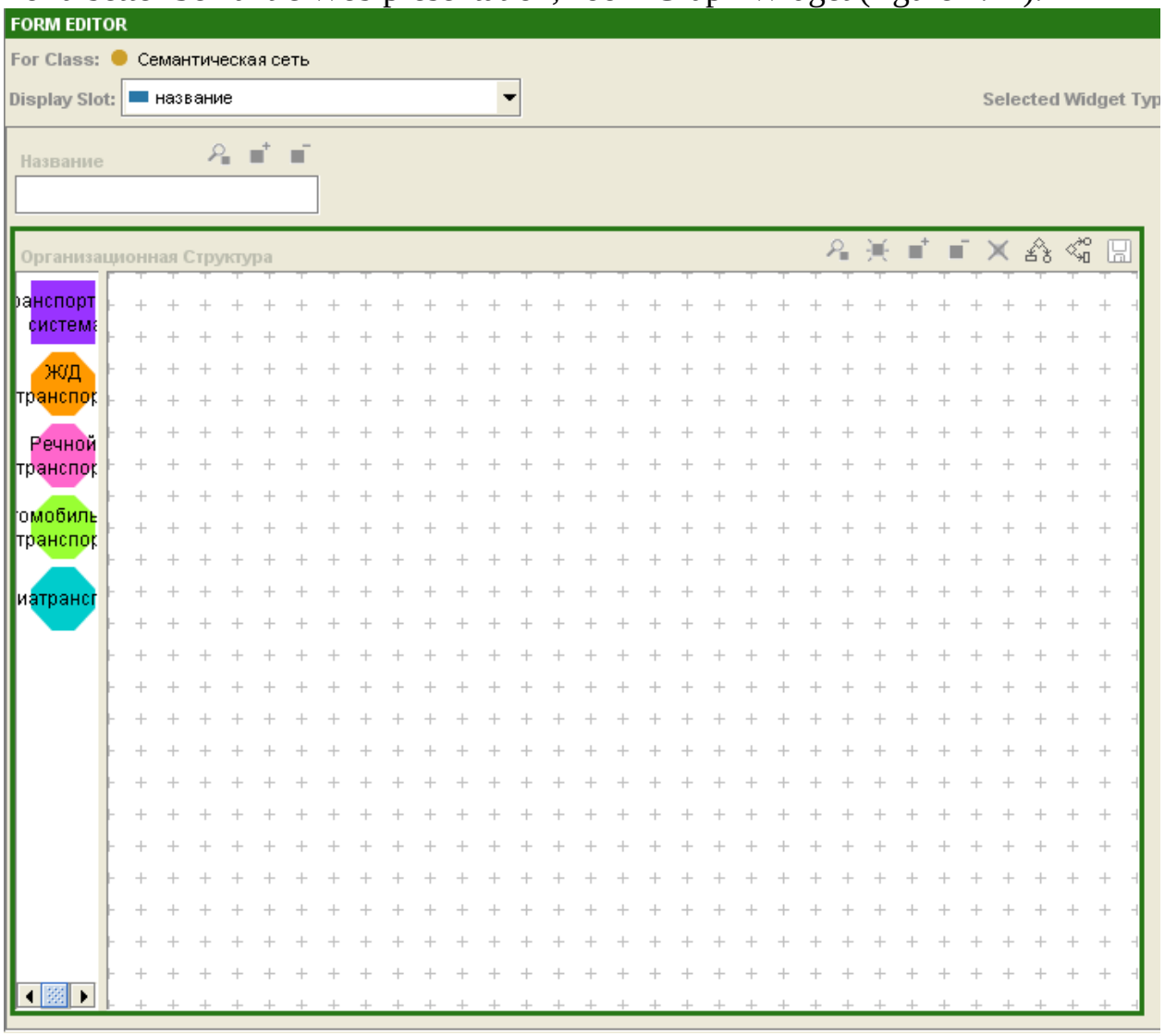

Figure 1.44. Modifying the size of the widget

1.1.2.16. Creating semantic representation of an ontology

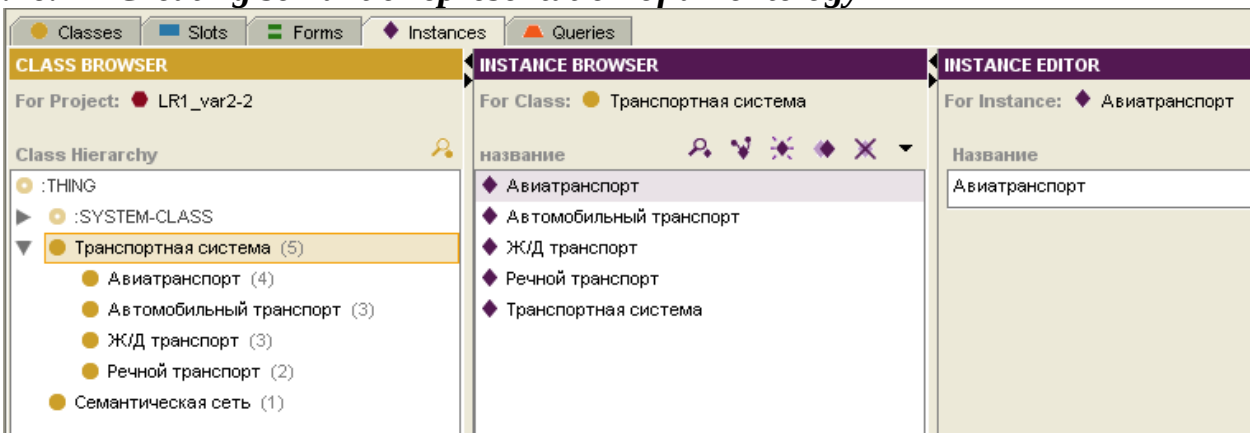

Figure 1.45. Creating instances for the
“Транспортная система” (Transport system) class

Navigate to the Instances Tab. Create the following instances for the “Транспортная система” (Transport system) class: “Транспортная система” (Transport system), “Авиатранспорт” (Air transport), “Автомобильный транспорт”

(Motor transport), “Ж/Д транспорт” (Rail transport), and “Речной транспорт” (River transport) (figure 1.45).

To create a semantic web, select the “Семантическая сеть” (Semantic Web) class in the Class Browser. Create an instance with a name “Транспортная система” (Transport system) for this class (figure 1.46).

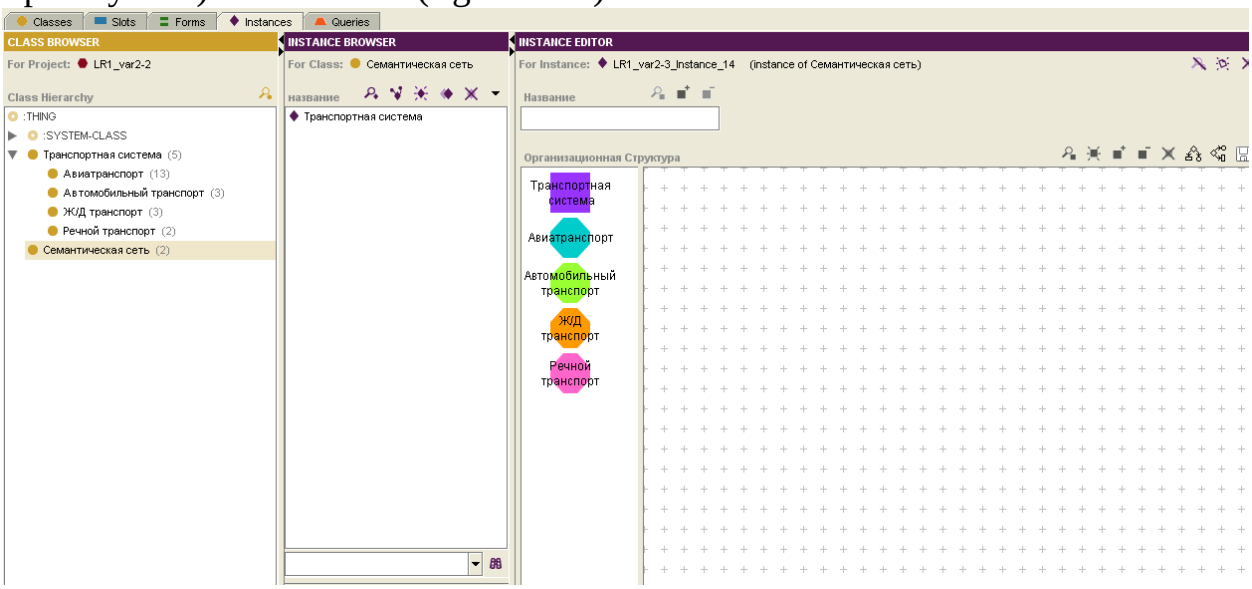

Figure 1.46. Creating an instance for the “Семантическая сеть” (Semantic Web) class

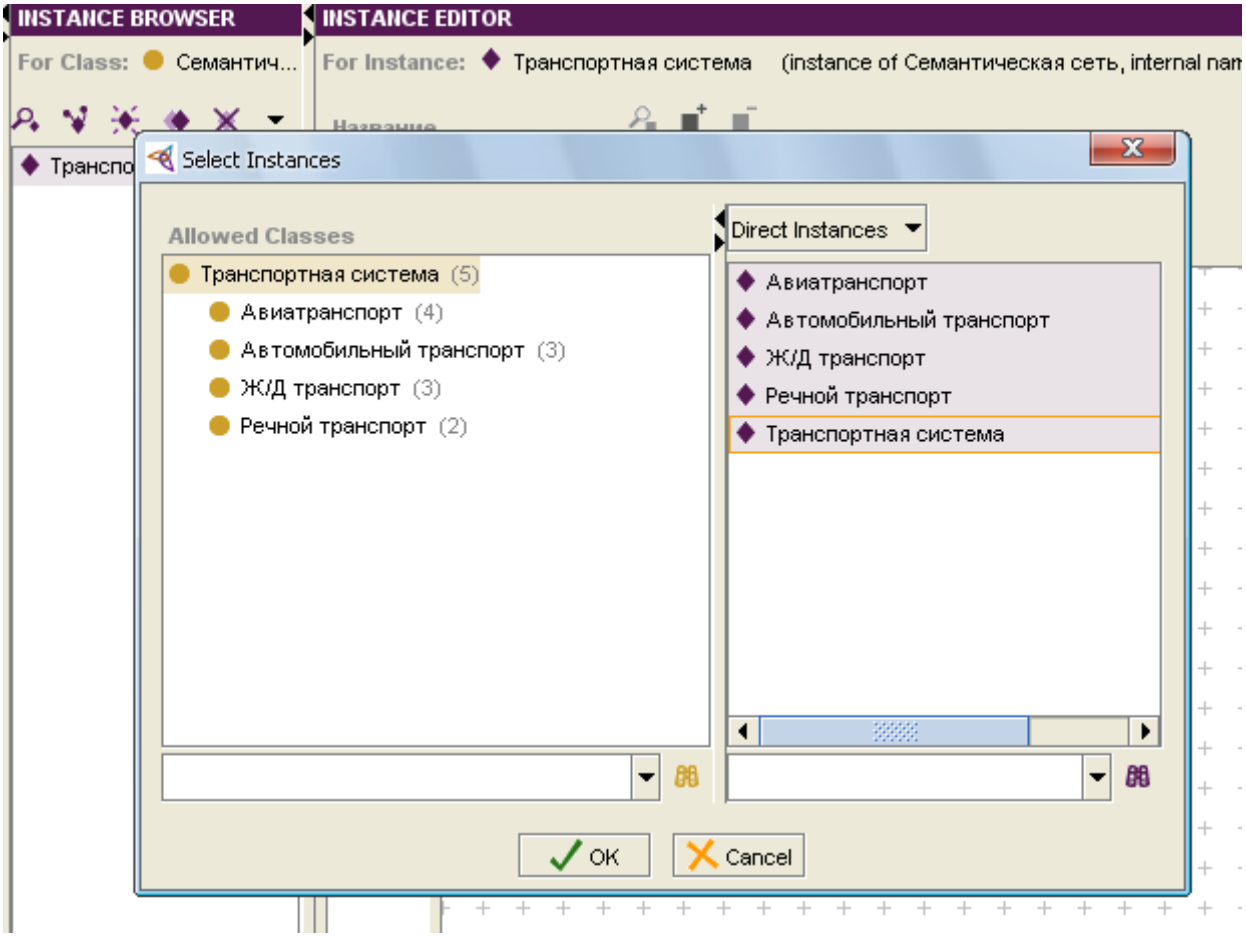

Figure 1.47. Selecting instances

Then, in the Instance Editor pane the “Организационная структура” (Organisational structure) dialogue will appear. Click the Add Existing Instance button in the right upper corner. In the new dialogue select the “Транспортная система” (Transport system) class on the left and on the right select all the instances of the class using the key combination Ctrl+A. Click OK (figure 1.47).

In the graph widget select the node labels and drag them as it is shown in figure 1.48. Change the size of the node labels in accordance with the size of the text.

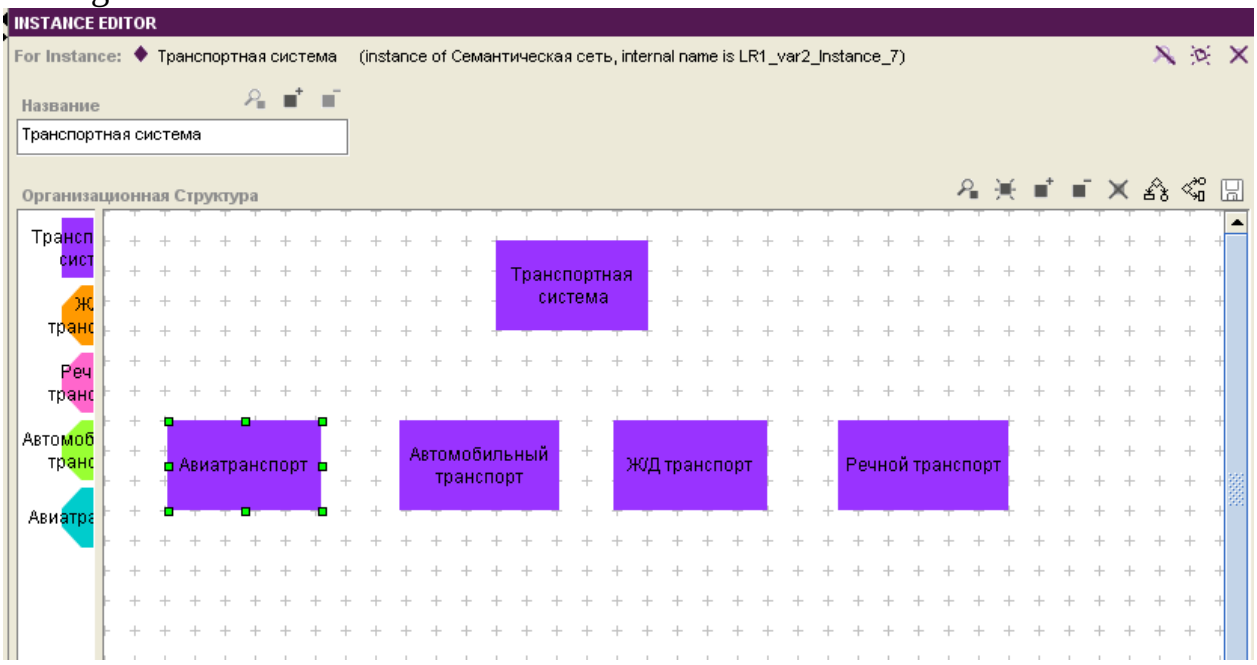

Figure 1.48. Nodes in the Graph Widget

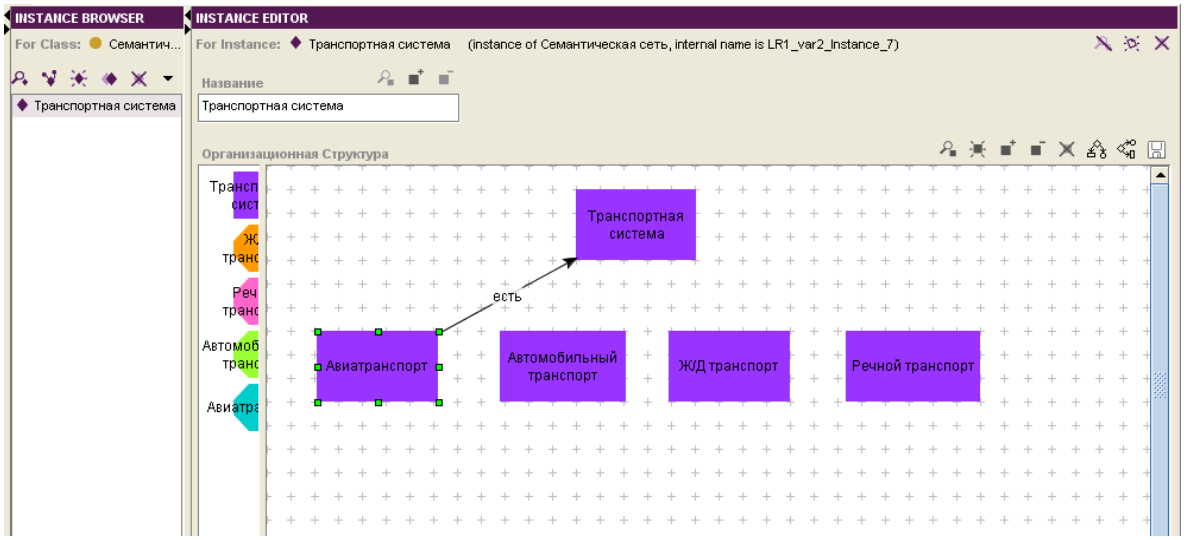

Figure 1.49. Creating a relationship between the instances “Авиатранспорт” (Air transport) and “Транспортная система” (Transport system)

To create relations between instances, select the “Авиатранспорт” (Air transport) label. When the mouse cursor takes the shape of a hand, press the node and select the “Транспортная система” (Transport system) node holding the left mouse button. When the line is created in the right direction, release the button. The “есть” (is) connector between the elements will be created, which literally means: “Авиатранспорт есть Транспортная система” (Air transport is Transport system) (figure 1.49).

Similarly, create connectors between “Автомобильный транспорт” (Motor transport), “Ж/Д транспорт” (Rail transport), “Речной транспорт” (River transport) and “Транспортная система” (Transport system) (figure 1.50).

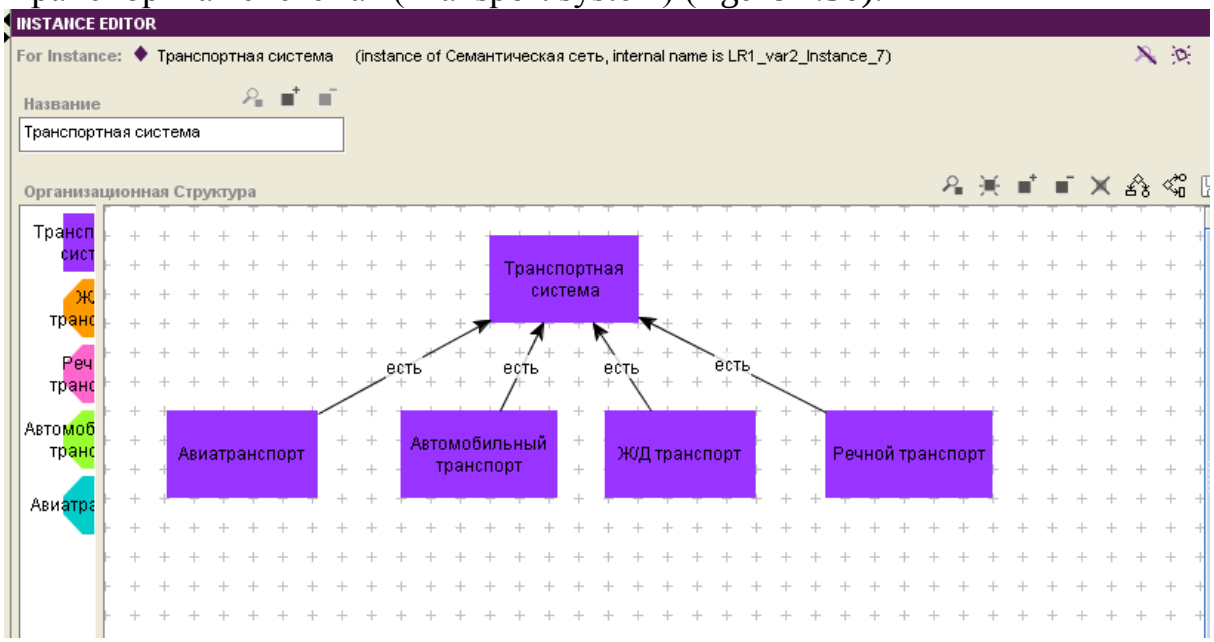

Figure 1.50. Creating connectors among “Автомобильный транспорт” (Motor transport), “Ж/Д транспорт” (Rail transport), “Речной транспорт” (River transport) and “Транспортная система” (Transport system)

The next step requires to include the instances “Авиатранспорт” (Air transport), “Автомобильный транспорт” (Motor transport), “Ж/Д транспорт” (Rail transport), “Речной транспорт” (River transport) into the semantic web. This can be done using the previously described method. The semantic web should be presented as shown on figure 1.51.

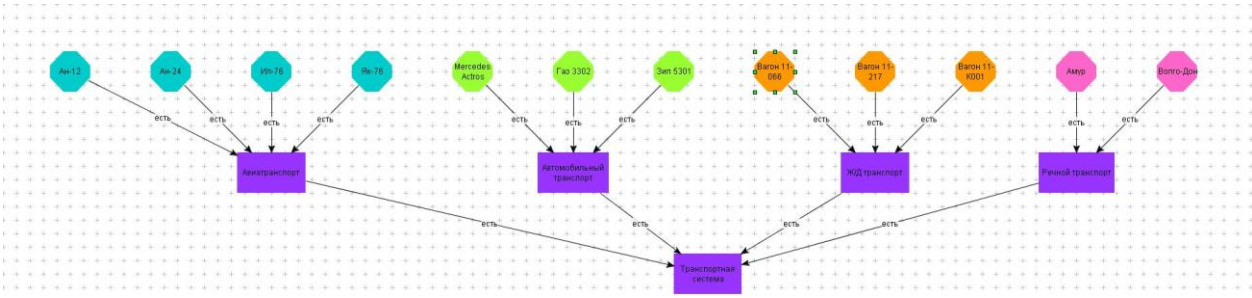

Figure 1.51. Semantic Web

1.1.2.17. Creating a query

When the ontology is created a vehicle should be selected according to the set parameters. Consider the creation of a query on the following example (Table1.2).

Table 1.2. Characteristics for creating a query

Cargo weight (kg)	Cargo capacity (m3)	Transportation time (h)	Season for transportation	Maximum cost (rub/kg)
3 500	200	10-60	winter	62

Moscow and Samara are the points of departure and destination respectively. The cost of transportation is the priority parameter. Therefore, “Стоимость” (Cost) should be the last criterion of the query.

Click on the Queries Tab. Click the Select Class button above the Class text box. Select the “Транспортная система” (Transport system) class in the new dialogue box (all its subclasses will be also included in the query). Click OK (figure 1.52).

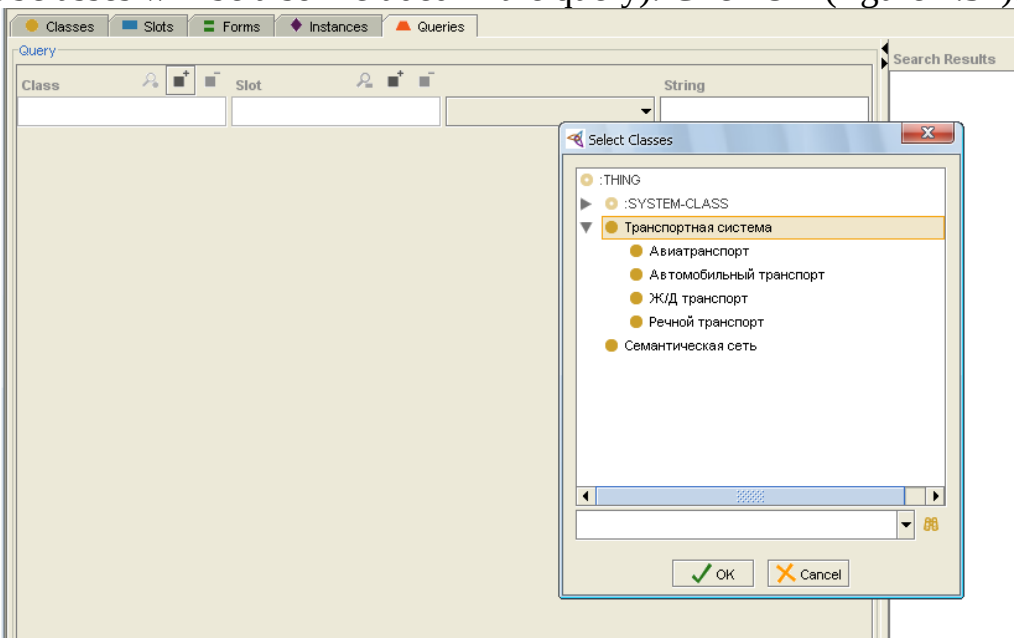

Figure 1.52. Selecting a class in Queries

Click the Select Slot button. Select “грузоподъемность” (Loading Capacity) in the new box (figure 1.53) and click OK.

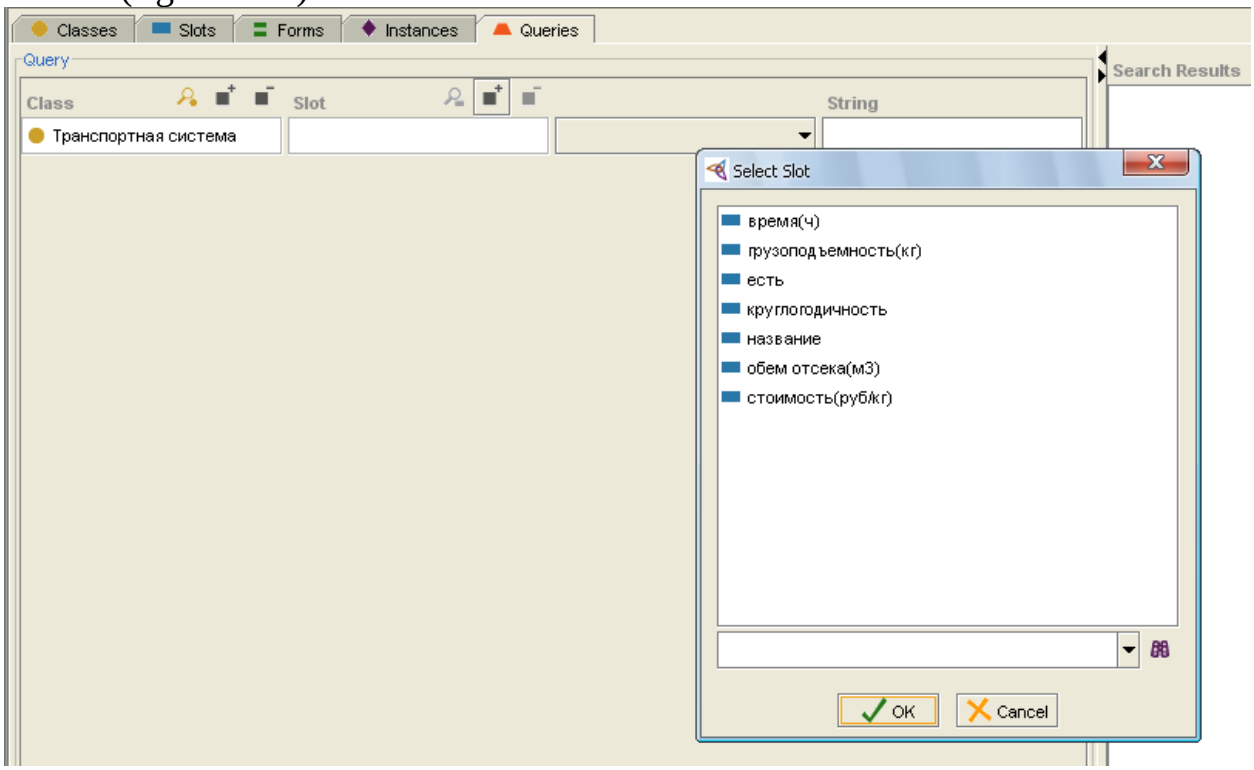

Figure 1.53. Selecting a slot

The menu to the right of the Slot text box is now active, and a choice can be made in accordance with the slot type. In our case, the Float type is a floating point number (figure 1.54).

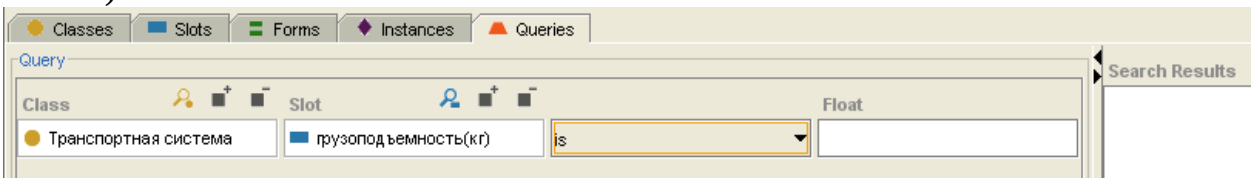

Figure 1.54. Queries Tab

In the drop down menu you can choose:

- Is greater than;
- Is less than;
- Is.

Select “is greater than” and enter “3500” in the Float text box (figure 1.55).

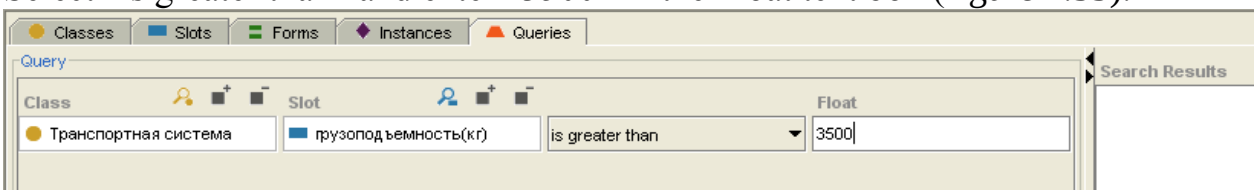

Figure 1.55. Selecting query parameters

To view the results of the query click the Find button. In the Search Results pane a list of vehicles that are able to transport more than 3500 kg will appear (figure 1.56).

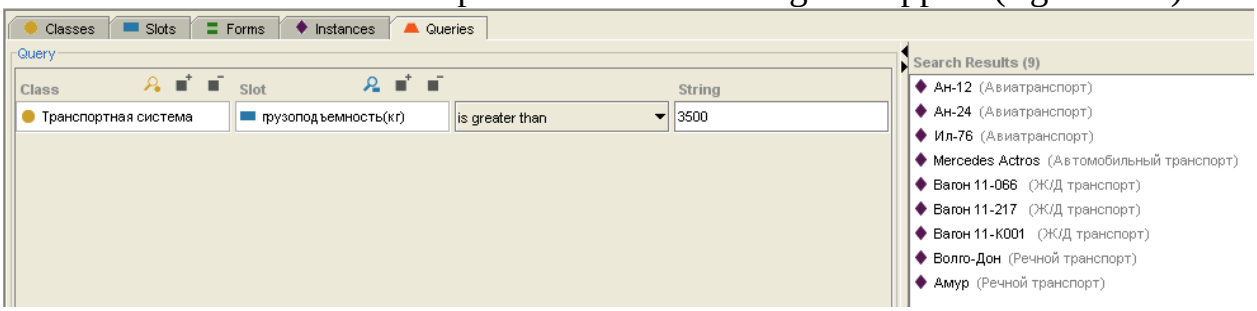

Figure 1.56. The Search results by weight

Then, click the More button to create a query of several criteria. An additional query bar will be displayed (figure 1.57).

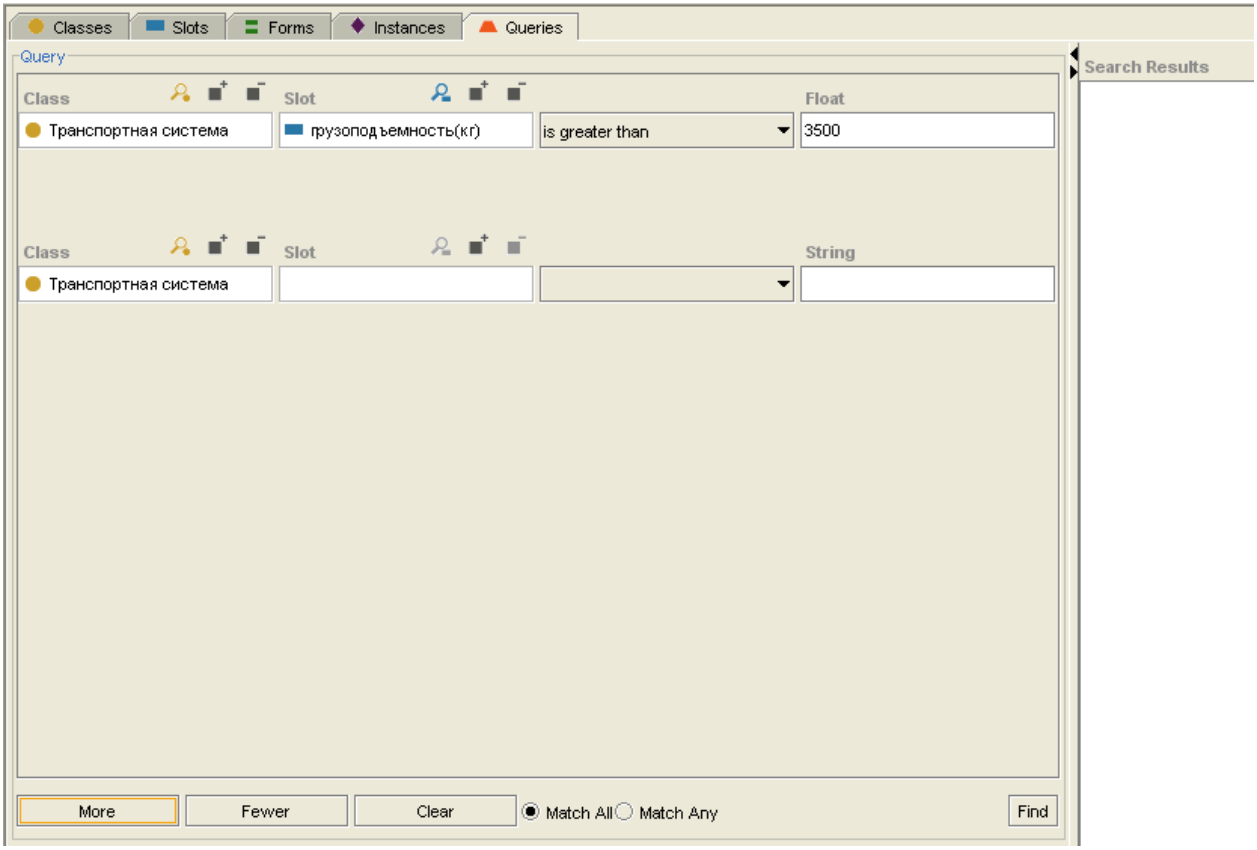

Figure 1.57. Adding a new query bar

Select the “Круглогодичность” (Year-round) slot. This is a String type slot, so select “Contains” in the down drop box and type “зима” (winter) using thy keyboard. The search result will be changed: the vehicles that do not operate in winters will be excluded (figure 1.58).

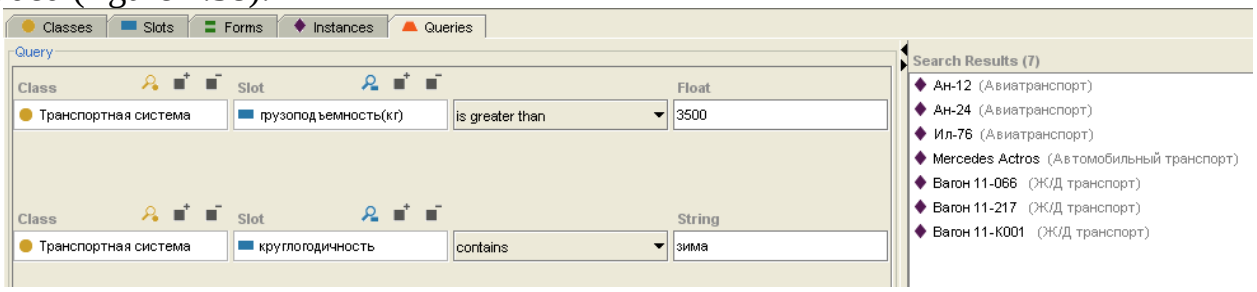

Figure 1.58. Adding query parameters

Select the “Время” (time) slot and set the value “is greater than 10”. Add another query bar with the value “is less than 60”. Press the Find button. The search result is reduced from 7 to 4 instances, because the vehicles that spend more than 60 hours on transportation are excluded (figure 1.59).

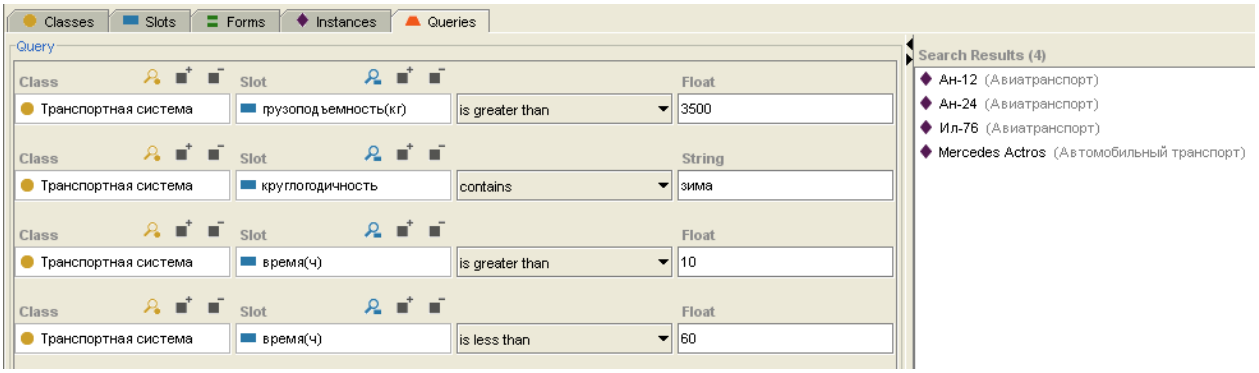

Figure 1.59. The search result taking into account transportation time

Select the “Объем отсека” (Cargo capacity) slot and set capacity limits. Click the Find button. The query result is equal to zero, that means that none of the vehicles meeting the previous parameters is able to transport more than 200m3 of cargo (figure 1.60).

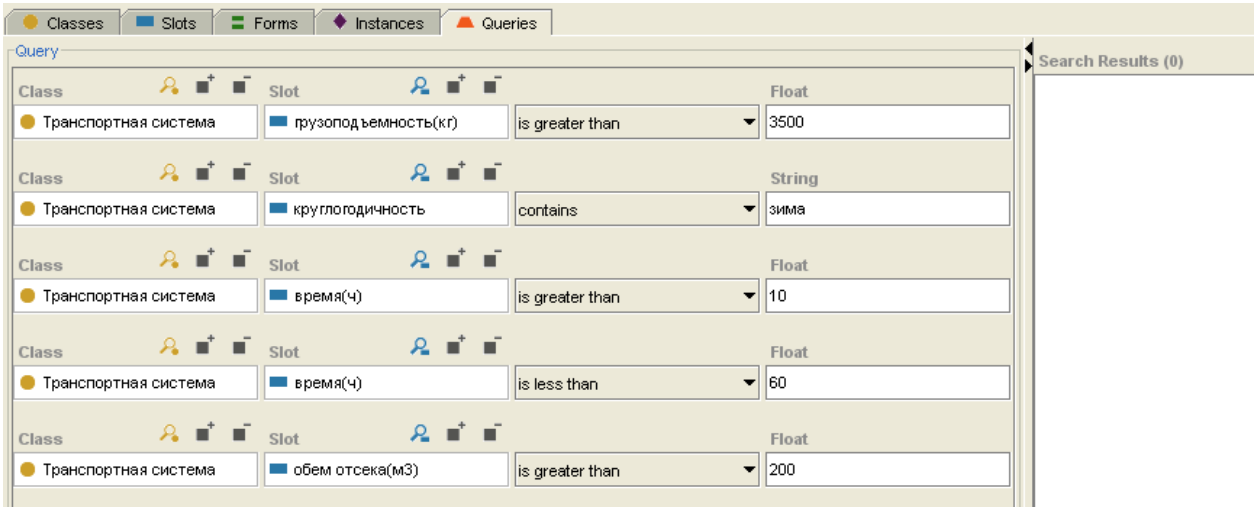

Figure 1.60. The search results taking into account the cargo capacity

In accordance with the task conditions (see 1.1.3. paragraph 4), the freight can be divided into two, three, or four parts. Divide it into two parts and change the data in the “грузоподъемность” (loading capacity) and “объем отсека” (cargo capacity) boxes. As a result, two vehicles, which are able to transport more than 100m3 freight, are displayed (figure 1.61).

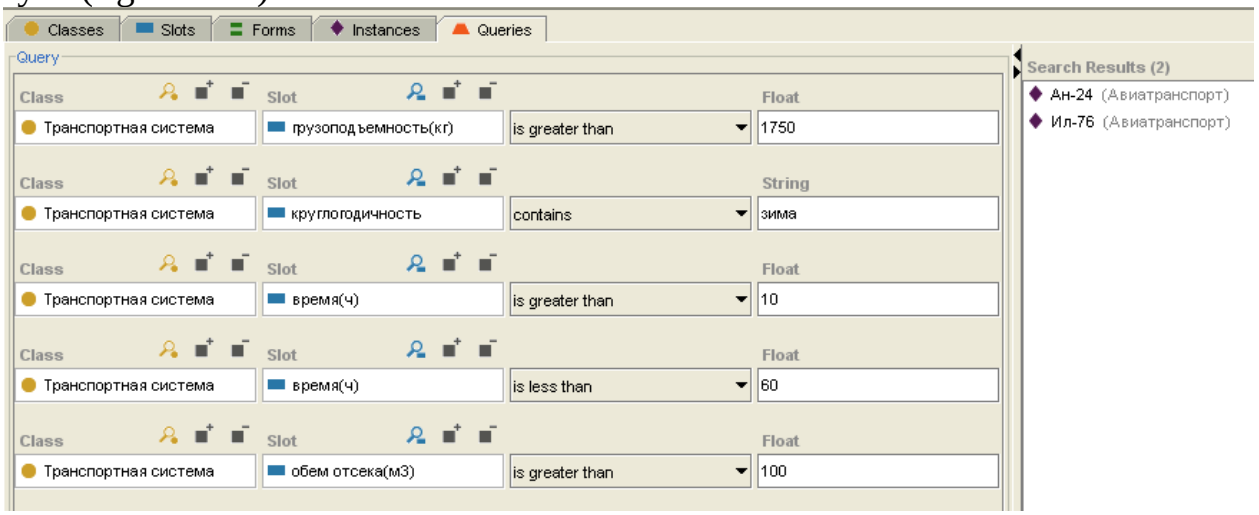

Figure 1.61. The result of the modified query

When it is known that the freight is divided into two parts and transported via air transport, include the cost parameter in the query. Type “is less than 62” in the Float box. The result has not changed, that is the two vehicles can transport freight of the set weight and capacity on the date and in accordance with the season (figure 1.62).

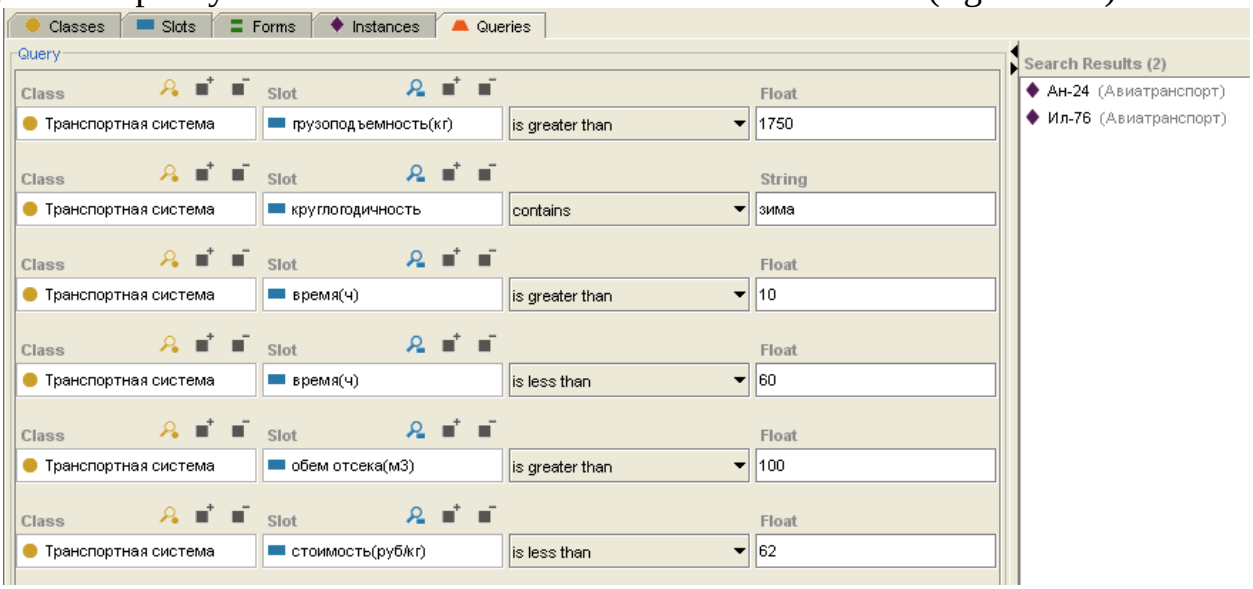

Figure 1.62. The final view of the query

The next step is to choose an optimal vehicle by cost.

To see the properties of each vehicle including the cost, double click one-by-one “Ан-24” (An-24) and “Ил-76” (Il-76) in the Search result dialogue box (figure 1.63).

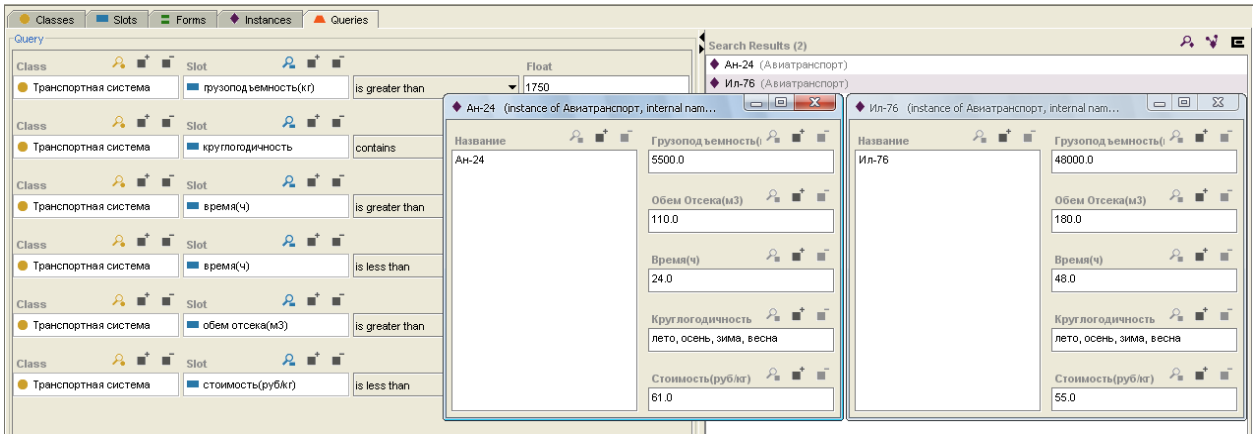

Figure 1.63. Instance Properties

Comparing the search results, the plane “Ил-76” (Il-76) may be chosen.

1.1.2.18. Saving a query

Protégé allows any query to be saved before or after it is executed. To save a query press Add to Query Library button, type “Запрос№1” (Query1) in the Input Query Name dialogue box. Click OK (figure 1.64).

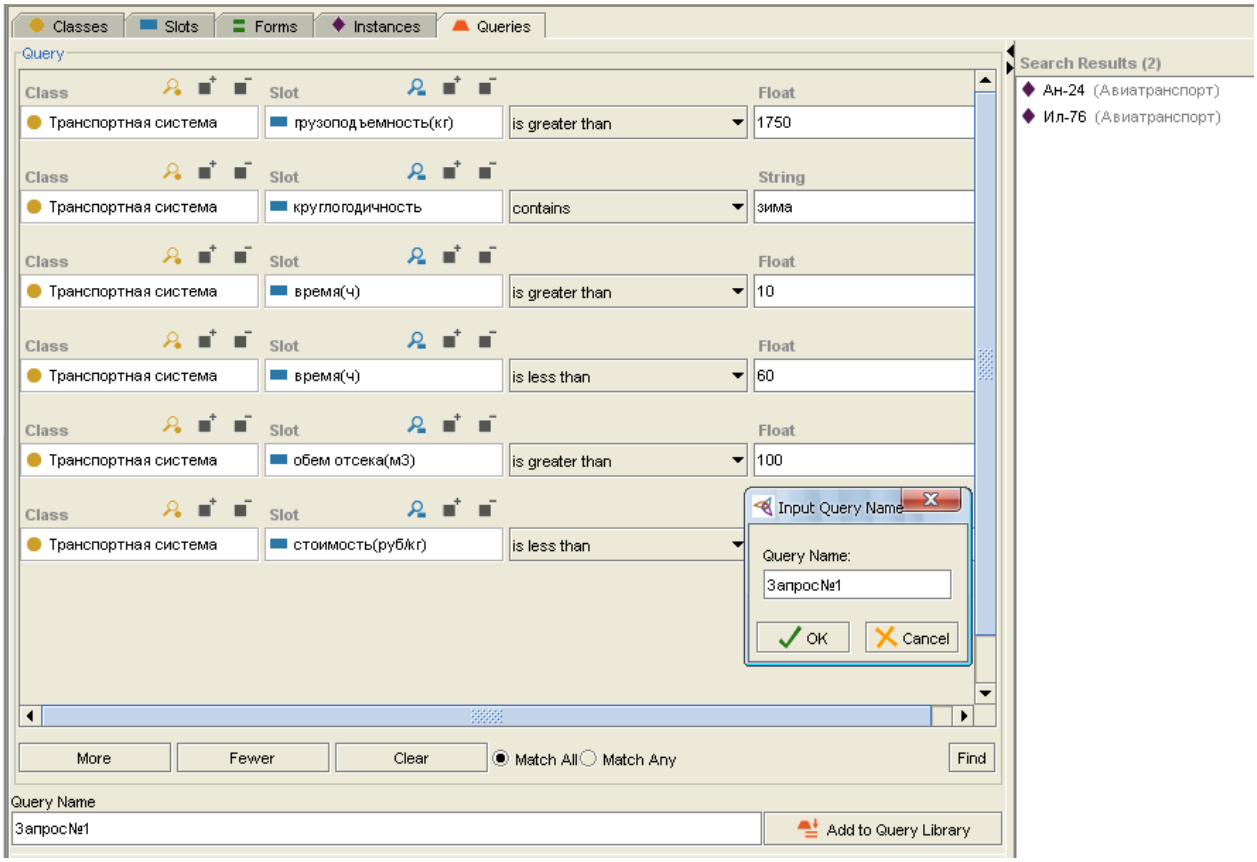

Figure 1.64. Saving a query

1.1.2.19. Retrieving a query

First click the Clear button to clear the query results. Select “Запрос№1” (Query1) in the Query Library and click the Retrieve Query button. The query parameters are displayed in the Query pane (figure 1.65).

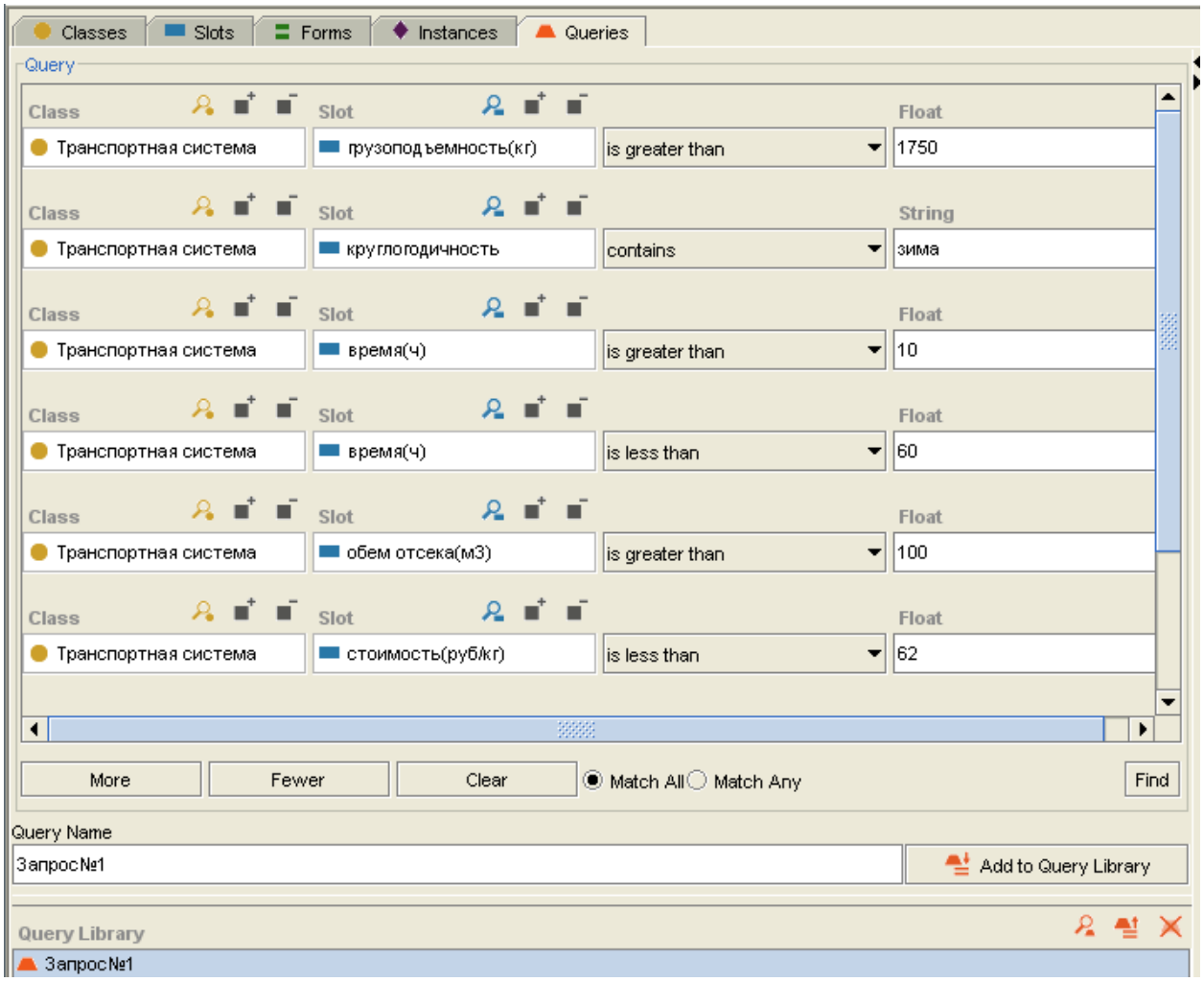

Figure 1.65. Retrieving a query

1.1.3. SELF-STUDY TASKS

1. Supplement the ontology with the aircraft model in accordance with the variant (Table 1.3);
2. Select a vehicle for freight transportation from one population centre to another with the defined parameters;
3. Consider Moscow as a departure point, and Samara as a destination point.
4. Take into account that the freight can be divided into two, three, or four parts while being transported.

Table 1.3. Task variants

NN	Aircrafts	Cargo weight (kg)	Cargo capacity (m3)	Transportation time (h)	Season for transportation	Maximal cost of transportation (rub/kg)
1	АН-26Д (An-26D) Lockheed C-5M SUPER GALAXY	10 000	45	30-70	winter	50
2	McDonnell Douglas C-17 C-80	1 000	100	10-30	summer	10
3	АН-124 РУСЛАН (An-124 Ruslan) Galaxy C-38	80 000	150	90-100	autumn	90
4	Lockheed C-5M SUPER GALAXY АН-26Д (An-26D)	57 000	80	50-100	spring	70
5	C-80 Boeing C-40	3 000	10	24	spring	66
6	АН-70 (An-70) Lockheed C-5M SUPER GALAXY	5 000	82	24-48	winter	15
7	Boeing C-18 Lockheed C-130J SUPER HERCULES	70 000	70	60-75	winter	82
8	Shorts C-23 SHERPA Boeing C-18	40 000	95	72	summer	53
9	ИЛ-76МФ (Il-76MF) Lockheed C-130J SUPER HERCULES	15 000	35	30	autumn	34
10	АН-124 РУСЛАН (An-124 Ruslan) Lockheed C-141C STARLIFTER	1 500	8	40	spring	56
11	Boeing C-32 Cessna U-27	25 000	100	80	winter	40
12	Cessna C-35 АН-124 РУСЛАН (An-124 Ruslan)	120 000	200	100-120	summer	67
13	МиГ Т-101 ГРАЧ (MiG T-101 Grach) Cessna UC-35	11 000	10	56	summer	12
14	АН-38Д (An-38D) Boeing C-32	38 000	9	56-75	autumn	15
15	Gulfstream C-37 АН-124 (An-124)	60 000	20	80-100	winter	65

1.2. Part 2. Selecting a plane

1.2.1. Ontology expansion

1.2.1.1. Loading a project

To load an already created project after you run the programme, select “LR1_varX” (X is your variant) in the welcome window and click the Open Recent button. This opens your project in the Class Browser (figure 1.66).

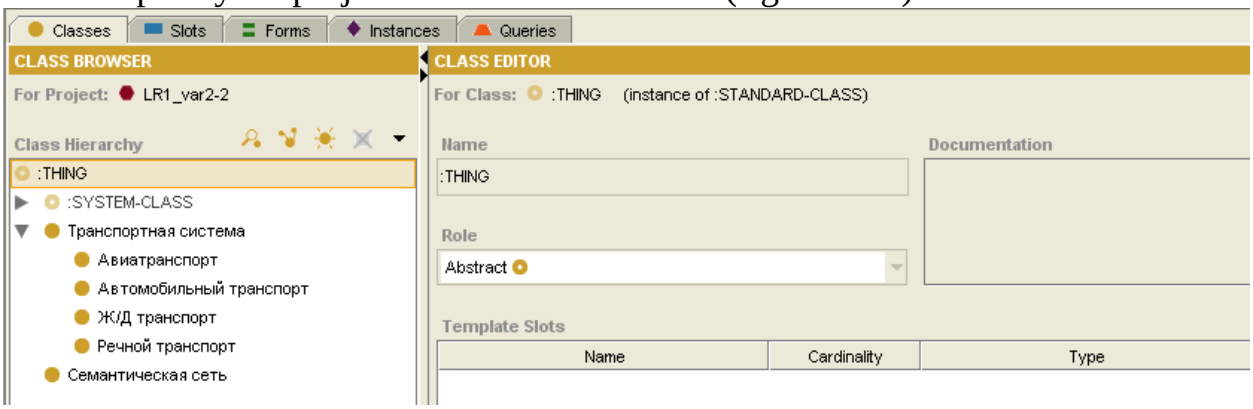

Figure 1.66. Loading a project

1.2.1.2. Creating additional slots

To add a slot, navigate to the Slots Tab and click the Create Slot button. Change the default name to “взлетная масса(кг)” (take-off weight), and in the Value type pane change String to Float. Put “0” in the Minimum text box (figure 1.67).

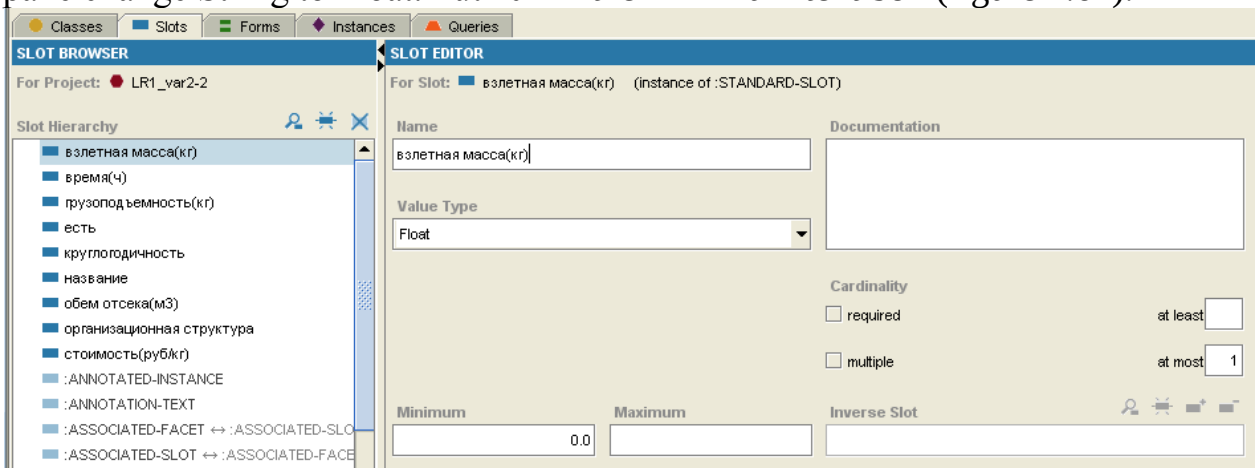

Figure 1.67. Creating the “взлетная масса” (take-off weight) slot

To include this slot into a class, click the Add Class button in the domain pane and select the “Авиатранспорт” (Air transport) class from the list. Click OK (figure 1.68).

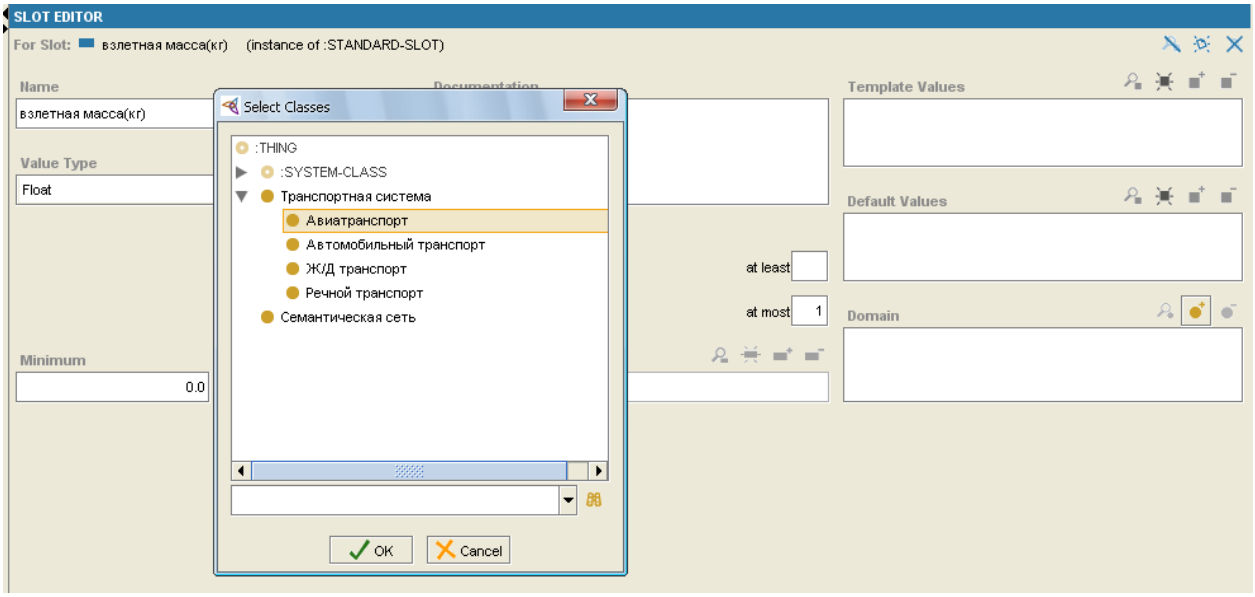

Figure 1.68. Including the “взлетная масса” (take-off weight) slot into the “Авиатранспорт” (Air transport) class

Similarly, create the “дальность полета(км)” (flight distance) and “количество пассажиров” (number of passengers) slots. Set the Float type, put “0” as a minimum value, and then in the “” (Air transport) class. Navigate to the Classes Tab and select the “Авиатранспорт” (Air transport) class. Check the new slots in the Template Slots pane. Notice, that these slots are included only in the “Авиатранспорт” (Air transport) class, so the icon is without parentheses (figure 1.69).

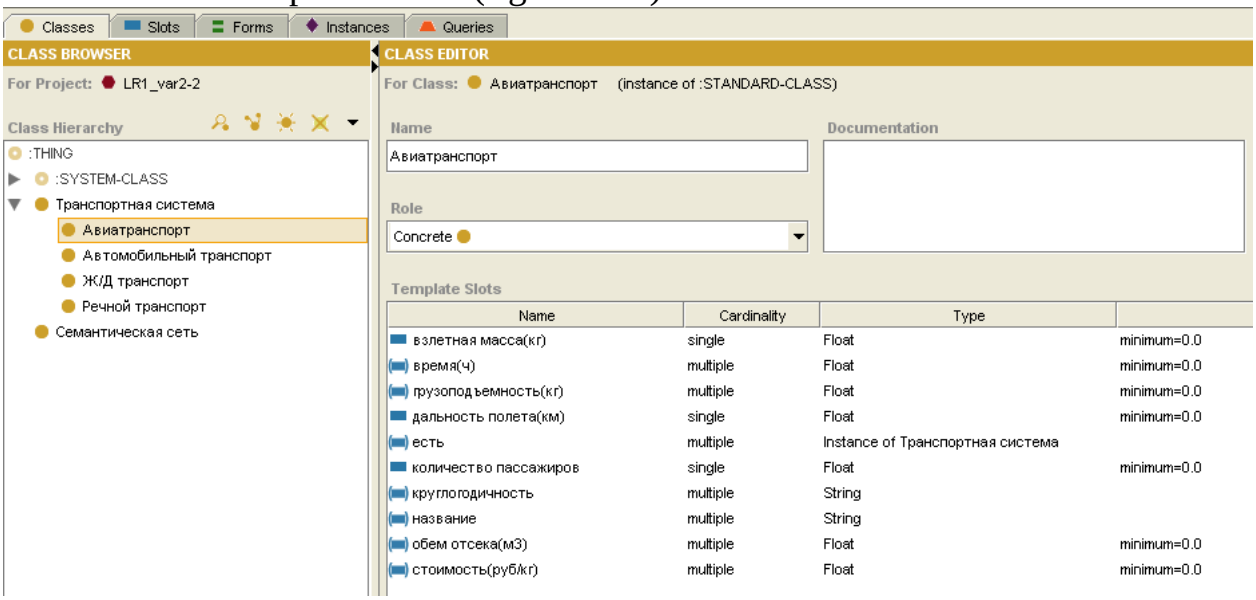

Figure 1.69. The “Авиатранспорт” (Air transport) class

1.2.1.3. Customising widget forms of created slots

Navigate to the Forms Tab and select the “Авиатранспорт” (Air transport) class. In the bottom of the pane the new slots “взлетная масса” (take-off weight), “дальность полета” (flight distance) and “количество пассажиров” (number of passengers) appeared in a default form (figure 1.70).

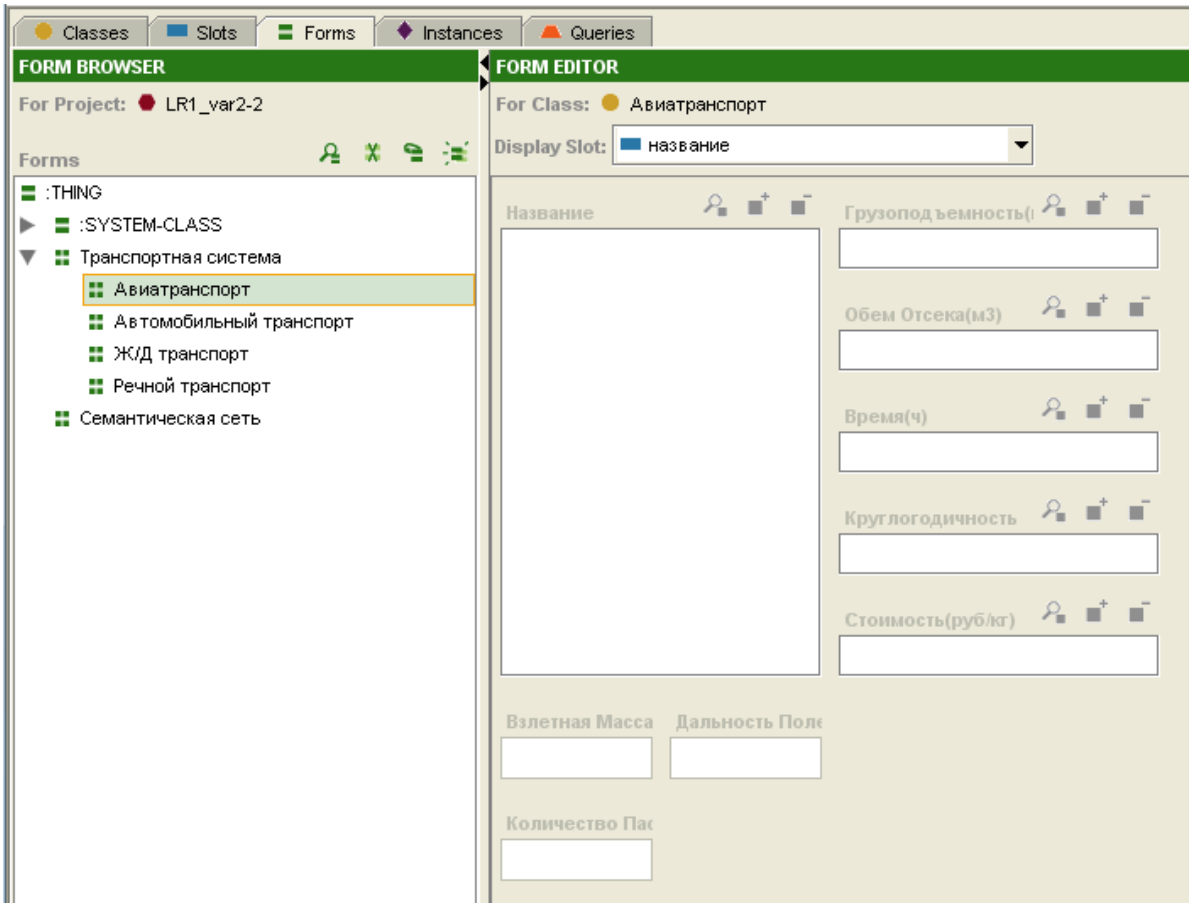

Figure 1.70. The “Авиатранспорт” (Air transport) class form

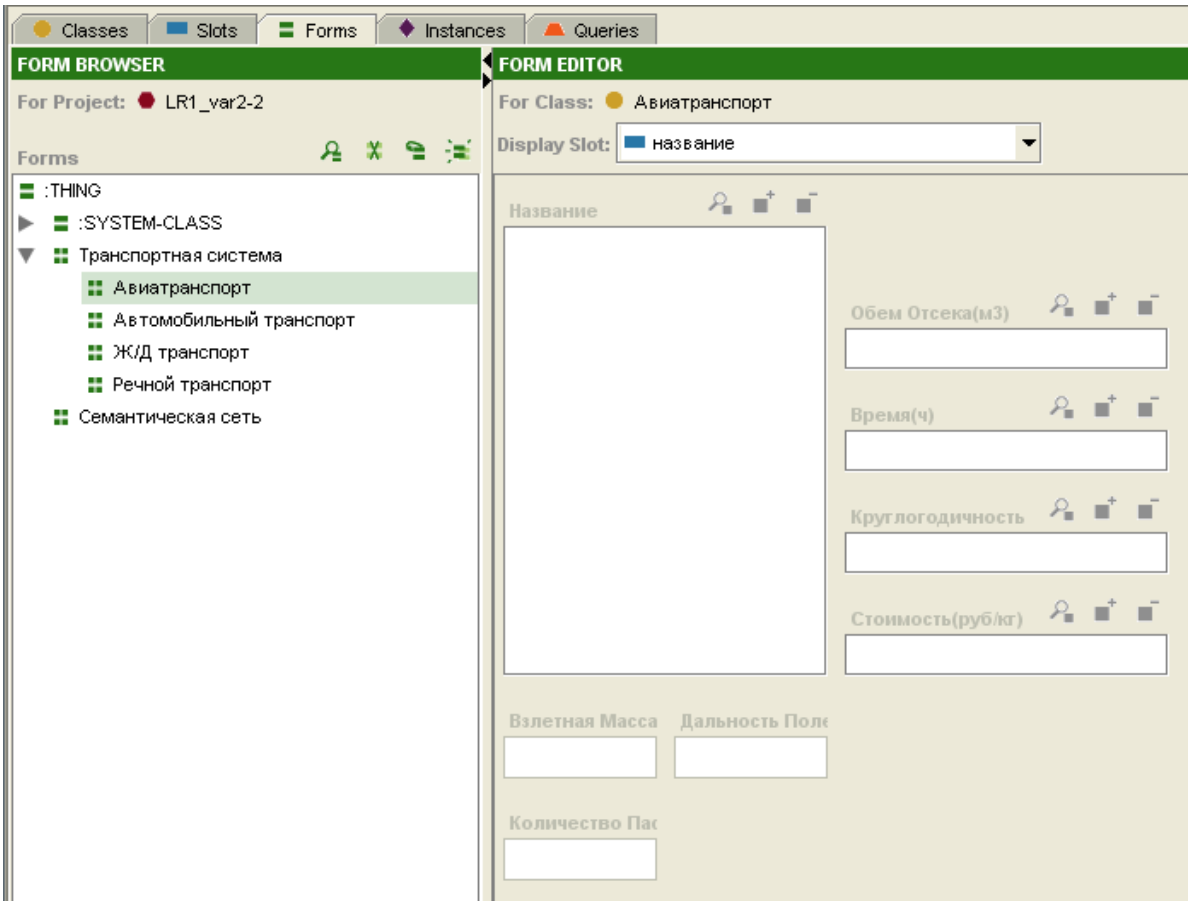

Figure 1.71. Customising the “грузоподъемность” (loading capacity) widget visibility

For this part of the lab you need the widgets of the “название” (name), “взлетная масса” (take-off weight), “дальность полета” (flight distance), and “количество пассажиров” (number of passengers) slots. The rest should be hidden.

To change the visibility of the slots, click on them (e.g. “грузоподъемность” (loading capacity) in the Form Editor pane. In the upper right corner select <none> in the Selected Widget Type box (figure 1.71).

Similarly, hide the widgets of the “объем отсека” (cargo capacity), “время” (time), “круглогодичность” (year-round), and “стоимость” (cost) (figure 1.72).

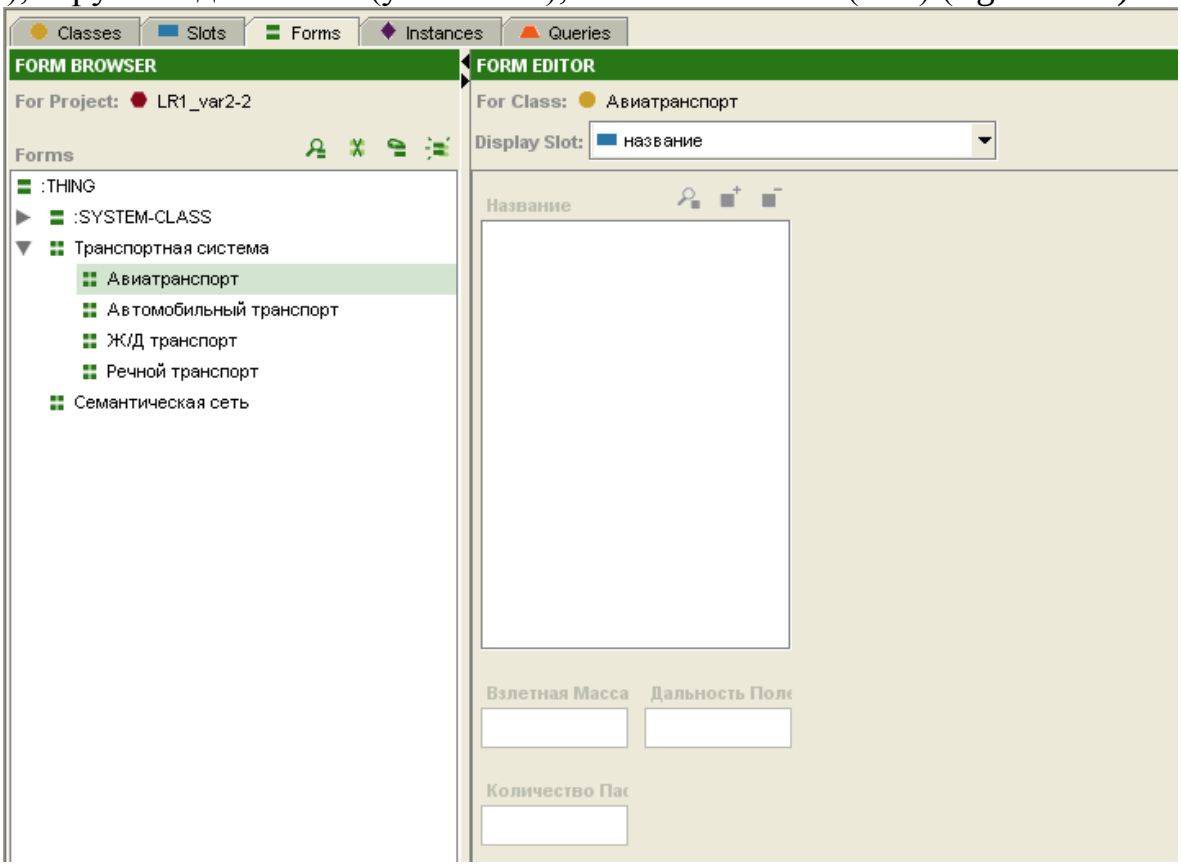

Figure1.72. Customising visibility of the “объем отсека” (cargo capacity), “время” (time), “круглогодичность” (year-round), and “стоимость” (cost) widgets

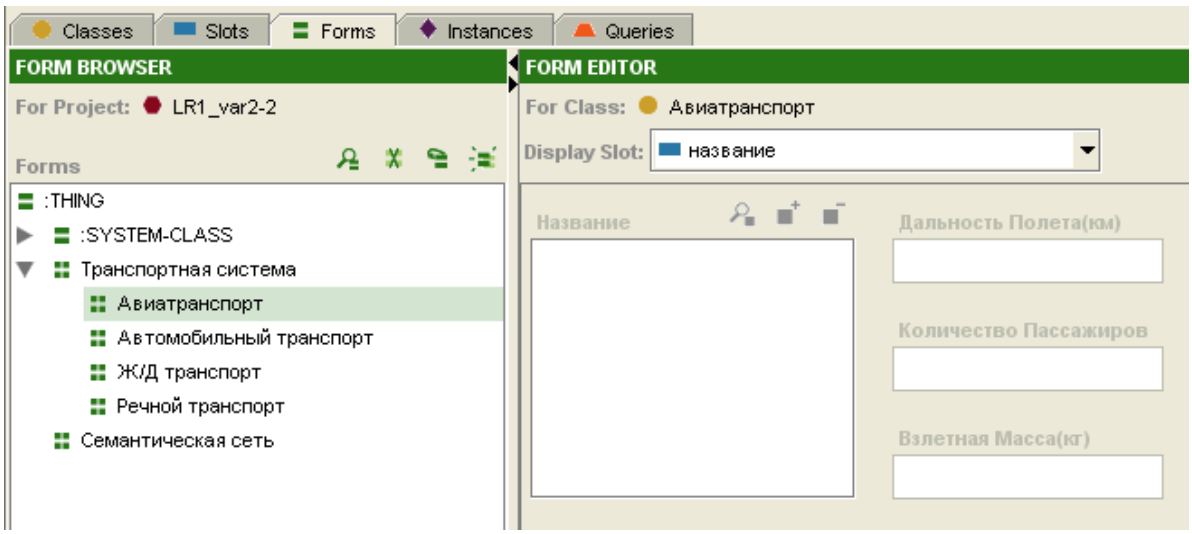

Figure 1.73. Customised Form Editor pane

Place the “взлетная масса” (take-off weight), “дальность полета” (flight distance), and “количество пассажиров” (number of passengers) widgets as it is shown in figure 1.73.

For more information about resizing and changing the place of widgets go to paragraph 1.1.2.12 (Laboratory work 1. Part 1).

1.2.1.4. Creating instances

Navigate to the Instances Tab and select the “Авиатранспорт” (Air transport) class. In the Instance Browser you will see previously created cargo aircrafts (figure 1.74).

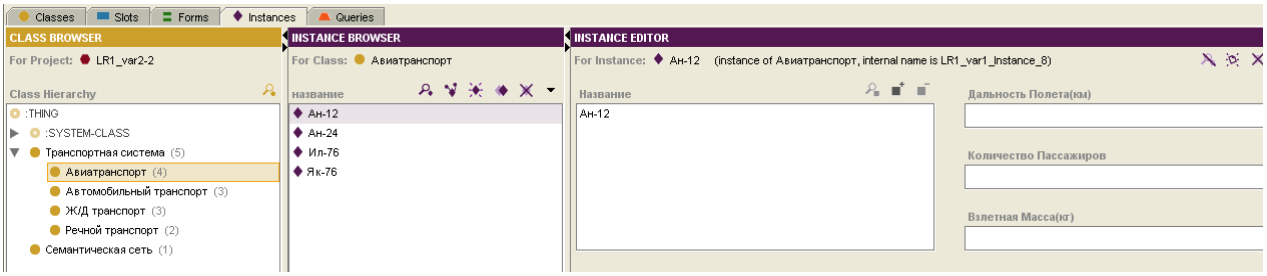

Figure 1.74. Instances of the “Авиатранспорт” (Air transport) class

To create new instances for the “Авиатранспорт” (Air transport) class click the Create Instances button. In the Instances Editor pane click the Add Value button near the “Название” (name) box. Type the name of the aircraft “Бритиш Аэро-спейс 146 - 300” (British Aerospace 146-300) in the Create String Value dialogue box. Click OK (figure 1.75).

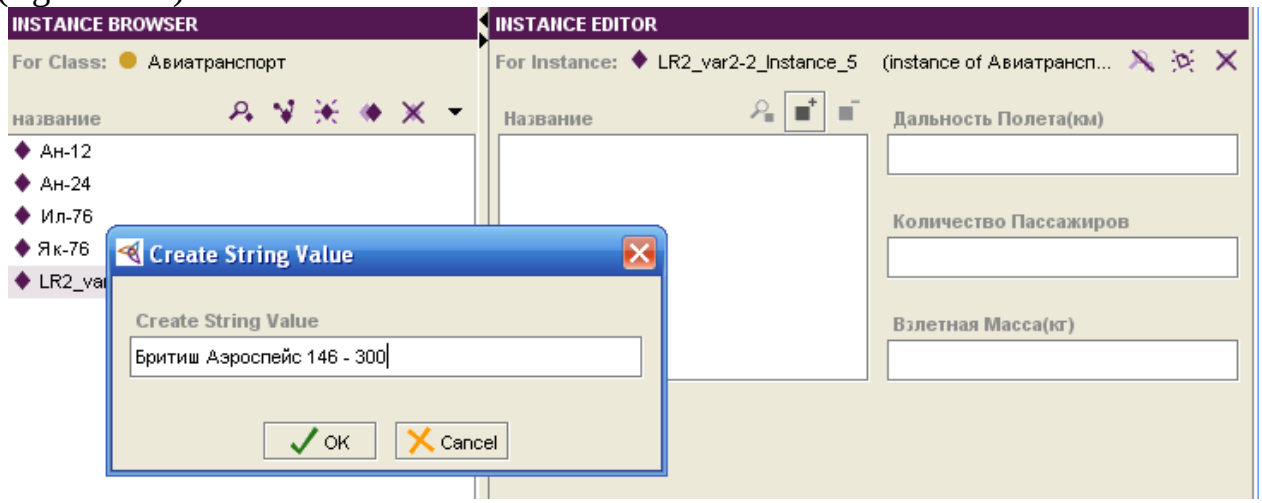

Figure 1.75. Creating a new instance

Similarly, using the button, fill the “взлетная масса” (take-off weight), “дальность полета” (flight distance), and “количество пассажиров” (number of passengers) data for the “Бритиш Аэро-спейс 146 - 300” (British Aerospace 146-300) instance (using Table 1.4) (figure 1.76).

Notice, that the data should be entered directly in the box, because, when they were created, the Multiple box was not marked.

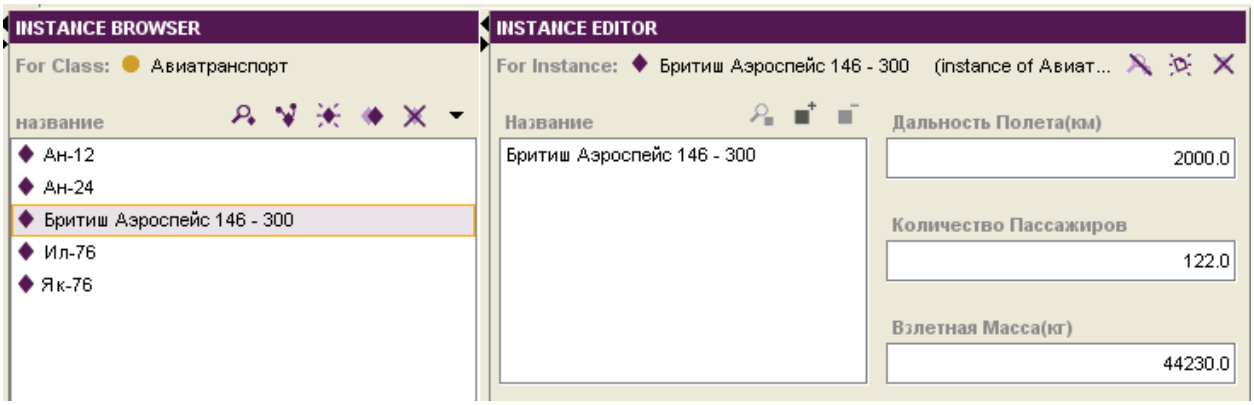

Figure 1.76. Data entry for the “Бритиш Аэро-спейс 146 - 300” (British Aerospace 146-300) instance

Similarly, create other instances using the data from Table 1.4 (figure 1.77).

Table 1.4. Instances characteristics

Aircraft	Flight distance, km	Number of passengers	Take-off weight, kg
British Aerospace 146 - 300	2 000	122	44 230
British Aerospace 146 - 200	2 180	109	42 180
British Aerospace 146 - 100	1 730	93	38 010
Fokker 70	2 000	79	36 740
Fokker 100	2 390	107	43 090
RJ100	2 260	100	46 000
Tu – 134B	2 020	96	47 600
Tu – 334 - 100	2 000	110	46 100
Yak – 42D	2 150	120	56 500

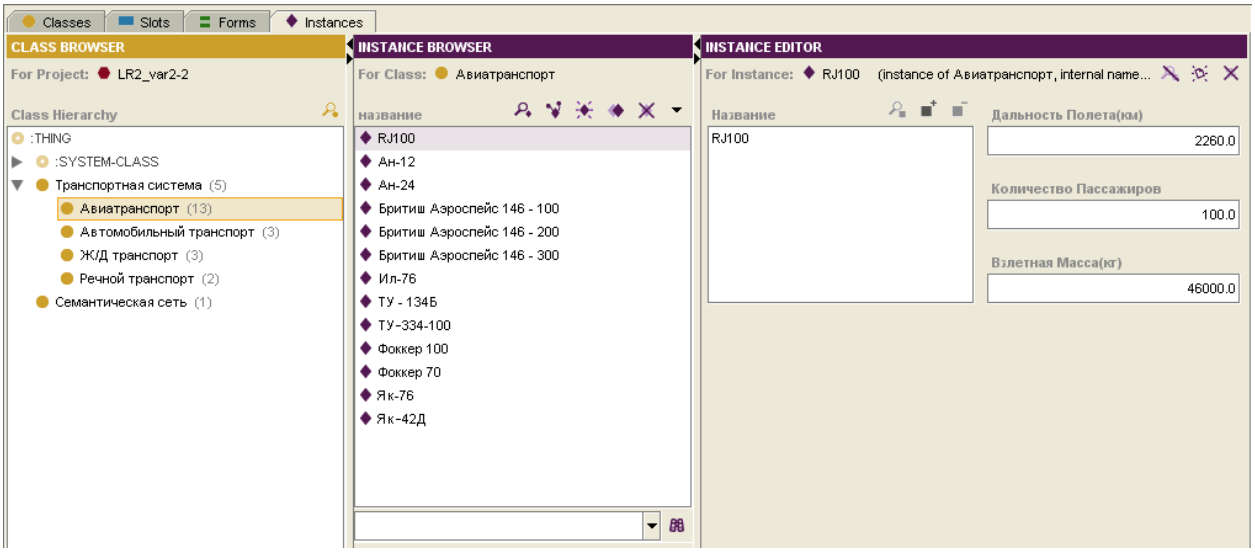

Figure 1.77. Instances of the “Авиатранспорт” (Air transport) class

1.2.1.5. Setting the display slot

In the first part of the laboratory work only the “название” (Name) display slot was selected. To select several display slots for the “Авиатранспорт” (Air transport) class select Set Display Slot from the pop-up menu of the Instance Browser and click Multiple Slots… (Figure 1.78).

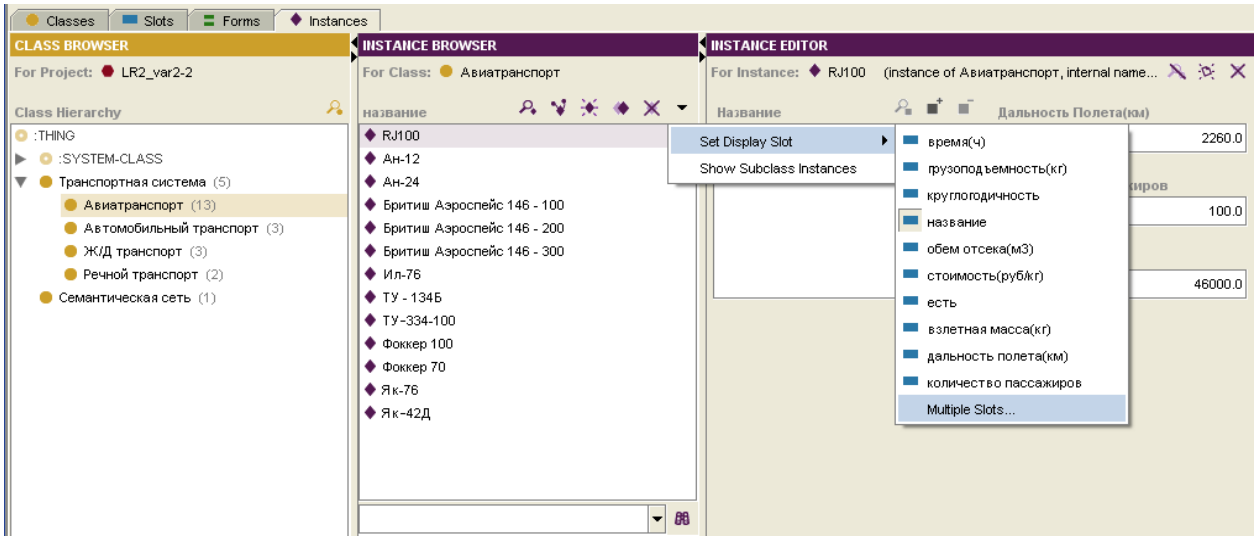

Figure 1.78. Setting several display slots

Then select the display slots in the Multislot Display Pattern dialogue box (figure 1.79).

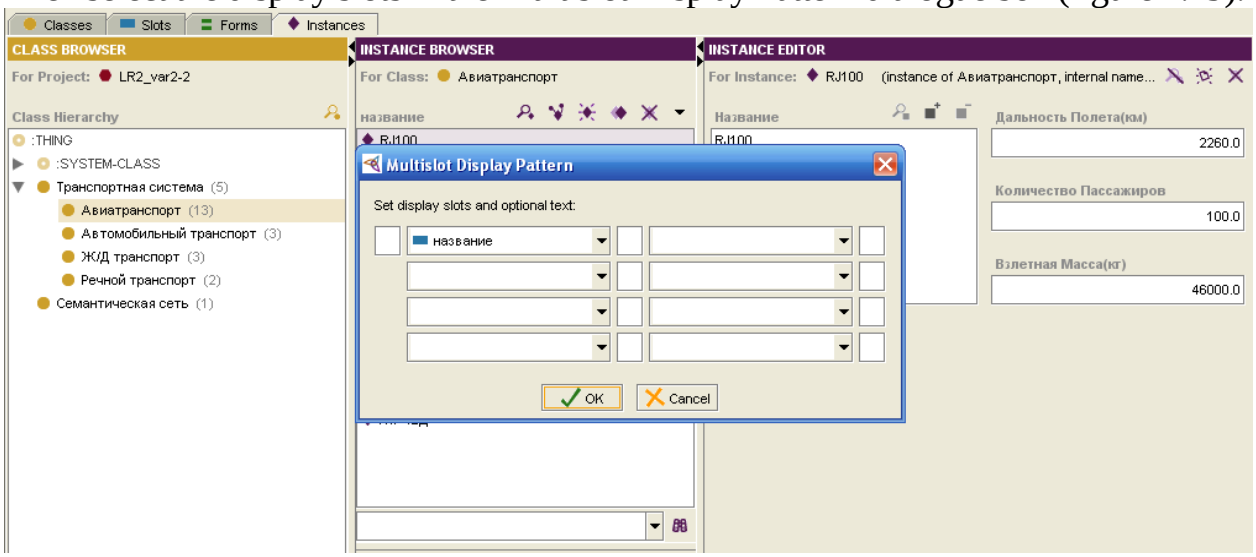

Figure 1.79. Multislot Display Pattern

Select the slots from the pop-up menu in the order as it is shown in figure 1.80. Click OK.

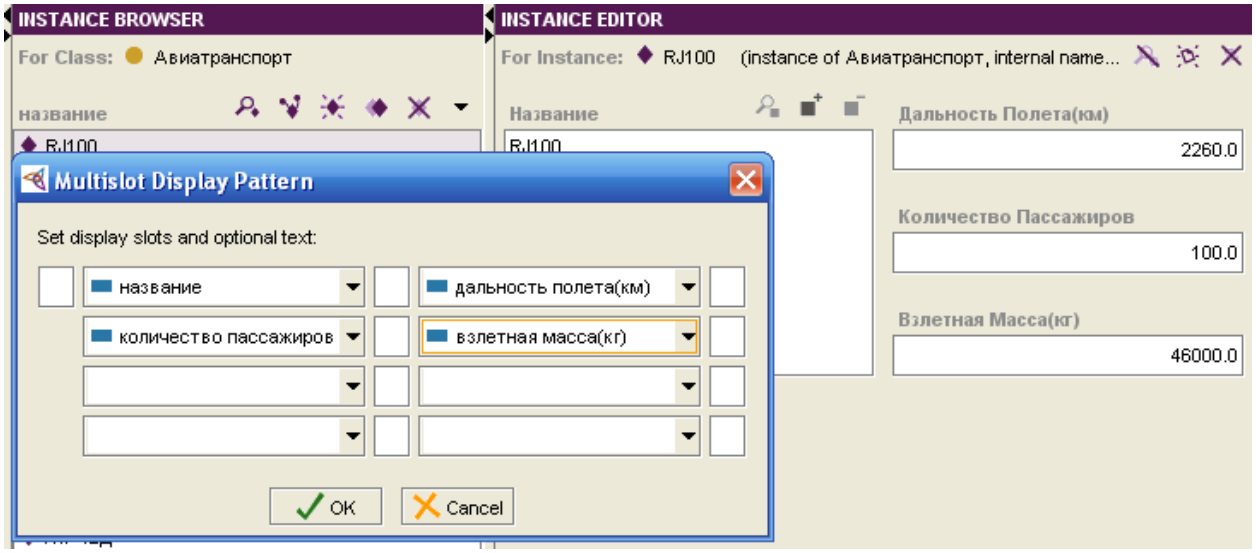

Figure 1.80. The order of selecting the slots

In the Instance Browser the slot values are displayed without spaces and names (figure 1.81).

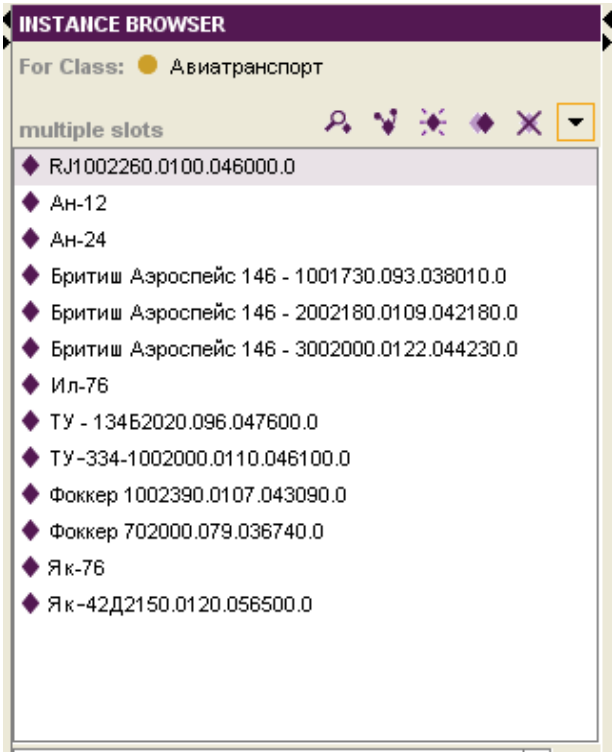

Figure 1.81. The slot values of the “Бритиш Аэро-спейс 146 - 100” (British Aerospace 146-300) instance

One more time open the Multislot Display Pattern dialogue box (Instance Browser -> Set Display Slot -> Multiple Slots…). In this box, there are boxes for comments. In the work they are used for entering the distance, quantity and weight values. Notice, that the Comment box is on the left of the slot, and for the second and the next boxes (e.g. “количество пассажиров” (number of passengers)) it is near the previous one. Enter the following comments for the slots (instead of the low dash use the same number of spaces, quotation marks are not needed):

- “название” (name) – “___”
- “дальность полета” (flight distance) – “___L_=_”
- “количество пассажиров” (number of passengers) – “___n_=_”
- “взлетная масса” (take-off weight) – “___m_=_”.

Compare the data with figure 1.82. Then click OK.

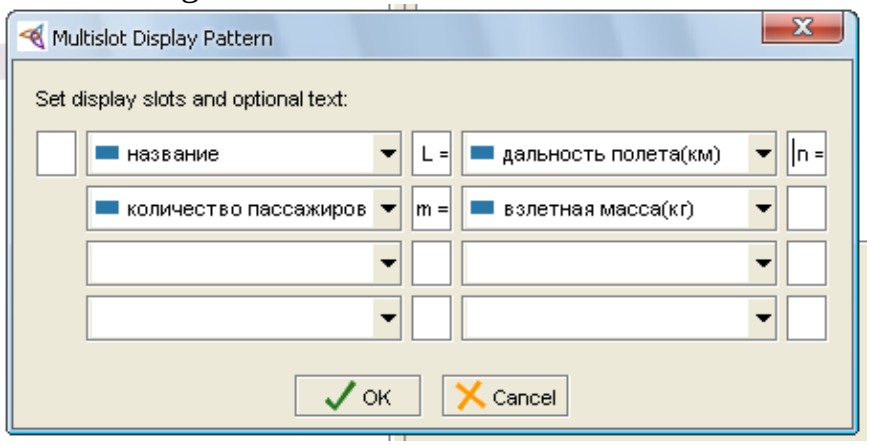

Figure 1.82. Comments entering for the slots

The data display has changed (figure 1.83).

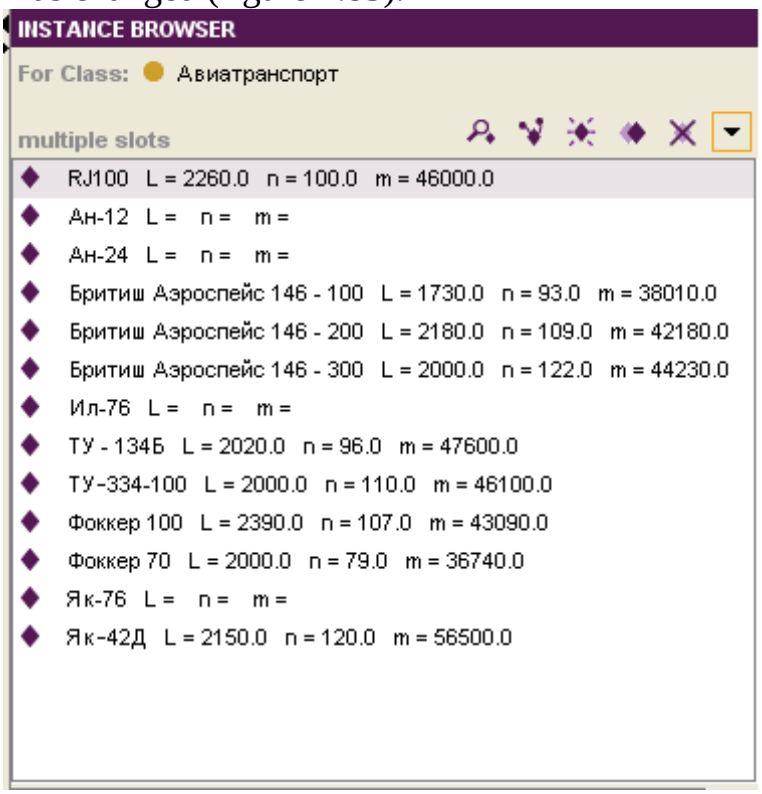

Figure 1.83. Customised Instance Browser

1.2.1.6. Creating a query

When the ontology is created it is required to select an aircraft taking into account deviation and weight minimisation. Consider creating a query (Table 1.5).

Table 1.5 – Data for the query

Flight distance, km	**Number of passengers**	**Deviation, %**	**Take-off weight, kg**
2 000	100	±10	min

To create a query, navigate to the Queries Tab. Click the Select Class button. In the new dialogue box select the “Авиатранспорт” (Air transport) class and click OK (dig. 1.84).

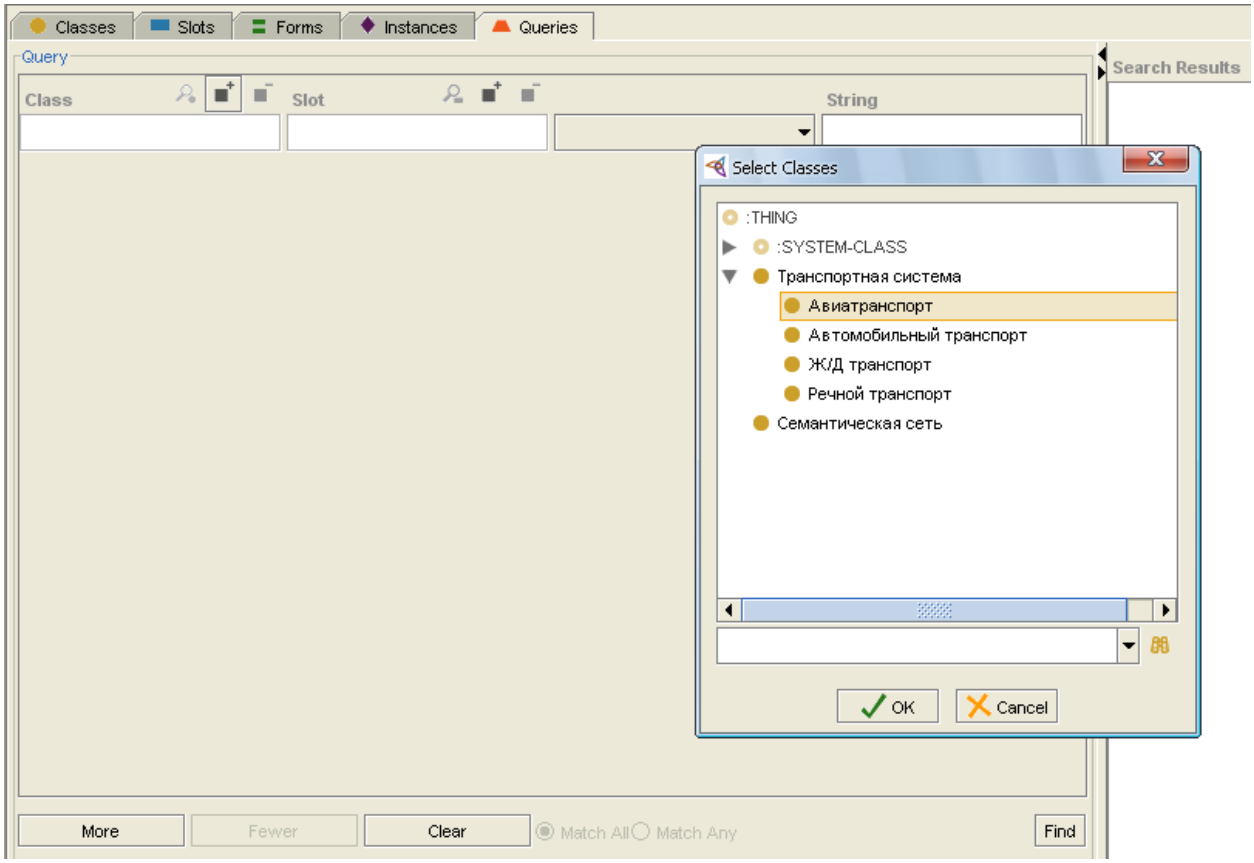

Figure 1.84. Selecting a class for a query

Click the Select Slot button. In the new dialogue box (figure 1.85) select “дальность полета” (flight distance) and click OK.

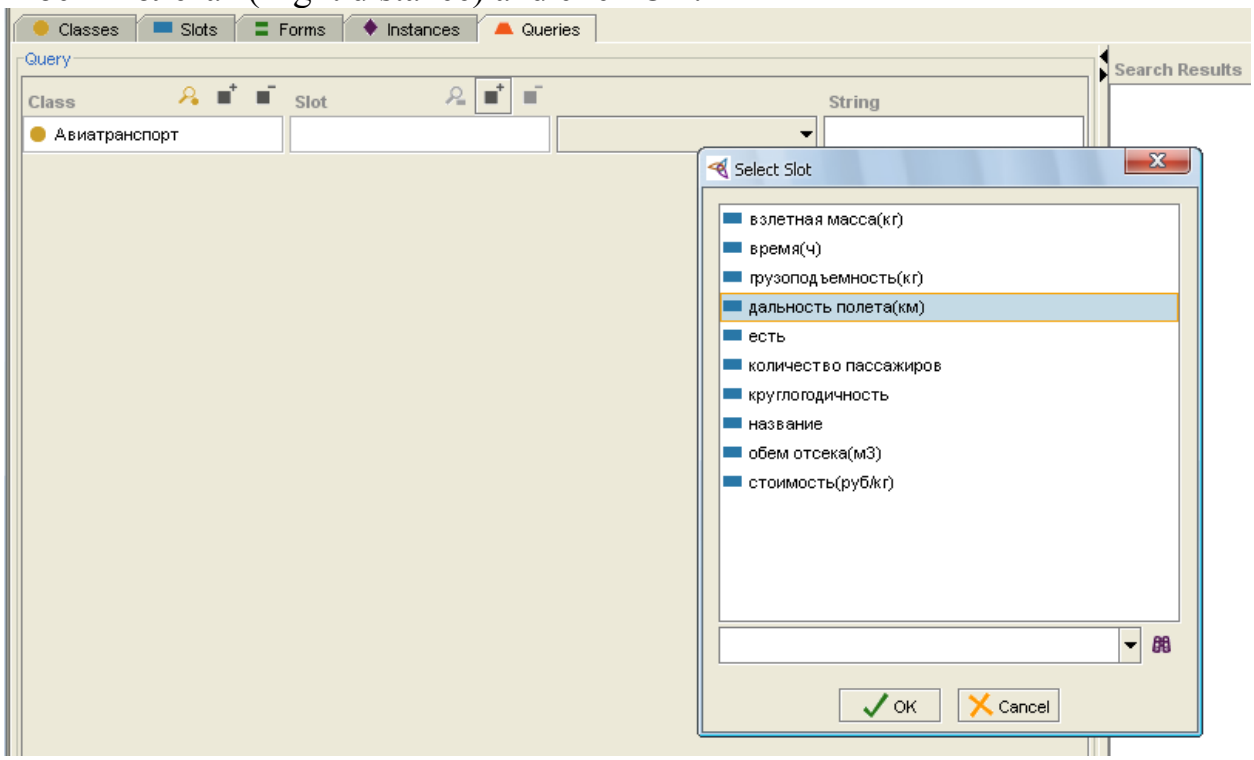

Figure 1.85. Selecting a slot for a query

When the slot is selected, the box on its right gets activated and a slot type can be selected. In our case, the Float type is a floating point number (figure 1.86).

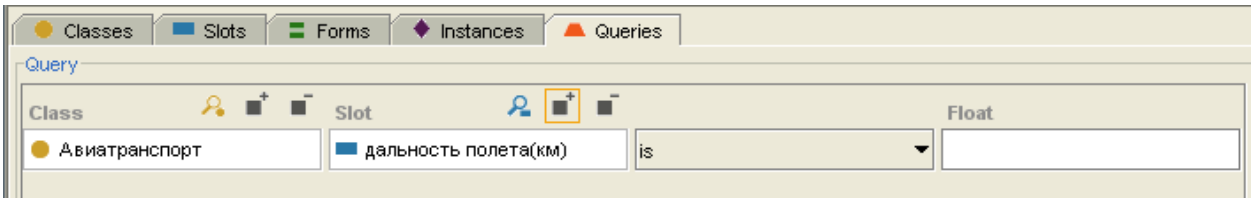

Figure 1.86. Queries Tab

In the drop down menu you can choose:

- Is greater than;
- Is less than;
- Is.

Select “is greater than” and enter “2000” in the Float text box (figure 1.87).

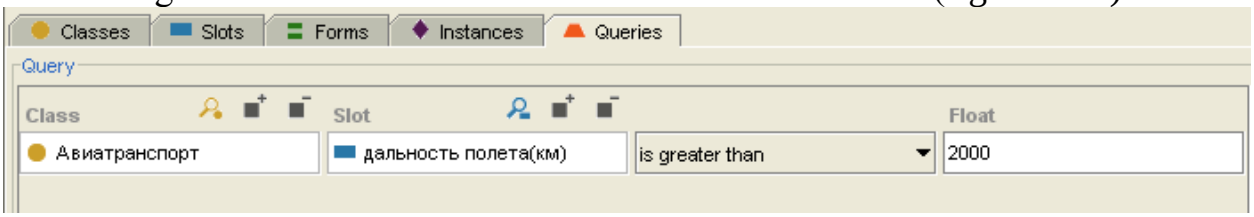

Figure 1.87. Selecting query parameters

To view the results of the query click the Find button. In the Search Results pane a list of vehicles that are able to fly for longer distances than 2000 appears (figure 1.88).

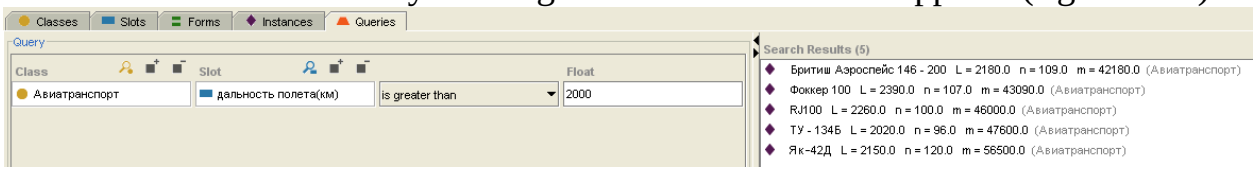

Figure 1.88. The Search results by flight distance

When creating a query, it is necessary to take into account the ±10% deviation. Therefore, the aircrafts that have the flight distance from 1800km to 2200km meet that query requirements. Modify the query according to these conditions (figure 1.89). Use the More button to add a new query bar.

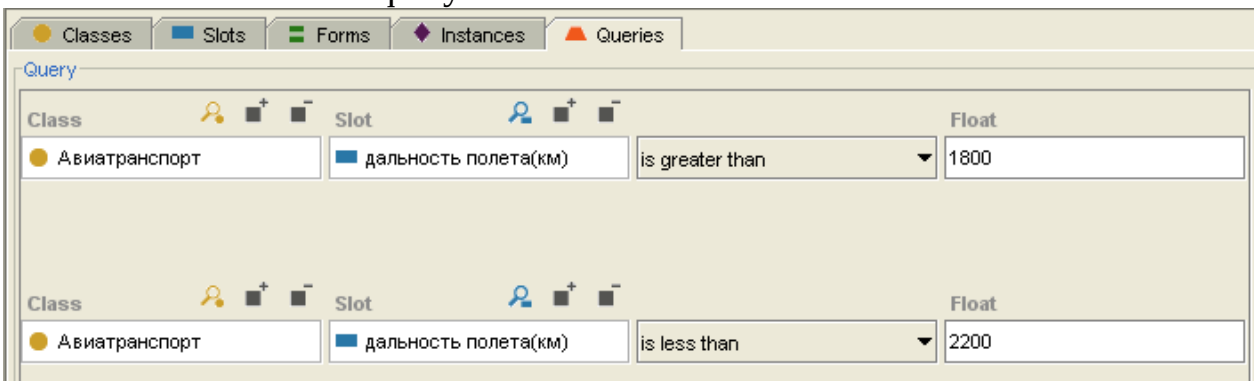

Figure 1.89. Query modification taking into account deviation

Similarly, create two more query bars to add restrictions on number of passengers with deviation (figure 1.90).

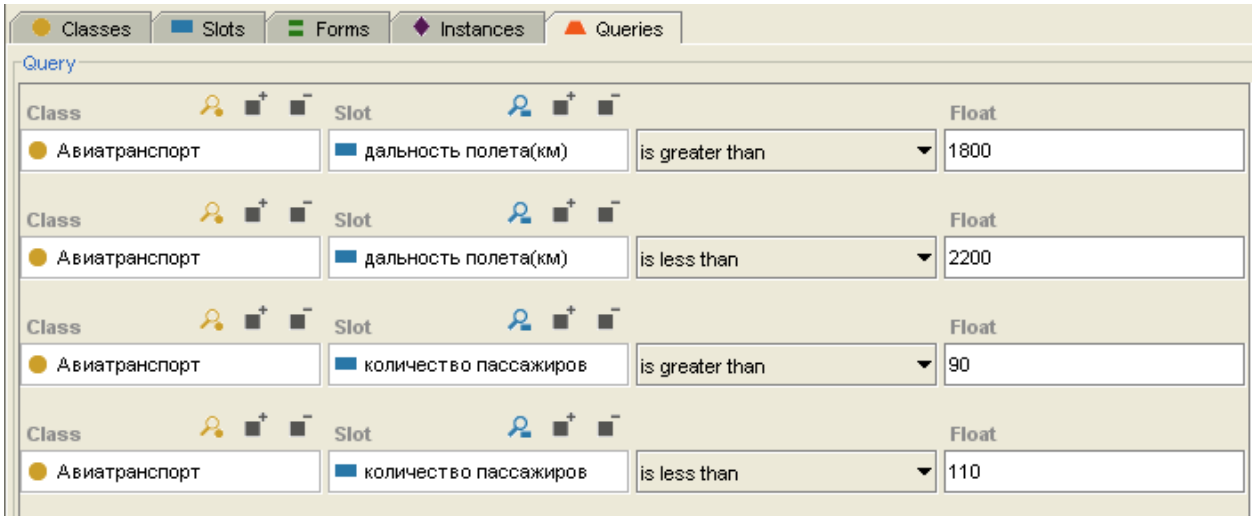

Figure 1.90. Query with deviation

To view the results of the query click the Find button. In the Search Results pane a list of vehicles that meet the conditions appears (figure 1.91).

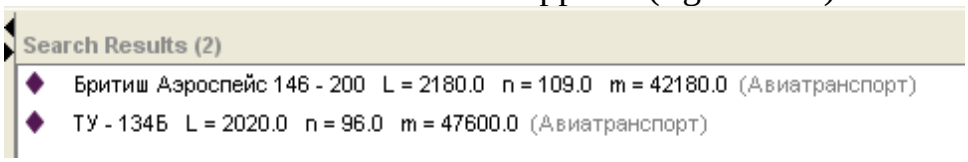

Figure 1.91. The search results

Two aircrafts “Бритиш Аэроспейс 146-200” (British Aerospace 146-200) and “Ту 134Б” (Tu-134B) meet the conditions. According to the task, it is required to select an aircraft with a minimum take-off weight. The “Бритиш Аэроспейс 146-200” (British Aerospace 146-200) aircraft meets this condition. It weighs 42,180kg (m) and has a 2,180 km flight distance (L). It is able to transport 109 passengers (n).

1.2.2. SELF-STUDY TASKS

1. Include the aircrafts in the ontology in accordance to the variant (table 1.6)
2. Select an aircraft that meets the conditions on the flight distance and number of passengers taking into account deviation.
3. Select a final aircraft based on the minimum weight.

Table 1.6. Task variants

NN	Include the following aircrafts in the ontology	Flight distance, km	Number of passengers	Deviation, %
1	АН-26Б (An-26B) Aero Spacelines GUPPY-201	2 000	98	±3
2	АН-70 (An-70) Aero Spacelines B-377 PREGNANT GUPPY	1 500	100	±10
3	АН-124 РУСЛАН (An-124 Ruslan) Boeing 747-400	2 500	150	±15
4	АН-225 МРИЯ (An-225 Mriya)	2 300	200	±10

	Miles M.69 MARATHON II			
5	АН-124-100 РУСЛАН (An-124 Ruslan) McDonnell Douglas MD-11F	2 000	150	±8
6	ИЛ-114Т (Il-114T) Let L-410 TURBOLET	1 900	150	±10
7	ТУ-330 (Tu-330) Aero Spacelines B-377	2 200	120	±5
8	Boeing 747-400 АН-124 РУСЛАН (An-124 Ruslan)	2 000	120	±5
9	Boeing 757-200 ИЛ-114Т (Il-114T)	2 500	130	±8
10	Boeing 767-300 АН-70 (An-70)	2 600	100	±3
11	Airbus A3XX-F АН-124 РУСЛАН (An-124 Ruslan)	2 400	150	±10
12	McDonnell Douglas MD-11F Miles M.69 MARATHON II	1 900	130	±15
13	Антонов АН-8 (Antonov An-8) Aero Spacelines B-377 PREGNANT GUPPY	1 800	90	±12
14	Breguet Br.761 DEUX PONTS ИЛ-114Т (Il-114T)	1 700	105	±9
15	АН-26Б (An-26B) Boeing 747-400	2 300	130	±5

1.2.3. A QUESTION CHECKLIST

1. What is an ontology?
2. What is a knowledge base?
3. What is the purpose of creating an ontology?
4. What steps are included in the process of creating an ontology in Protégé?
5. How to define whether an ontology is created correctly?
6. At which stage the Class Hierarchy cannot be modified?
7. How many slots a class can contain?
8. Can a slot be connected with several classes? How?
9. Which function allows a subclass of a class to inherit the slot of that class?
10. Is it possible to create relations among classes that are not defined in the hierarchy? How?
11. How many instances a class can contain?
12. What is a display slot? Why is it used?
13. How many display slots can be set?
14. Which parameter indicates that a widget was modified?
15. Why the Query Library is needed?

LABORATORY WORK 2

INTRODUCTION

Based on the listed below characteristics of an aircraft which information is stored in the data base, it is needed to calculate and select wing loading (part 1) and Required thrust-to-weight ratio of an aircraft (part 2).

The main characteristics and the requirements to a developing aircraft can be described in terms of ***ontology***. A product of MAGENTA CORPORATION LIMITED is used as an ontology editor. A developing aircraft will be considered as a ***project,*** or a ***demand***, and a ***resource*** is an object that is required for carrying out calculations. Based on the results of the process of searching for mutual correspondence between the project and resources (matching), a decision about reservation or releasing resources (a relation between a project and corresponding resources is set) is taken or reviewed. Thus, the necessary calculations of the values of concept attributes are done. For this purpose simplified functions based on the formulae [4] and long-haul aircraft statistics were used.

As the initial data for performing this laboratory work the following aircraft characteristics are required:

- For the first part:
  - ✓ Efficient mechanisation, yes/no;
  - ✓ Unit load on the wing (wing loading), daN/m2;
  - ✓ Take-off runway length, m;
  - ✓ Number of engines on an airplane, pcs;
  - ✓ Maximum aerodynamic efficiency;
  - ✓ Mach number;
  - ✓ Flight altitude, m.
- For the second part:
  - ✓ Flight distance, km;
  - ✓ Cruising altitude, m;
  - ✓ Mach number;
  - ✓ Number of passengers, pers.;
  - ✓ Number of passengers in a row, pers.;
  - ✓ Efficient mechanisation, yes/no.

The **aim** of the Laboratory work is to assimilate the method of ontological analysis of design procedures that deal with selecting aircraft parameters by developing applications that use scripts for calculations in unilateral matching when taking decisions.

The following **tasks** are carried out:

- Mastering the use of the ontology editor’s tools;
- Mastering the design techniques of descriptive ontology and ontology of demands and resources;
- Mastering script design techniques to calculate the values of concept attributes;
- Mastering the techniques of designing and modelling an ontology;
- Learning the structure of demand and resource agents.

Each section sets out general theoretical concepts and provides simplified formulae on defining aircraft parameters.

2.1. Choosing unit load on the wing

2.1.1. Calculation of the condition that provides the given landing speed

It is done by using the following formula (2.1)

$$p'_0 \le \frac{C_{y\max nос} \cdot V^2_{З.П.}}{30{,}2 \cdot \left(1 - \overline{m_T}\right)}, \qquad (2.1)$$

where:

– p'_0 - landing unit load on the wing, daN/m2;

– $C_{y\max nос}$ - maximum lift coefficient (it is chosen based on statistics depending on wing mechanisation). To simplify the task it is set:

– For efficient mechanisation $C_{y\max nос}$ = 2.5;

– For inefficient mechanisation $C_{y\max nос}$ = 2.0.

– $V_{З.П.}$ - landing speed, m/sec (the value is chosen based on statistics). In our task it is set as: $V_{З.П.} = 230 км/ч = 63{,}89 м/с$;

– $\overline{m_T}$ - assumed fuel mass ratio.

Using the chosen values of statistics, landing unit load on the wing is computed from the formulae (2.2) and (2.3) depending on mechanisation (initial data).

– For efficient mechanisation: $p'_0 = 10204.83/(30.2(1-\overline{m_T}))$; (2.2)

– For inefficient mechanisation: $p'_0 = 8163.86/(30.2(1-\overline{m_T}))$. (2.3)

Assumed fuel mass ratio is calculated from formula (2.4):

$$\overline{m_T} = 0.075 + \frac{0.0175 * L_p}{V_{кр}}, \qquad (2.4)$$

where:

– Lp – calculated flight distance, km (initial data);

– $V_{кр}$ - cruise speed, m/sec.

Cruise speed is calculated from formula (2.5)

$$V_{кр} = M * 20.04679 * \sqrt{288.15 - 41318.979 * H_{кр} / \left(6356766 + H_{кр}\right)}, \qquad (2.5)$$

where:

– M - Mach number (initial data);

– $H_{кр}$ – cruise altitude, m (initial data).

2.1.2. Calculation of the condition that provides the given cruise speed at the planned altitude

It is defined by using formula (2.6)

$$P_0^{''} \le \frac{0.208 * \Delta H_{кр} * V_{кр}^{2} * \sqrt{Cx_0}}{1 - 0.6 * \overline{m_t}} \quad , \tag{2.6}$$

where:

– p_0'' - unit load on the wing with a given cruise speed at the planned altitude, daN/m2;

– $V_{кр}$ - cruise speed, m/sec. It is calculated from formula (2.5);

– $\overline{m_T}$ - assumed fuel mass ratio. It is calculated from formula (2.4);

– $\Delta H_{кр}$ - air density ratio at the planned altitude, m;

– C_{x0} - zero-left drag coefficient.

Air density ratio at the planned altitude is defined from formula (2.7):

$$\Delta H_{кр} = 0.8 - 0.000046 * H_{кр}, \tag{2.7}$$

where: $H_{кр}$ - cruise altitude, m (initial data).

Zero-left drag coefficient is calculated from formula (2.8)

$$C_{X0} = (0,882 + 0,147M) \cdot \left[0,015288 + 0,000656 \cdot \lambda_ф + \frac{0,041}{\lambda^2_ф} \right]. \tag{2.8}$$

where:

– M - Mach number (initial data);

– $\lambda_ф$ - fuselage expansion.

Fuselage expansion is calculated from formula (2.9)

$$\lambda_ф = \frac{0.906 * n\_vr + 1.323 * n\_pas}{(0.649 * n\_vr + 0.6) * n\_vr}. \tag{2.9}$$

where:

– *n_vr* – number of passengers in a row, pers. (initial data should be chosen from 3 to 10 people);

– *n_pas* – number of passengers, pers. (initial data from the performance specification).

Maximum aerodynamic L/D ratio is roughly calculated from formula (2.10)

$$K_{\max} = \frac{1}{0,43\sqrt{C_{X0}}} + 1. \tag{2.10}$$

where:

– K_{max} – maximum aerodynamic L/D ratio;

– C_{x0} - zero-left drag coefficient is calculated from formula (2.8).

2.1.3. **Unit load on the wing calculation**

It is calculated by using the following formula (2.11)

$$P_0 = \min (p'_0 \ , \ p''_0). \qquad (2.11)$$

where:

– P_0 – calculated value for unit load on the wing, daN/m2;

– p'_0 - landing unit load on the wing, daN/m2. It is calculated from formula (2.1);

– p''_0 - unit load on the wing with a given cruise speed at the planned altitude, daN/m2. It is calculated from formula (2.6).

2.1.4. **Descriptive ontology**

2.1.4.1 Creating an ontology

Run the ontology editor (file *OntCons.exe).* Create a new ontology library (*File –> New*). It has a default name *OntologyLibrary_1*. Rename it to *OntologyLibrary_Wing Load*. Create a descriptive ontology of the Wing Loading domain (*New Item -> Descriptive ontology*). It has a default name *Ontology_1*. Rename it to *Ontology_Wing Load*.

2.1.4.2 Creating and deleting a concept

When an ontology is created, the tree, which nodes are the categories of ontology concepts, can be opened by clicking the <+> button. These are abstract base classes. To create your own ontology it is necessary to inherit descendant classes from them. To create a descendant class, select a concept, which will be an ancestor for this concept, click the right mouse button and select *New Item*. The created concept can be deleted by selecting it and pressing the <Del> button, or by selecting *Delete* from the shortcut menu.

'Object' concept

The Object concept is an entity that exists in the world described in ontology. When the descriptive ontology of the Wing Loading domain is made it is required to create two Object concepts.

Create the object *Wing_Demand (Objects ->New Item -> Object)*, and rename it to *Wing_Demand.* Set three different icons for this object. Similarly create the object *Wing_Resource,* rename it to *Wing_Resource.* Set icons for this object.

Each Object concept can have a certain list of attributes.

'Attribute' concept

The Attribute concept is a value that characterise an object (quantitative).

- Create a **Boolean attribute** (*Attributes -> New Item -> Boolean Attribute*), and rename it to *_Effective_mehan*.
- Create **integer attributes** (*Attributes -> New Item -> Integer Attribute*), and rename them to *_H_kr, _L, _n_pas, _n_vr, X, Y.*

- Create **float attributes** (*Attributes ->New Item -> Float Attribute*), and rename them to *_M, Motn_topl, V_kr, DeltaH_kr, K_max, CX_0, Lambda_fuz, P0_zahpos, P0_H, P0.*

2.1.4.3 Creating relations among concepts

To create a relation among concepts the drag-and-drop action is used. For example, to add an attribute to a list, "grab" it and drag it to the Object concept.

To create relations among created concepts of the descriptive ontology of the Wing Loading domain, i.e. to indicate that the *Wing_Demand* object has the following attributes: *_Effective_mehan, _H_kr, _M, _L, _n_pas, _n_vr, Motn_topl, V_kr, DeltaH_kr, CX_0, Lambda_fuz, P0_zahpos, P0_H, P0, K_max, X, Y,* and the *Wing_Resource* object has the *X,Y* attributes, "grab" the *_Effective_mehan, _H_kr, _M, _L, _n_pas, _n_vr, Motn_topl, V_kr, DeltaH_kr, CX_0, Lambda_fuz, P0_zahpos, P0_H, P0, K_max, X, Y* attributes on the concept tree and drag them onto the *Wing_Demand* object. Similarly, "grab" and drag the *X,Y* attributes onto the *Wing_Resource* object.

The list of object attributes can be viewed in the property editor of the Object concept.

As a result, the object will have a list of attributes' names in the *Uses* tab; and the attribute will have the name of the object (names of objects) using this attribute in the *Used by* tab. Figure 2.1. shows the properties of the *Wing_Demand* object in the *Uses* tab. This object has the following attributes: *_Effective_mehan, _H_kr, _M, _L, _n_pas, _n_vr, Motn_topl, V_kr, DeltaH_kr, CX_0, Lambda_fuz, P0_zahpos, P0_H, P0, K_max, X, Y* (the relations are presented in the *Uses* tab). The *Used by* tab shows the relations with the corresponding object.

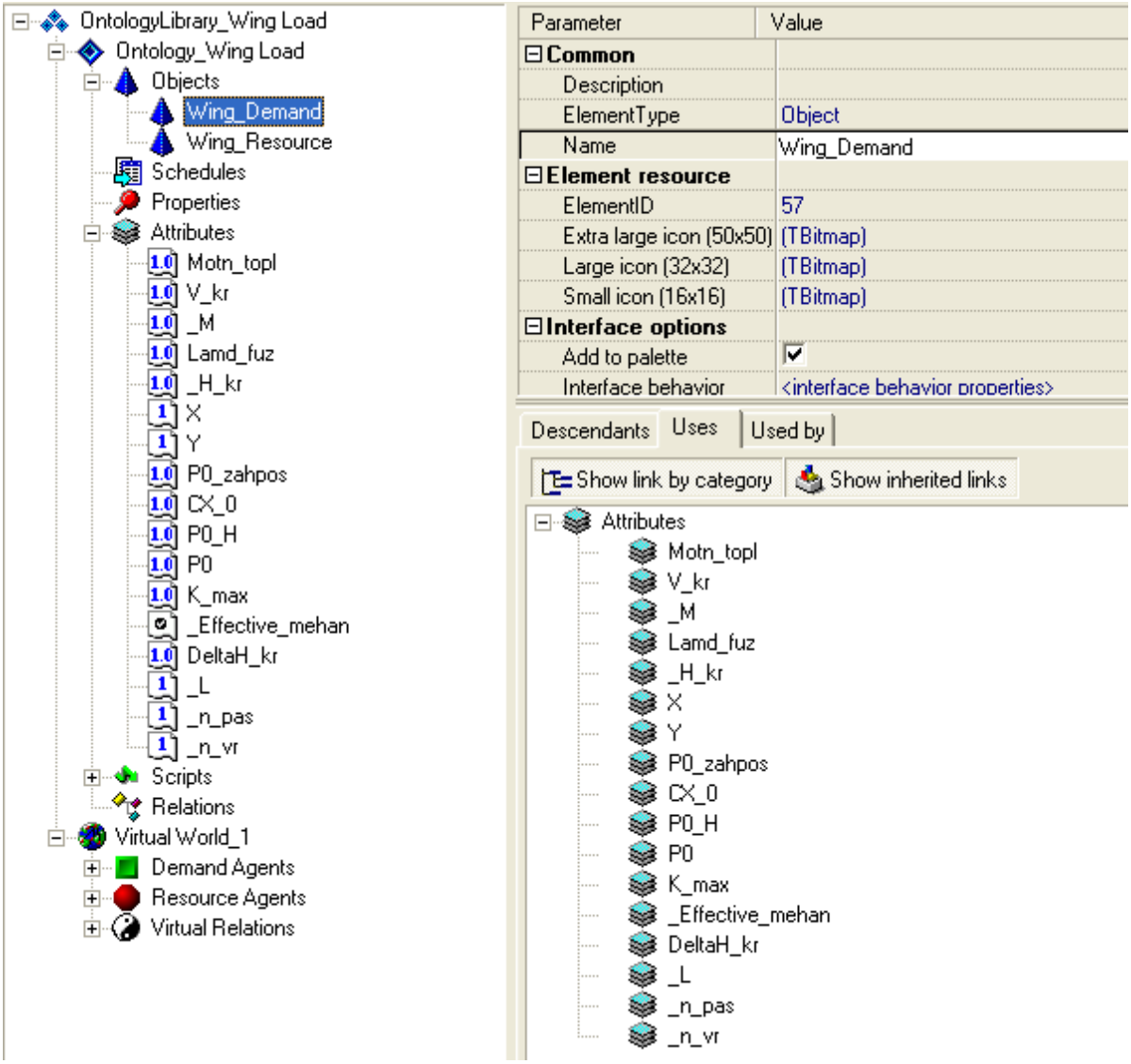

Figure 2.1. Attributes of the *Wing_Demand* object in the *Uses* tab

'Script' concept

The 'Script' concept is a specific rule in programming language for calculating a value. Ontology builders use a subset of Object Pascal to write scripts. Scripts should be used to calculate an attribute value depending on the values of other attributes. Only the concepts and their parameters (e.g. attributes of an object, etc.), which are, in turn, the script's parameters (i.e. they are in the script's *Uses* tab), can be used in the script.

To indicate that a concept is a parameter of a script, drag this concept onto the corresponding Script concept. The result of calculations carried out in the script should be connected with an attribute of a corresponding object. To this end, drag the Script concept onto the desired object.

Then, it is required to select script editor, or to edit the script body (select *Other -> Script* in the property editor of the script, or click the Edit script button in the *Script body* tab).

– **Defining a script to calculate cruise speed**

To calculate cruise speed with formula (2.5) using a script, it is needed to create a Script concept that will calculate cruise speed. To this end, select the *Scripts* category in the descriptive ontology concept tree, then select *New item -> Script* in the shortcut

menu. The created script rename to *V_kr Calculate* and connect it to the *V_kr* attribute (i.e. drag the script onto the *V_kr* attribute). Indicate the script's parameters: drag the necessary attributes onto the script concept (all the script's parameters can be seen in the *Uses* tab). The parameters of the *V_kr Calculate* script, which calculates cruise speed, are the *_M* and *H_kr* attributes (figure 2.2). Write the script body: Select the *V_kr Calculate* script in the descriptive ontology concept tree, then navigate to the *Script body* tab and click the Edit script button. The Script Editor dialogue box appears; write there the written below text (the concept names should be quoted, insignificant space is not allowed). The concept names should be selected in the list of concepts which are the script's parameters. The script ends with a semi-colon.

begin
Result := "_M" * 20.04679 * sqrt(288.15 - 41318.979 * "_H_kr"/
(6356766+"_H_kr"));
end;

Save the script by clicking the button. Close the Script Editor window. Check the script syntax by clicking the *Check syntax* button. If an error is found, it is necessary to open the script editor and make the required corrections.

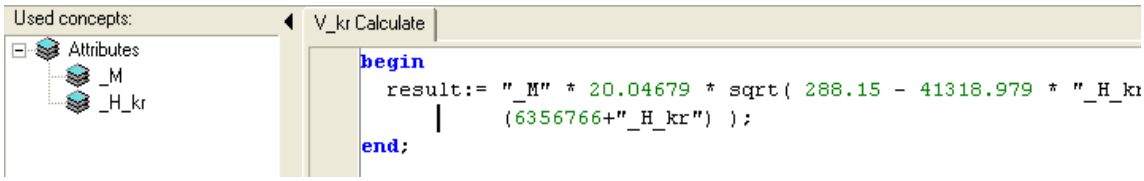

Figure 2.2. The attributes and the body of the *V_kr Calculate* script

Similarly create the following scripts.

- **Defining a script to calculate fuel mass ratio (by using formula 2.4)**

Create a script and rename it to *Motn_topl Calculate*. Connect it to the *Motn_topl* attribute. The *_L* and *V_kr* attributes are the parameters of the *Motn_topl Calculate* script. Write the script body (figure 2.3).

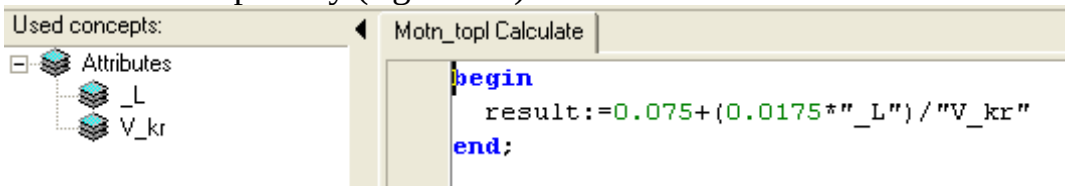

Figure 2.3. The attributes and the body of the *Motn_topl Calculate* script

- **Defining a script to calculate landing unit load on the wing (by using formulae 2.2 and 2.3)**

Rename the created script to *P0_zahpos Calculate* and connect it to the *P0_zahpos* attribute. The *_Effective_mehan* and *Motn_topl* attributes are the parameters of the *P0_zahpos Calculate* script. Write the script body (figure 2.4).

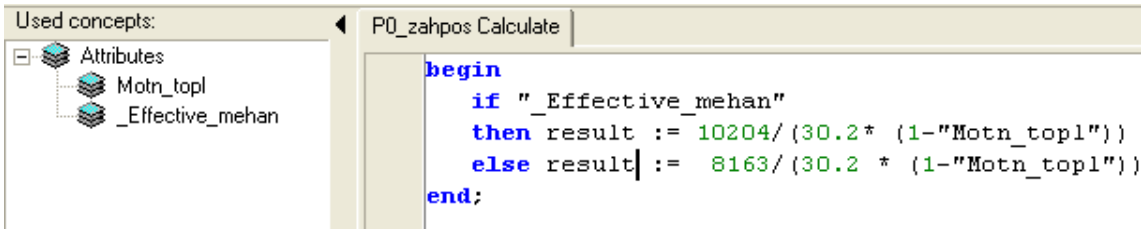

Figure 2.4. The attributes and the body of the *P0_zahpos Calculate* script

- **Defining a script to calculate air density ratio at the planned altitude (by using formula 2.7)**

Rename the created script to *DeltaH_kr Calculate* and connect it to the *DeltaH_kr* attribute. The *_H_kr* attribute is the parameter of the *DeltaH_kr Calculate* script. Write the script body (figure 2.5).

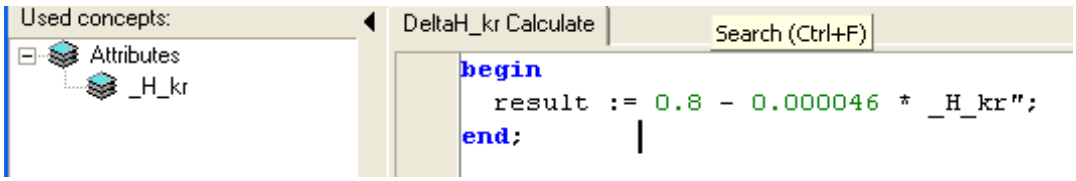

Figure 2.5. The attributes and the body of the *DeltaH_kr Calculate* script

- **Defining a script to calculate fuselage expansion (by using formula 2.9)**

Rename the created script to *Lambda_fuz Calculate* and connect it to the *Lambda_fuz* attribute. The *_n_pas* and *_n_vr* attributes are the parameters of the *Lambda_fuz Calculate* script that calculates fuselage expansion. Write the script body (figure 2.6).

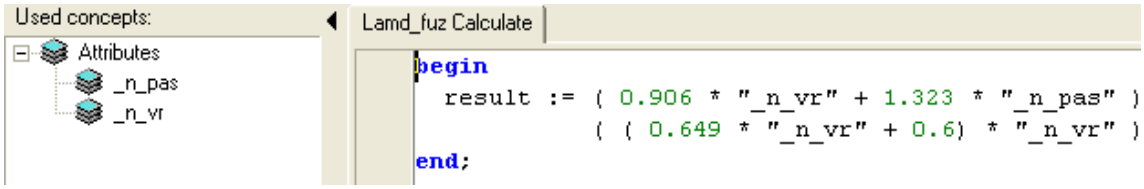

Figure 2.6. The attributes and the body of the *Lambda_fuz Calculate* script

- **Defining a script to calculate zero-left drag coefficient (by using formula 2.8)**

Rename the created script to *CX_0 Calculate* and connect it to the *CX_0* attribute. The *_M* and *Lamd_fuz* attributes are the parameters of the *CX_0 Calculate* script that calculates zero-left drag coefficient. Write the script body (figure 2.7).

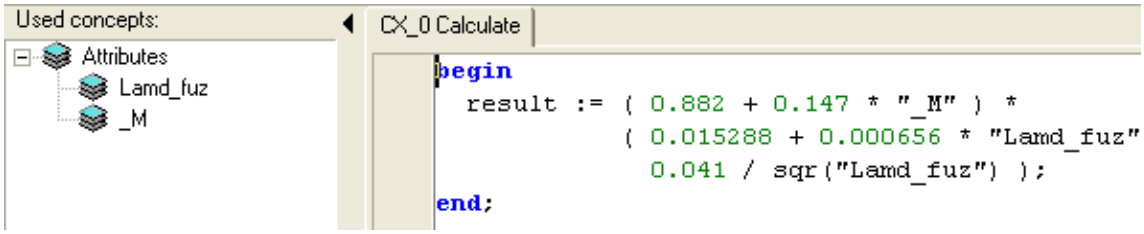

Figure 2.7. The attributes and the body of the *CX_0 Calculate* script

- **Defining a script to calculate unit load on the wing with a given cruise speed at the planned altitude (by using formula 2.6)**

Rename the created script to *PO_H Calculate* and connect it to the *PO_H* attribute. The *V_kr*, *DeltaH_kr*, *CX_0* and *Motn_topl* attributes are the parameters of the *PO_H Calculate* script that calculates zero-left drag coefficient. Write the script body (figure 2.8).

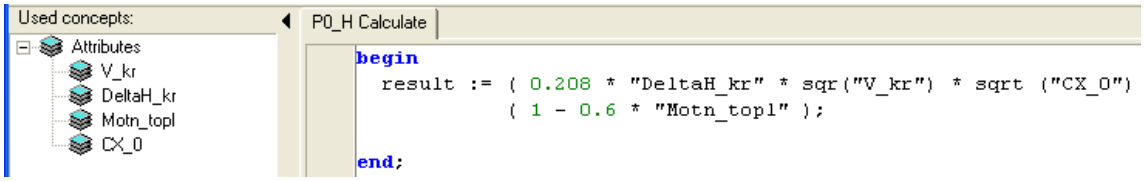

Figure 2.8. The attributes and the body of the *PO_H Calculate* script

- **Defining a script to calculate unit load on the wing (by using formula 2.11)**

Rename the created script to *PO Calculate* and connect it to the *PO* attribute. The *P0_zahpos* и *PO_H* attributes are the parameters of the *PO Calculate* script that calculates zero-left drag coefficient. Write the script body (figure 2.9).

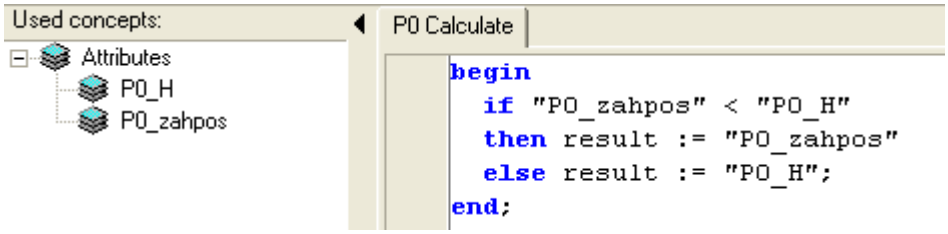

Figure 2.9. The attributes and the body of the *PO Calculate* script

- **Defining a script to calculate maximum aerodynamic L/D ratio (by using formula 2.10)**

Rename the created script to *K_max Calculate* and connect it to the *K_max* attribute. The *CX_0* attribute is the parameter of the *K_max Calculate* script that calculates maximum aerodynamic L/D ratio. Write the script body (figure 2.10).

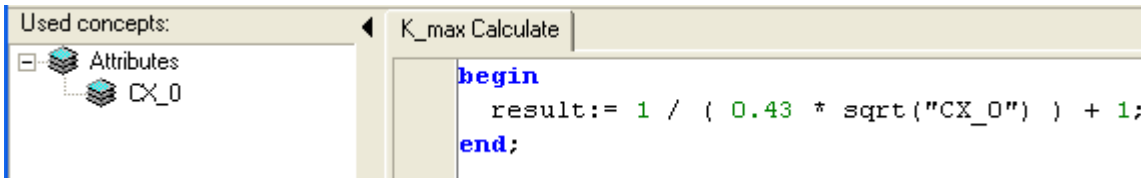

Figure 2.10. The attributes and the body of the *K_max Calculate* script

Thus, the following scripts can be seen on the descriptive ontology tree concept of the Wing Loading domain (figure 2.11).

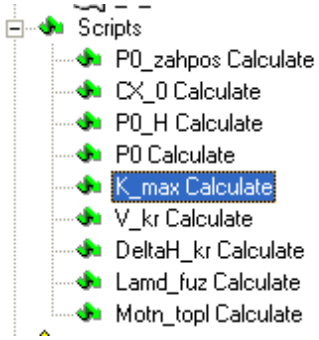

Figure 2.11. Scripts of descriptive ontology

2.1.4.4 Ontology representation in the Semantic Web

Descriptive ontology can be represented not just as a concept tree but also in the Semantic Web, that is an directed graph, where the nodes are the concepts of ontology and the connectors are the relations between the concepts. A user is able to drag the concepts of the Semantic Web within the screen using the mouse.

Select *Tools -> Ontology as network* for descriptive ontology representation in the Semantic Web.

Descriptive ontology is represented as a concept tree on the left of the *Ontology Network* pane, and in the Semantic Web at the right side (figure2.12).

Closing the window *Ontology Network* returns you to the Ontology editor.

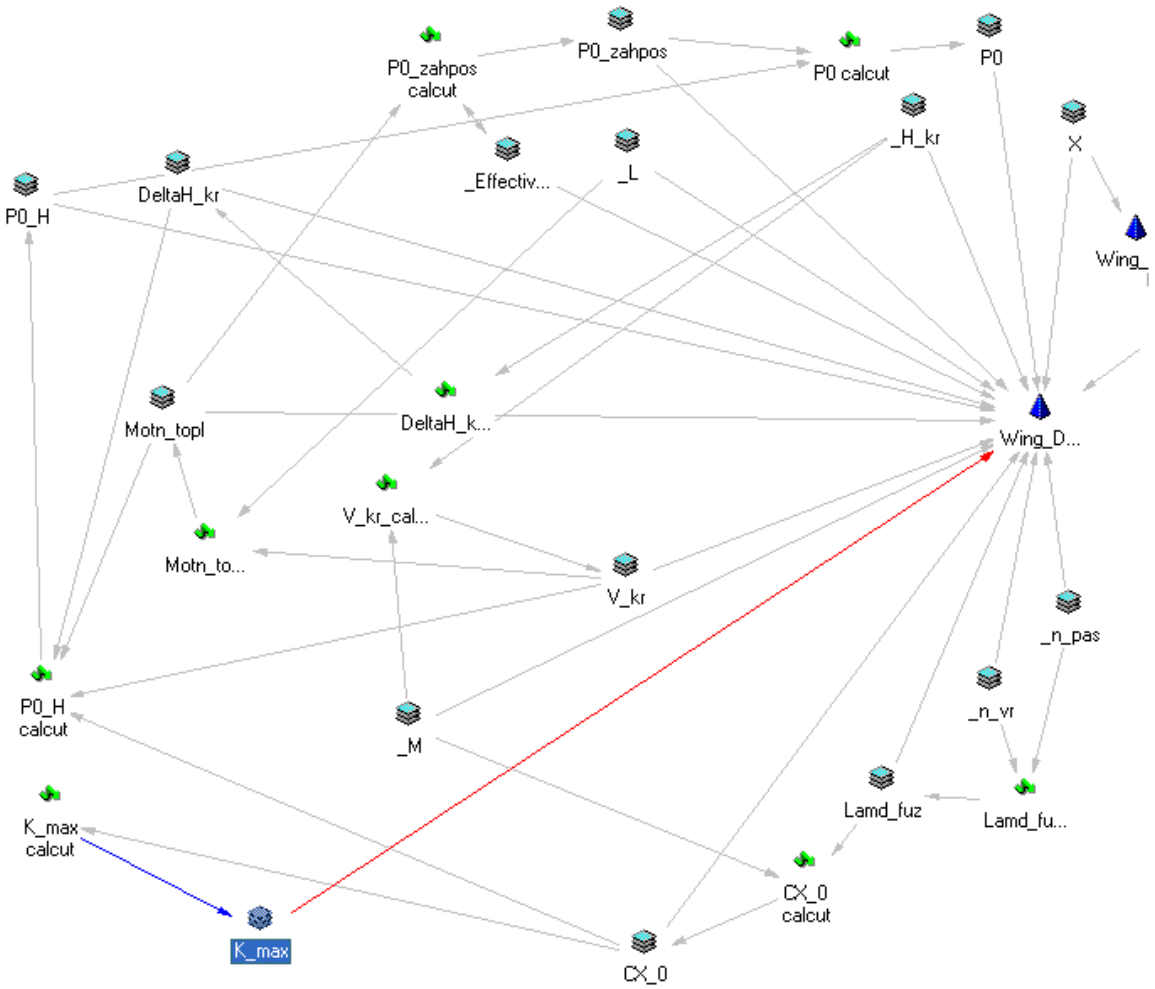

Figure2.12. Descriptive ontology representation in the Semantic Web of the Wing Loading domain

2.1.5. **Ontology of demands and resources**

2.1.5.1 ***Creating an ontology***

To create an ontology of demands and resources (virtual world ontology), select the ontology library and click *New item -> Virtual world ontology* in the shortcut menu. A dialogue box of creating an ontology of demands and resources appears on the right side of the screen. This dialogue allows choosing the 'Object' concepts which require creating demand or resource agents. If an object should have both a demand agent and a resource agent, tick the box on the left from this object. The agents are created automatically. If an object in the virtual world corresponds to either a demand agent or a resource agent, then the box should not be ticked. The agents will be created later on an individual basis. Thus, in this example the object "designing plane" serves as a demand and should have only a demand agent in the virtual world. In turn, the object resource is a resource and should have only a resource agent in the virtual world. Clicking the <*OK*> button confirms the necessity of creating a virtual world ontology (figure 2.13).

When clicking <*OK*> the virtual world ontology appears in the left side of the screen. The concept tree, which contains the concepts of demand and resource agents as well as the relations among the agents, can be opened by clicking <+> (figure 2.13).

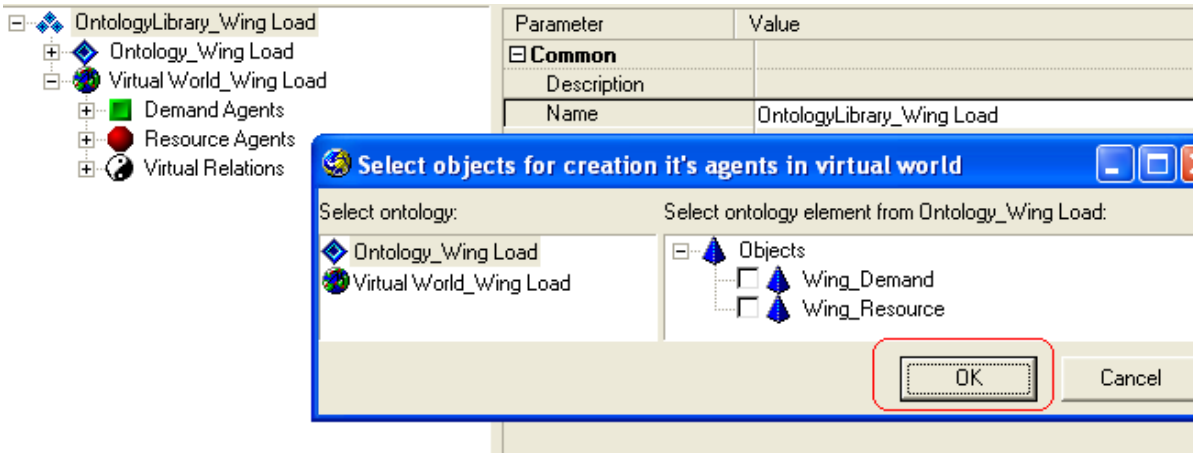

Figure2.13. Selecting objects for creation their agents

Create an ontology of demand and resources of the Wing Loading domain (*New Item -> Virtual World Ontology*). Type the name *Virtual World_Wing Load.* Disclose the concept tree of the virtual world ontology (figure 2.14).

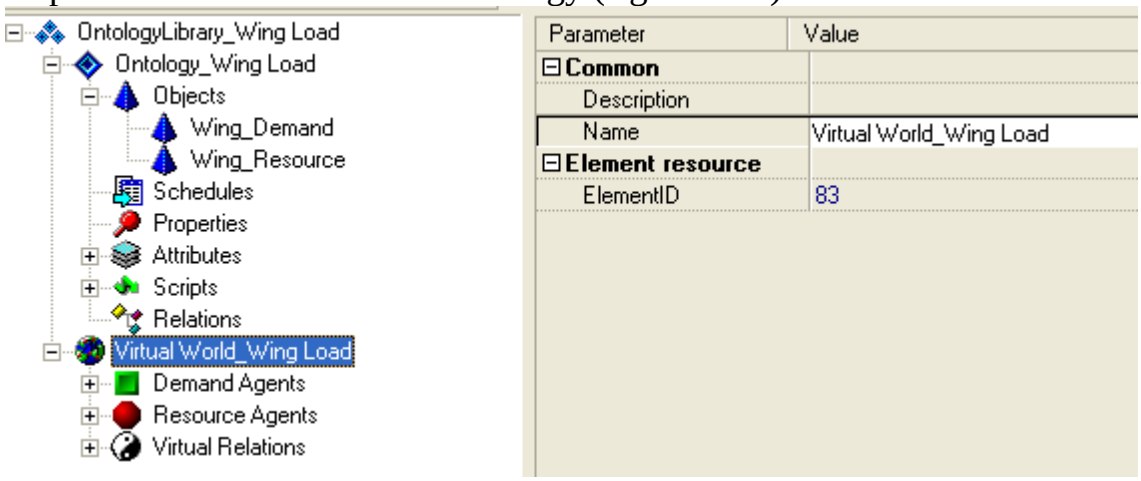

Figure 2.14. Concept categories of the demand and resource ontology

2.1.5.2 ***Creating the Demand Agent concept***

Select the *Demand Agents* category, then in the shortcut menu press *New Item -> Demand agent*. In the new dialogue select the *Wing_Demand* concept. Click <*OK*> and rename it to *Wing_Demand* (figure 2.15). Set three types of icons to this concept, which will be shown when working with the scene while modelling. Tick the *vaoAutoCreate* box (by default).

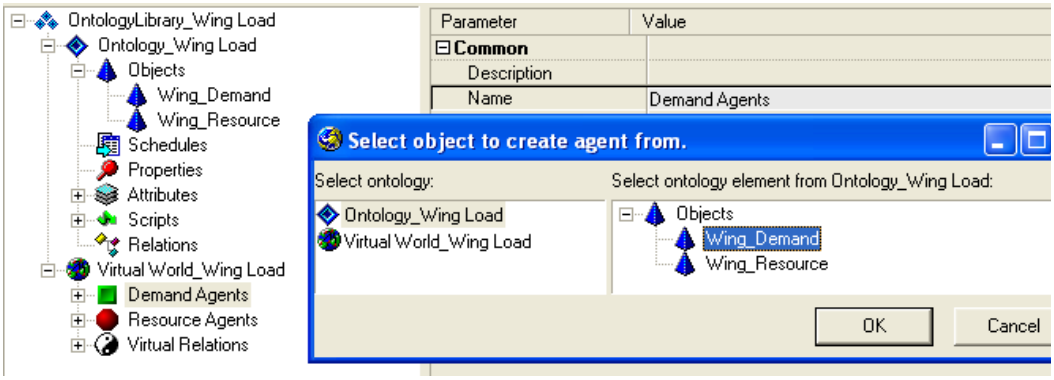

Figure 2.15 – Creating a demand agent for the *Wing_Demand* concept

2.1.5.3 ***Creating the Resource Agent concept***

Create the Resource Agent concept for the Object Resource concept: select the *Resource Agents* category and click *New Item -> Resource agent* in the shortcut menu. Select *BD_Plane* and click <*OK*>. Rename the created concept to *BD_Plane Resource* (figure 2.16). Set three types of icons to this concept, which will be shown when working with the scene while modelling. Tick the *vaoAutoCreate* box. Do not tick the *raoActive* box.

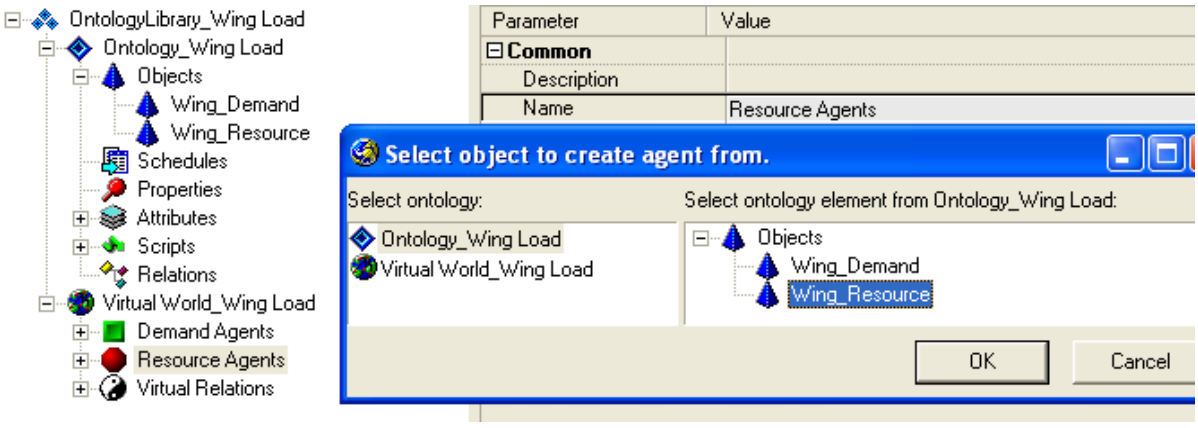

Figure 2.16 – Creating a resource agent for the *Wing_Resource* concept

2.1.5.4 ***Virtual relations. Matching relation***

A Matching relation is a utility class of virtual world relations. It connects the demand and resource concepts. A Matching relation shows a *possibility* of having a match among the agents, which concepts are connected by this relation in the ontology.

In other words, matching is possible but it does not necessarily take place, because the agents may not agree by some reasons (there may be a more beneficial offer, this offer does not suit the partner/agent, etc.)

A Matching relation is only possible between the agents of demand and resource. For example, matching between agents of demand is impossible. A Matching relation is the subject-object relation. Subject is the initiator of matching. A demand agent and a resource agent can set matching relation in the scene. Either demand agent or resource agent can be the matching subject (if *raoActive* is set).

Set unilateral matching relation between *Wing_Demand* and *Wing_Resource*: Select the *Matching relation* concept in the *Virtual Relations* category, then click *Establish relation* in the shortcut menu. On the right side open the Virtual World tree, and there the *DemandAgents* and *ResourceAgents* categories. Select *Wing_Demand* as the *Matching subject* and *Wing_Resource* as the *Matching object* (figure 2.17). In the *Used by* tab the *Wing_Demand.Wing_Resource* matching relation can be seen. (figure 2.17)

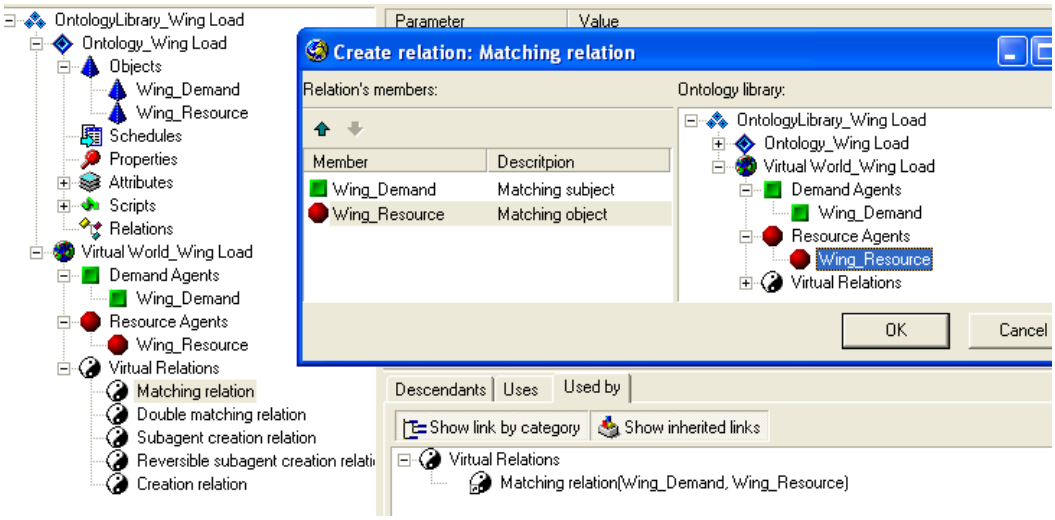

Figure 2.17 – Setting the matching relation for *Wing_Demand* and *Wing_Resource*

2.1.5.5 ***Matching conditions***

Navigate to the *Used by* tab (*Virtual relations -> Matching relation -> Used by*) and select *Matching relation (Wing_Demand, Wing_Resource)*; then click *Edit virtual relation properties* in the shortcut menu (figure 2.18). The Matching relation properties window appears (figure 2.19).

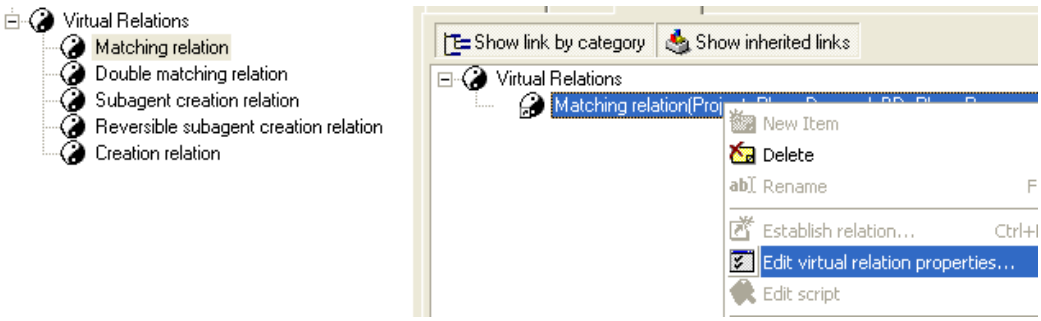

Fig 2.18 – Matching relation properties editing

The following tabs will be opened in the matching properties window:

– *Matching conditions:* creation and editing the matching conditions. Sign and script conditions are described above. *Name* is a condition type (recorded automatically). *Checking agent* is an agent that checks the condition of matching, e.g. agent-subject (recorded automatically).
– *Decision Making Machine conditions:* creating and editing the criteria, based on which in matching a decision to reserve a resource agent by a demand agent is taken.
– *Tasks:* forming tasks to calculate additional attributes for a matcher (is not used in this laboratory work).
– *Events:* event handler which is run when it is required to change a value of an agent attribute depending on the attribute values in matcher (is not used in this laboratory work).

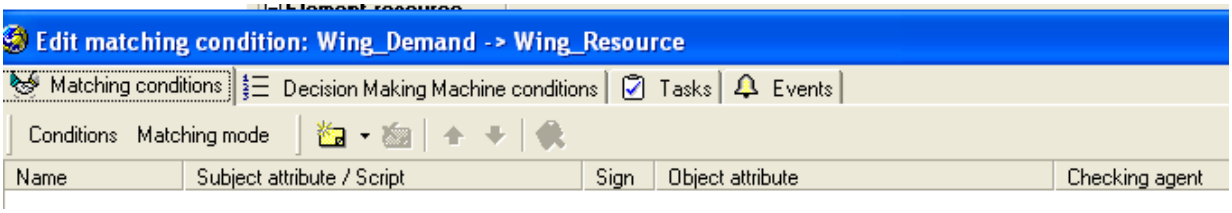

Figure 2.19 – Edit matching condition

In this task matching is needed just to run scripts that perform calculations, so special matching conditions are not required.

2.1.5.6 ***Decision Making Machine condition***

Decision Making Machine condition allows an agent to select an offer (matching) from other partners. It is created in the *Decision Making Machine conditions* tab of the *Edit Matching Conditions* pane by using the button. It is necessary to identify a condition attribute, optimisation (maximum/minimum) and weight coefficient that defines the importance of this condition.

– **Creating a decision making machine condition: choosing a minimum value of landing wing loading and of wing loading with the given cruise speed at the planned altitude**

The minimum from one of the two values should be taken as the calculated value of wing loading (formula 2.11): landing wing loading (formula 2.1) and wing loading with the given cruise speed at the planned altitude (formula 2.6).

Create Decision Making Machine condition 1 for the *Wing_Demand -> Wing_Resource* matching: click the button in the *Decision Making Machine conditions* tab. Set the following parameters (figure2.20):

– Attribute = ‘Wing_Demand.P0’;
– Order = ‘Min’;
– Weight = ‘100’.

Tick the *Active* box to activate the decision making machine condition.

– **Creating a decision making machine condition: maximising the aerodynamic efficiency**

Create Decision Making Machine condition 2 for the *Wing_Demand* -> *Wing_Resource* matching. Set the following parameters (figure 2.20):

– Attribute = 'Wing_Demand.K_max';
– Order = 'Max';
– Weight = '100'.

Activate the decision making machine condition.

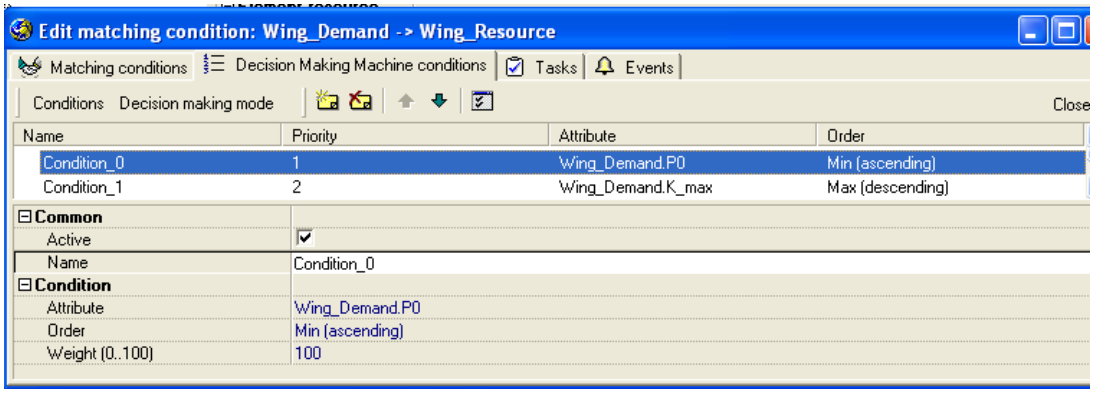

Figure 2.20 – Decision Making Machine conditions for the *Wing_Demand – Wing_Resource* matching

2.1.5.7 *Interface behaviour*

The Interface options pane (figure 2.21) identifies the behaviours of an instance of the Object concept.

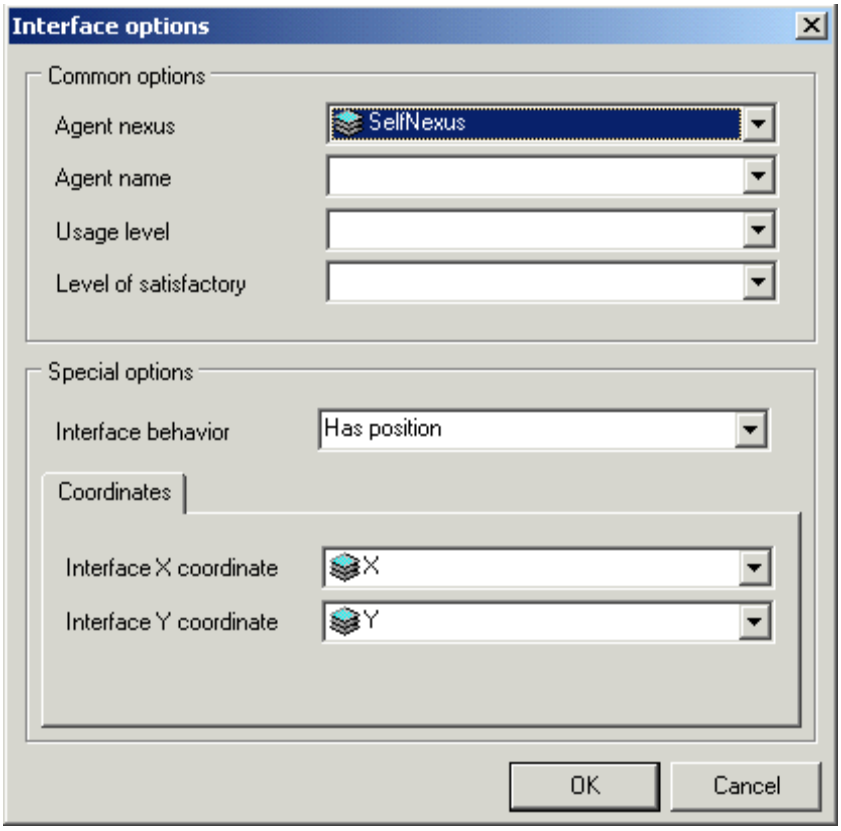

Figure 2.21 –Interface options

Connect the *X*, *Y* attributes with the agent position of the *Wing_Demand* object in the scene. To do so, open the *Interface behaviour* dialogue in the *Wing_Demand* properties, select the *Has position* parameter from the list. Then, select *X* and *Y* in the *Interface X coordinate* and *Interface Y coordinate* combo boxes respectively. Click <OK>.

Connect the *X*, *Y* attributes with the agent position of the *Wing_Resource* object in the scene. To do so, open the *Interface behaviour* dialogue in the *Wing_Resource* properties, select the *Has position* parameter from the list. Then, select *X* and *Y* in the *Interface X coordinate* and *Interface Y coordinate* combo boxes respectively. Click <OK>.

2.1.5.8 ***Saving an ontology of the Wing loading domain***

The concept tree of the descriptive ontology and the ontology of the virtual world of the Wing loading domain is shown at figure 2.22.

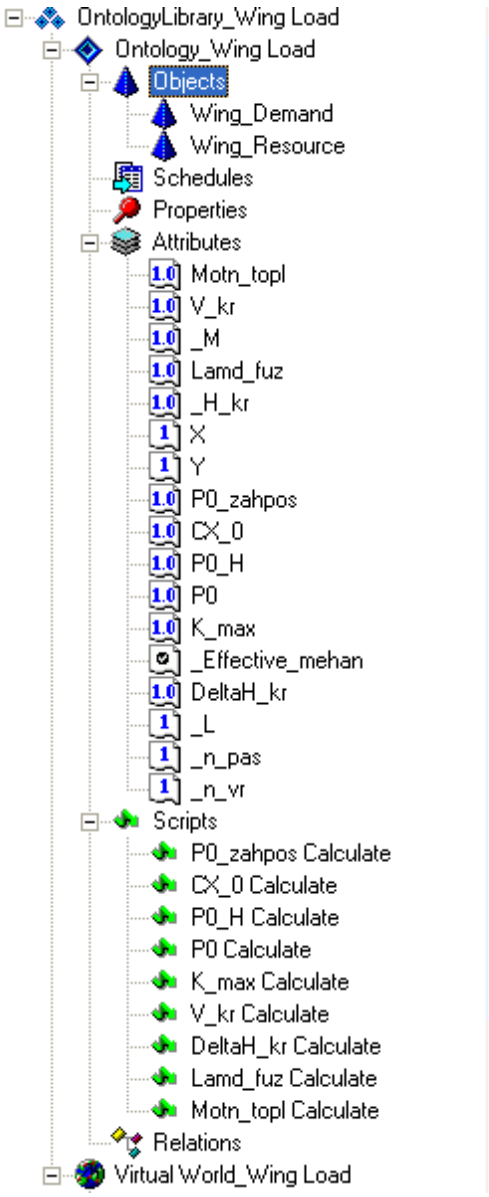

Figure 2.22 – The concept tree of the descriptive ontology and the ontology of the virtual world of the Wing loading domain

Save the created ontologies (descriptive ontology and demand and resource ontology) as *Wing Load Ontology* by using the button. The *.ocl* extension is added automatically. The file is located in the *Ontology Samples* category by default. Finish the work with the ontology constructor (*File -> Close*).

2.1.6. **Creating an ontological scene**

Modify some settings on your computer: Start -> Control panel -> Regional and Language options -> Customise this format -> select a dot in the Decimal symbol drop-down list (figure 2.23).

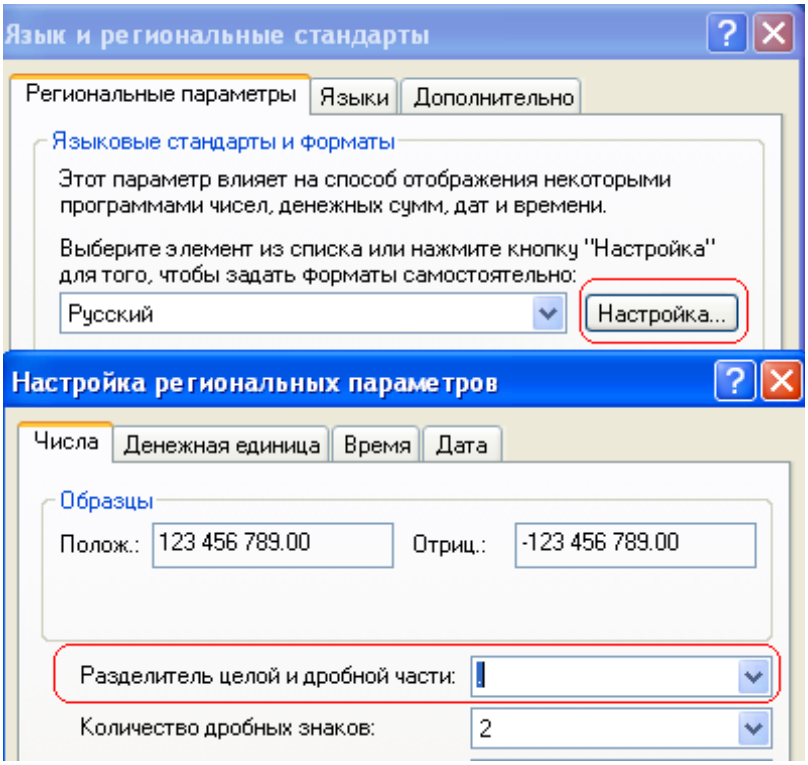

Figure 2.23 – Customising computer settings

Run the UniIntf.exe file that is located in the *OntConsUniIntf* folder. Create a new ontological scene (*File → New scene -> Load ontology*, select *Wing Load ontology.ocl)*. In the Physical world pane create *Wing Demand_1*, an agent of the designing aircraft, and *Wing Resource_1*, an object-resource agent to perform calculations (figure 2.24). Set the following values of the attributes for agents (for example, for Tu-154 aircraft) using Agent inspectors

Attributes	***Tu-154***	***Il-86***	***Yak-40***
_Effective_mehan	+	+	+
_H_kr	10100	10000	8100
_M	0.88	0.88	0.47
_L	5200	4350	1350
_n_pas	164	350	32
_n_vr(3..10)	6	8	3

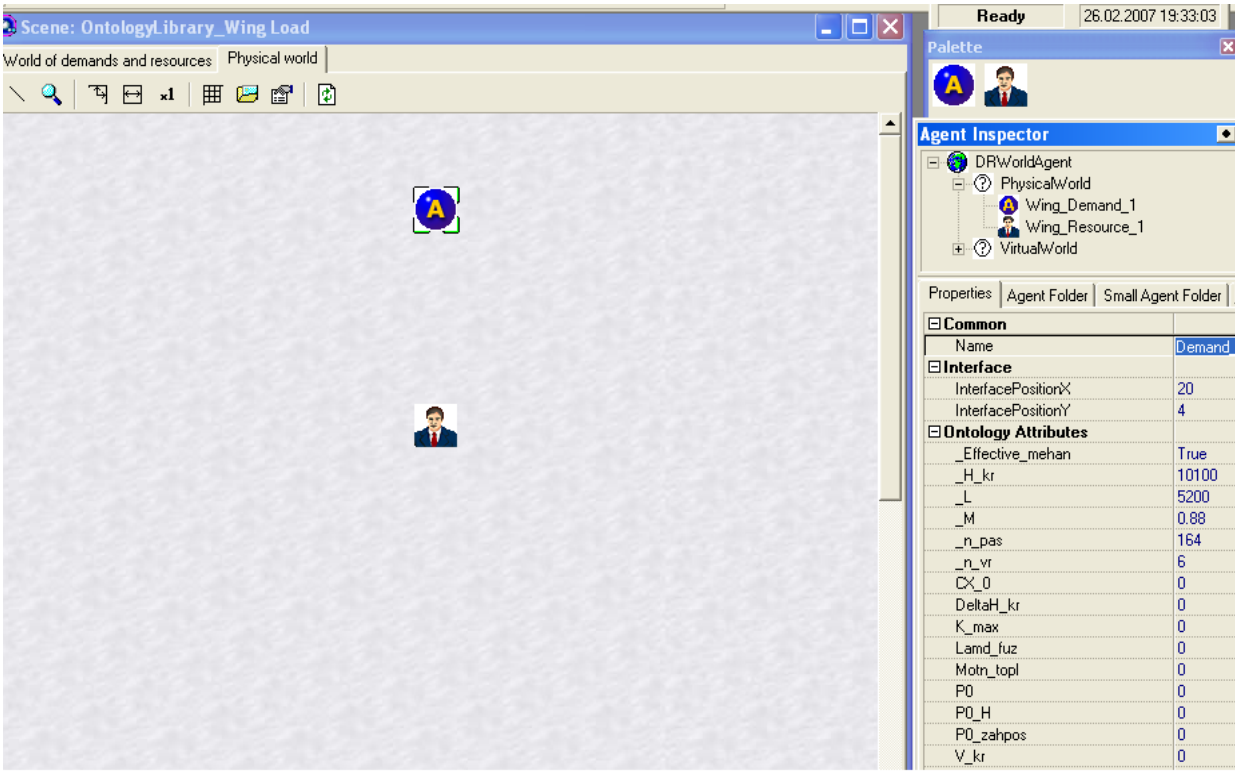

Figure 2.24 – Creating a scene of the Wing loading domain

2.1.7. **Modelling a virtual world scene**

2.1.7.1. Running the modelling scene for Tu-154

Navigate to the *Worlds of demands and resources* tab. Run the modelling scene (i.e. matching process) using the  button. Observe the matching process among the agents of the designing aircraft and the aircraft prototype agents in the database.

In the matching process the active agent of the designing aircraft runs scripts required to calculate the attribute values. Then the agent builds a decision making table, in which the only reserved object-resource agent is indicated. The table where the agent structure is presented (the Matchers tab) can be opened by double clicking the icon of the *Wing Demand_1* agent in the Physical world tab or the tab of world of demands and resources (or selecting *Actions -> Show agent structure* in the shortcut menu). The list of resources (partners), i.e. the agents that are reserved by the active agent is indicated in the table (figure 2.25). When a partner is selected the attributes of the active agent are shown on the right side of the screen:

- *Simple:* the values are set in the scene;
- *Scripted:* the values are calculated using scripts during the matching process.

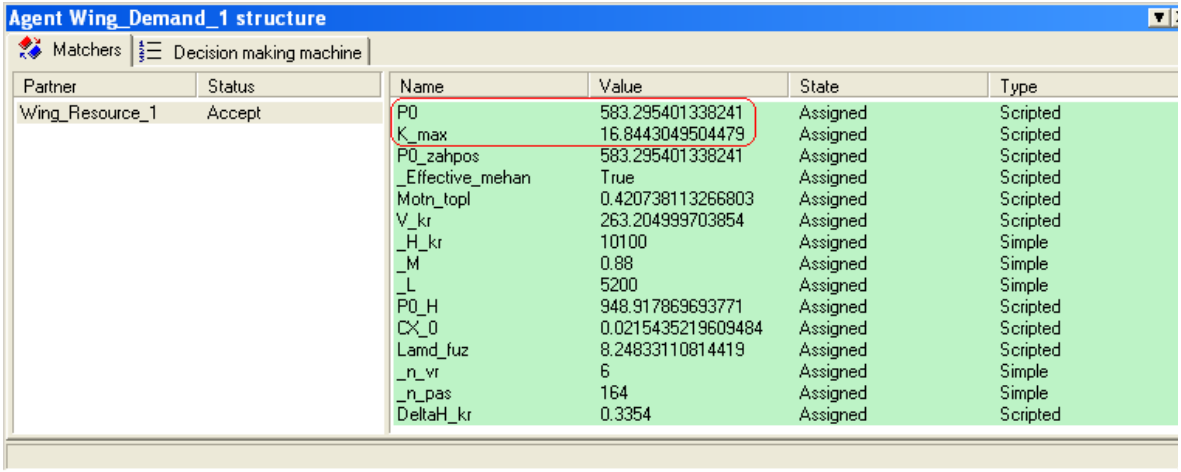

Figure 2.25 –Agent *Wing Demand_1* structure for Tu-154 aircraft

The basic parameters that are calculated using scripts are shown in the *Decision Making Machine* tab (figure 2.26).

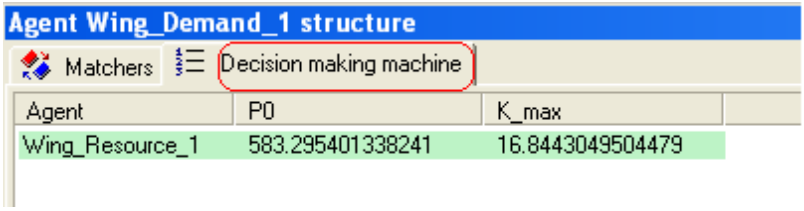

Figure 2.26 – The Decision Making Machine tab of the *Wing Demand_1* agent for Tu-154 aircraft

In this task the agent-resource does not have attributes, that is why the table that contains its structure, is empty.

The final matching results can be seen on figure 2.27. As a result the following reservation operation was performed:

– *Wing Demand_1 – Wing Resource_1*.

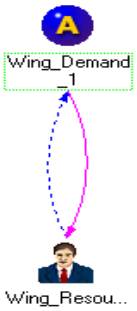

Figure 2.27 – Matching results

2.1.7.2. Saving a virtual world scene

A scene can be saved by using the button or by selecting *Tools - > Save ontology scene…* Save the scene as *Scene_Wing Load TY-154*. The.*osf* extension is added automatically. The file is located in the *Ontology Samples* category by default. Finish the work with the system (*File -> Close*).

2.1.7.3. Loading an ontology scene

To load a scene click the button or select *Tools - > Load ontology scene…* Load the *Scene_Wing Load TY-154.osf* file.

2.1.7.4. Modelling a scene for Il-86 aircraft

Clear the results of modelling by selecting *Tools -> Clear results*. Change the attribute values of agent *Wing Demand_1*. Navigate to the *Worlds of demands and resources* tab. Run the modelling scene (i.e. matching process) by clicking the button. Observe the matching process among the agents of the designing aircraft and the aircraft prototype agents. The matching results are shown on figure 2.28, and the Decision making machine tab is on figure 2.29.

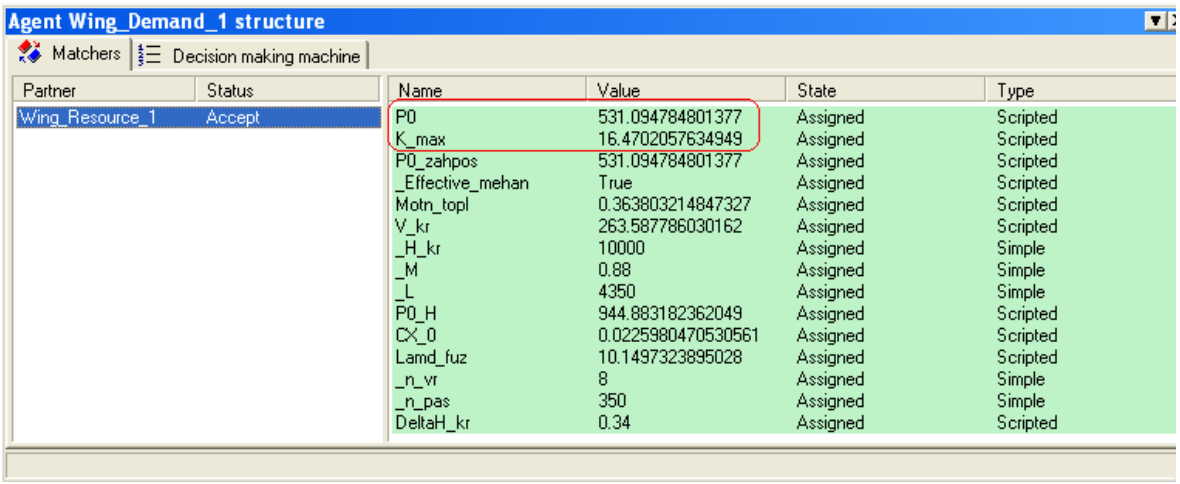

Figure 2.28 – Agent *Wing Demand_1* structure for Il-86 aircraft

The basic parameters that are calculated using scripts are shown in the *Decision Making Machine* tab (figure 2.29).

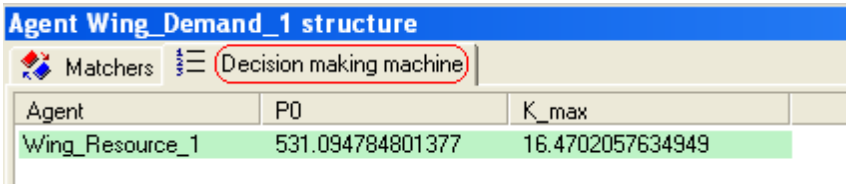

Figure 2.29 – The Decision Making Machine tab of the *Wing Demand_1* agent for Il-86 aircraft

In this task the agent-resource does not have attributes, that is why the table that contains its structure, is empty. Save the scene as *Scene_Wing Load ИЛ-86*. The.*osf* extension is added automatically. The file is located in the *Ontology Samples* category by default. Finish the work with the system (*File -> Close*).

2.1.7.5. Modelling a scene for Yak-40 aircraft

Similarly run the matching process for Yak-40 aircraft. The matching results are shown on figure 2.30, and the Decision making machine tab is on figure 2.31.

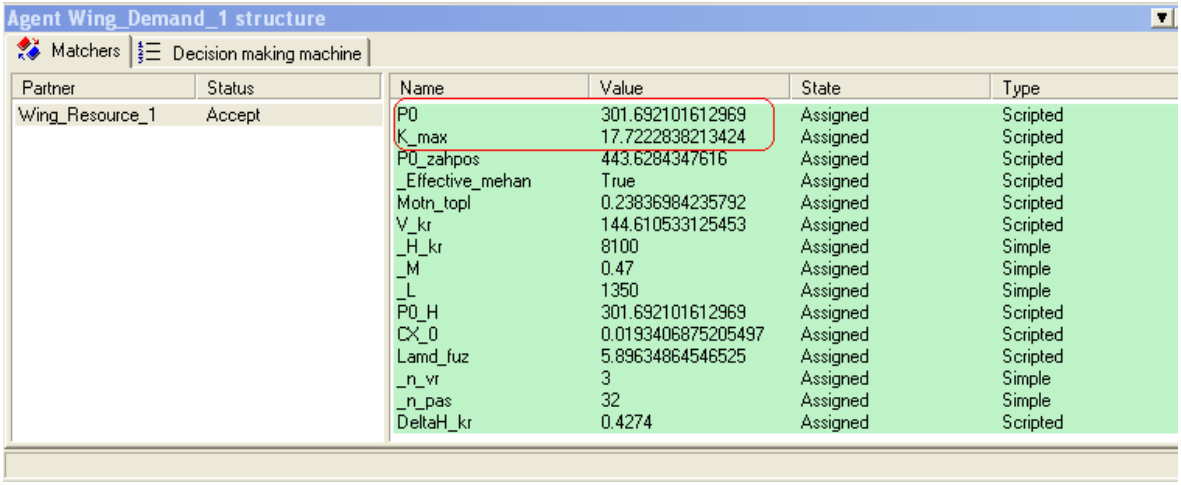

Figure 2.30 – Agent *Wing Demand_1* structure for Yak-40 aircraft

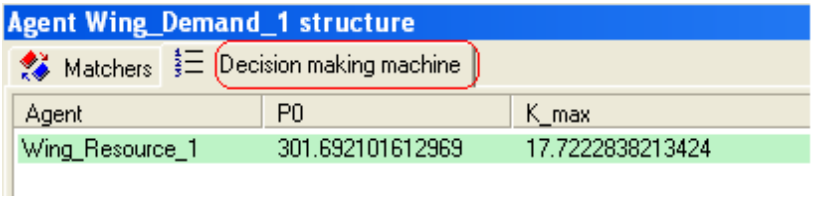

Figure 2.31 – The Decision Making Machine tab of the *Wing Demand_1* agent for Yak-40 aircraft

Save the scene as *Scene_Wing Load ЯК-40* and finish the work with the system.

2.2. Selecting a thrust-to-weight ratio for designing aircraft

2.2.1. Selecting the values of the required thrust-to-weight ratio of an aircraft

2.2.1.1. Thrust-to-weight ratio calculation from the condition of ensuring level flight at cruise speed and planned altitude

This value is defined from formula (2.12):

$$\overline{P_0^I} = \frac{1}{\xi \cdot \Delta H_{кр}{}^{0,85} 0.85 \cdot K_{крейс}}, \quad (2.12)$$

where:

- $\overline{P_0^I}$ - thrust-to-weight ratio from the condition of ensuring level flight at cruise speed and planned altitude;
- $K_{крейс}$ – aerodynamic efficiency in cruise mode;
- $\Delta H_{кр}$ - relative density of air at cruising altitude;
- ξ – coefficient that takes into account engine thrust changes at flight speed.

Aerodynamic efficiency in cruise mode is calculated from formula (2.13)

$$K_{крейс} = 0,8 * K_{max}, \quad (2.13)$$

where:

- K_{max} – maximum aerodynamic L/D ratio (is calculated in the first part of laboratory work 2).

Relative density of air at cruising altitude is calculated from formula (2.13)

$$\Delta H_{кр} = 0.8 - 0.000046\ H_{кр}, \quad (2.13)$$

where $H_{кр}$ – cruising altitude, m (initial data).

Coefficient that takes into account engine thrust changes at flight speed is calculated from formula (2.14)

$$\xi 1 = 1 - 0,32M + 0,4M^2 - 0,01M^3, \quad (2.14)$$

where M - Mach number (initial data).

2.2.1.2. Thrust-to-weight ratio calculation from the condition of the given runway length

Thrust-to-weight ratio from the condition of the given runway length is calculated from formula (2.15)

$$\overline{P_0^{II}} = \frac{1,26\,p_0}{C_{y\ \max\ взл} \cdot L_{разб}} + 0,09975 \quad , \tag{2.15}$$

where:

- p_0 - unit load on the wing (it is calculated in laboratory work 2), daN/m2;
- $L_{разб}$ - runway length (initial data), m;
- $C_{y\ max\ взл}$ – the aircraft maximum lift coefficient in the take-off configuration; its value is selected based on high-lift system:
  - For efficient mechanisation $C_{y\ max\ взл}$ = 2.5;
  - For weak (inefficient) mechanisation $C_{y\ max\ взл}$ = 2.0.

2.2.1.3. Thrust-to-weight ratio calculation from the condition of engine failure on take-off

Thrust-to-weight ratio from the condition of engine failure on take-off is calculated from formula (2.16)

$$\overline{P_0^{III}} = \frac{1,5 \cdot n_{дв}}{n_{дв} - 1}\left(0,083 + tg\theta_{\min}\right), \tag{2.16}$$

where:

- $n_{двиг}$ – number of engines on an aircraft, pcs (initial data);
- $tg\theta_{\min}$ - take-off climb gradient, rad.

Take-off climb gradient is defined as follows based on the number of engines:

Number of engines	Climb gradient
2	0.024
3	0.027
>3	0.030

2.2.1.4. Calculation of required thrust-to-weight ratio of an aircraft

Required thrust-to-weight ratio is calculated from formula (2.17)

$$P = \max\left(\overline{P_0^{I}}, \overline{P_0^{II}}, \overline{P_0^{III}}\right). \tag{2.17}$$

where:

- P – calculated value of required thrust-to-weight ratio;
- $\overline{P_0^{I}}$ - thrust-to-weight ratio from the condition of ensuring level flight at cruise speed $V_{крейс}$ and planned altitude $H_{крейс}$ is calculated from formula (2.12);

– $\overline{P_0^{II}}$ - thrust-to-weight ratio from the condition of the given runway length is calculated from formula (2.15);
– $\overline{P_0^{III}}$ - thrust-to-weight ratio from the condition of engine failure on take-off is calculated from formula (2.16).

2.2.2. **Designing a Descriptive Ontology**

2.2.2.1. Creating an ontology

Start the ontology constructor (the *OntCons.exe* file). Create a new ontology library (*File –> New*). Rename it to *OntologyLibrary_Draught*. Create a descriptive ontology of the Required thrust-to-weight ratio of an aircraft domain (*New Item -> Descriptive ontology*). Rename it to *Ontology_Draught*.

2.2.2.2. Creating an ontology

– **'Object' concept**

It is required to create two concepts 'Object' when the descriptive ontology of the Required thrust-to-weight ratio of an aircraft domain is created. The attribute names, which are the initial data for calculations, start with an underscore.

Create *Demand_Draught* and *Resource_Draught* objects *(Objects ->New Item -> Object).* Set three types of icons for these objects. Each object can have a specific list of attributes.

– **'Attribute' concept**

The Attribute concept is a value that characterises an object (quantification of characteristic).

– Create a **Boolean attribute,** *_Effective_mehan* (*Attributes -> New Item ->* Boolean Attribute), and rename it to _Effective_mehan.
– Create **Float attributes** (Attributes -> New Item -> Float Attribute) and rename them to:
  – _H_kr,
  – _M,
  – _P,
  – _Dist_razb,
  – _K_max,
  – DeltaH_kr,
  – Ksi,
  – K_kreis,
  – TG,
  – P0_1,
  – P0_2,
  – P0_3,

  - P0_max,
  - Temp.
- Create **Integer attributes** (New Item -> Integer Attribute) and rename them to:
  - _N_dvig,
  - X,
  - Y.

2.2.2.3. Creating relations among concepts

Set relations among the created concepts of the descriptive ontology of the Required thrust-to-weight ratio of an aircraft domain, i.e. indicate that the *Demand_Draught* object has the following attributes *_Effective_mehan, _H_kr, _M, _P, _Dist_razb, _N_dvig, _K_max, DeltaH_kr, Ksi, K_kreis, TG, P0_1, P0_2, P0_3, P0_max, Temp, X, Y,* and the *Resource_Draught* object has *X*,*Y* attributes.

The list of object attributes can be viewed in the properties of the Object concept.

As a result, the object has a list of attributes' names in the *Uses* tab; and in the *Used by* tab the attribute has a name of an object (objects) that uses this attribute. The properties of the *Demand_Draught* object are shown on figure 2.32 in the *Uses* tab. This object has *_Effective_mehan, _H_kr, _M, _P, _Dist_razb, _N_dvig, _K_max, DeltaH_kr, Ksi, K_kreis, TG, P0_1, P0_2, P0_3, P0_max, Temp, X, Y* attributes (corresponding relations are presented in the *Uses* tab). In the *Used by* attributes' tab their relations with the relevant object are shown.

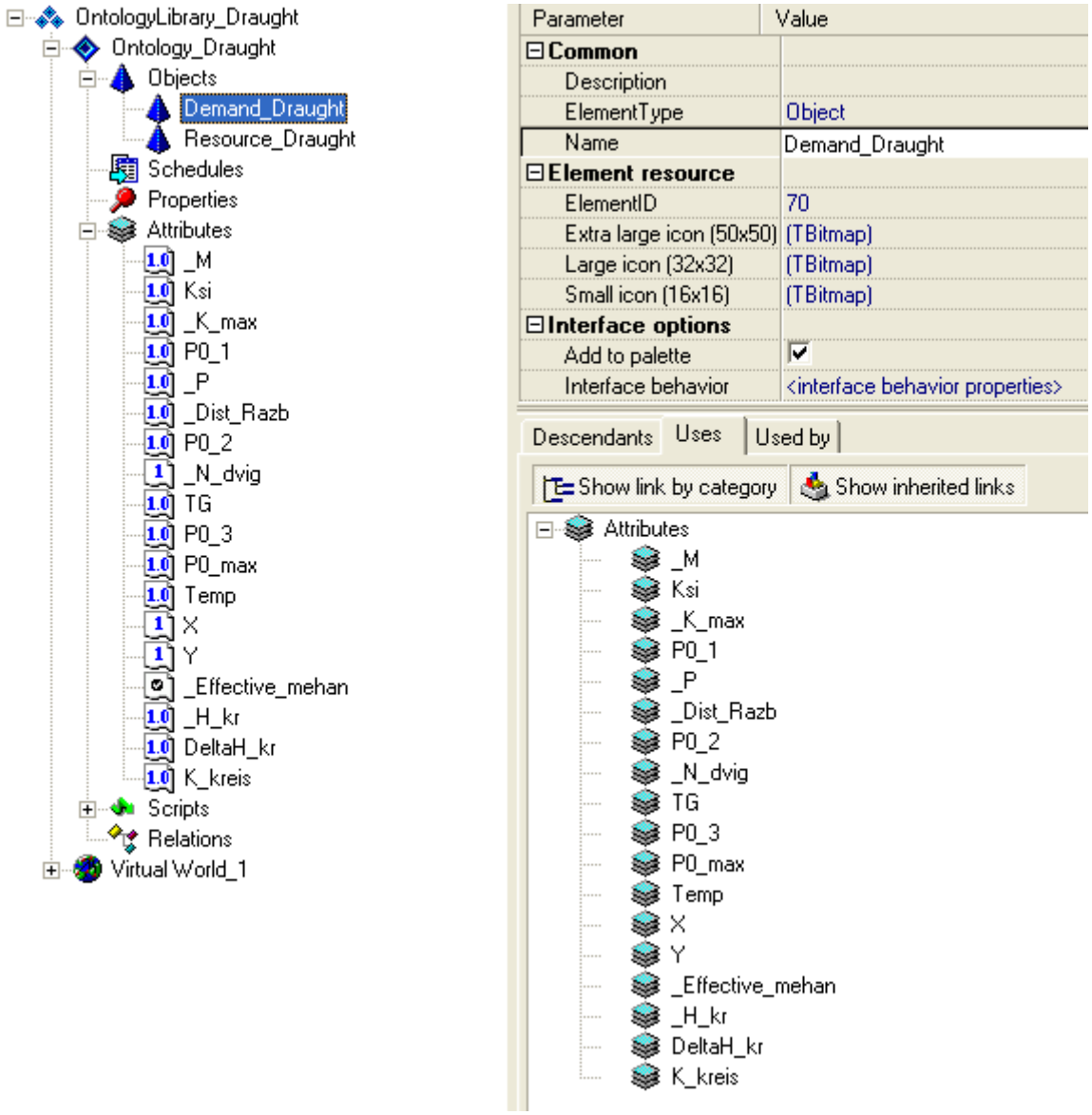

Figure 2.32 – Attributes of the *Demand_Draught* object in the *Uses* tab

– **'Script' concept**

Open Script and write, or edit the script body (*Other -> Script* in Script editor, or click the Edit script button in the *Script body* tab).

– ***Defining a script to calculate aerodynamic qualities in cruise mode***

To calculate aerodynamic quality in cruise mode from formula (2.12) it is necessary to create the Script concept. Select the *Scripts* category in the concept tree, then click *New item -> Script* in the shortcut menu. Rename the created script to *K_kreis Calculate* and connect it to the *K_kreis* attribute (i.e. drag the script onto the *K_kreis* attribute). Indicate parameters of the script: drug the necessary attributes onto the Script concept (all parameters of the script can be seen in the *Uses* tab). The *_K_max* attribute is the parameter of the *K_kreis Calculate* script that calculates aerodynamic quality in cruise mode (figure 2.33). Write the script body: select the *K_kreis Calculate* script in the concept tree and navigate to the *Script body* tab, click the Edit script button. The script editor pane opens. Type the following text (the concept names are quoted,

insignificant spaces at the beginning of the names are not allowed). The names of concepts should be selected from the list of concepts, which are the parameters of the script. The script ends with a semicolon.

```
begin
  result:= 0.8 * "_K_max";

end;
```

Click the  button to save the script. Close the Script editor window. Click the *Check syntax* button. If an error is found, it is necessary to open script editor and make corrections.

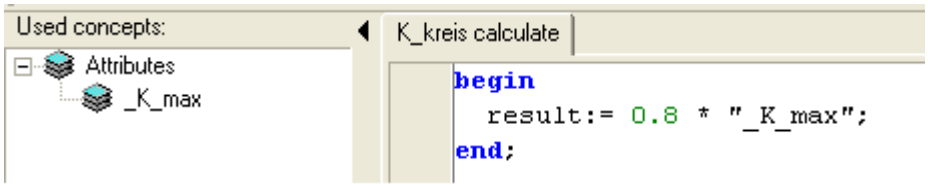

Figure 2.33 – Attributes and the script body of *K_kreis Calculate*

Similarly create other scripts.

– ***Defining a script to calculate relative air density at planned altitude (by formula 2.13)***

Rename the created script to *DeltaH_kr Calculate* and connect it to the *DeltaH_kr* attribute. The *_H_kr* attribute is the parameter of the *DeltaH_kr Calculate* script that calculates relative air density at planned altitude (figure 2.34).

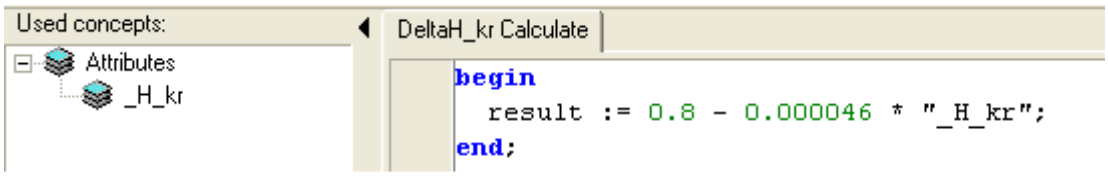

Figure 2.34 – Attributes and the script body of *DeltaH_kr Calculate*

– ***Defining a script to calculate coefficient that takes into account engine thrust changes at flight speed (by formula 2.14)***

Rename the created script to *Ksi Calculate* and connect it to the *Ksi* attribute. The *_M* attribute is the parameter of the *Ksi Calculate* script that calculates coefficient that takes into account engine thrust changes at flight speed. Write the script body (figure 2.35).

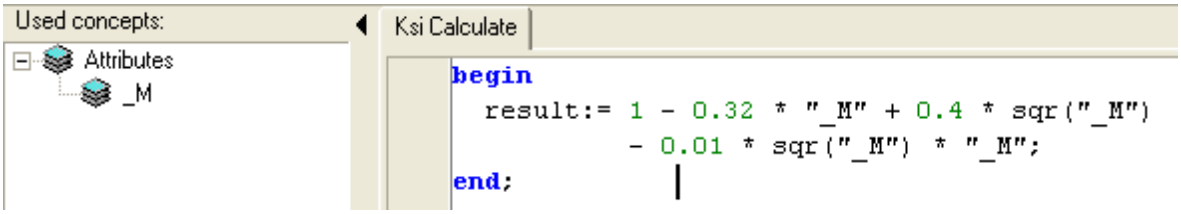

Figure 2.35 – Attributes and the script body of *Ksi Calculate*

– ***Defining a script to calculate thrust-to-weight ratio from the condition of ensuring level flight at cruise speed and planned altitude (by formula 2.12)***

Rename the created script to *P0_1 Calculate* and connect it to the *P0_1* attribute. The *Ksi*, *K_kreis* and *DeltaH_kr* attributes are the parameters of the *P0_1 Calculate* script that calculates thrust-to-weight ratio from the condition of ensuring level flight at cruise speed and planned altitude. Write the script body (figure 2.36).

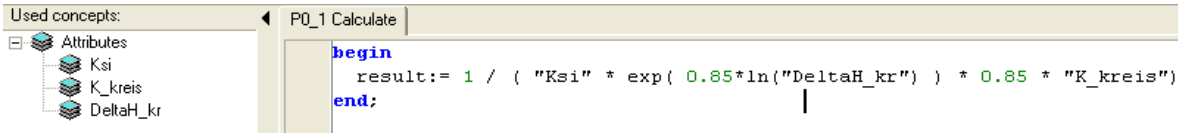

Figure 2.36 – Attributes and the script body of *P0_1 Calculate*

– ***Defining a script to calculate thrust-to-weight ratio from the condition of the given runway length (by formula 2.15)***

Rename the created script to *PO_2 Calculate* and connect it to the *PO_2* attribute. The *_P*, *Dist_razb*, *_Effective_mehan* attributes are the parameters of the *PO_2 Calculate* script that calculates thrust-to-weight ratio from the condition of the given runway length. Write the script body (figure 2.37).

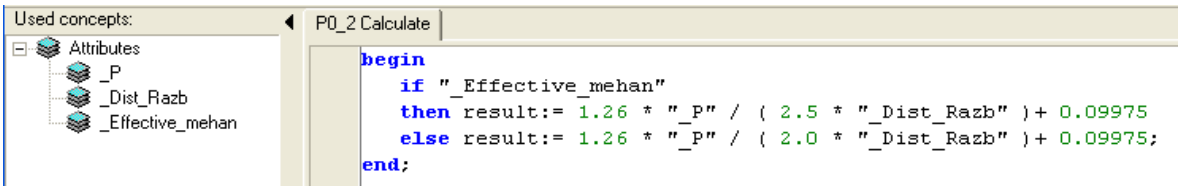

Figure 2.37 – Attributes and the script body of *PO_2 Calculate*

– ***Defining a script to calculate take-off climb gradient***

Rename the created script to *TG Calculate* and connect it to the *TG* attribute. The *_N_dvig* attribute is the parameter of the *TG Calculate* script that calculates take-off climb gradient. Write the script body (figure 2.38).

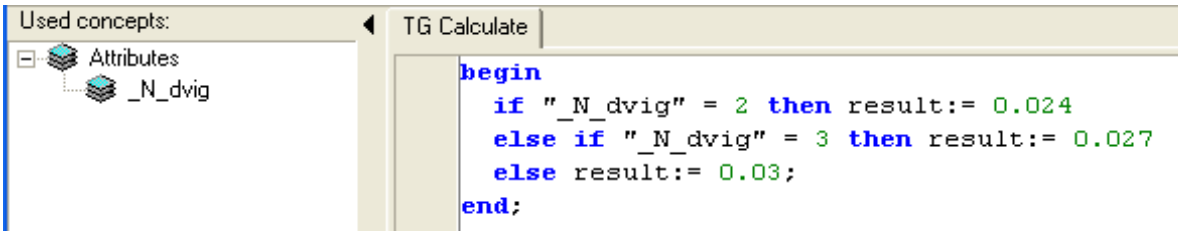

Figure 2.38 – Attributes and the script body of *TG Calculate*

– ***Defining a script to calculate thrust-to-weight ratio from the condition of engine failure on take-off (by formula 2.16)***

Rename the created script to *P0_3 Calculate* and connect it to the *P0_3* attribute. The *_N_dvig* and *TG* attributes are the parameters of the *P0_3 Calculate* script that calculates thrust-to-weight ratio from the condition of engine failure on take-off. Write the script body (figure 2.39).

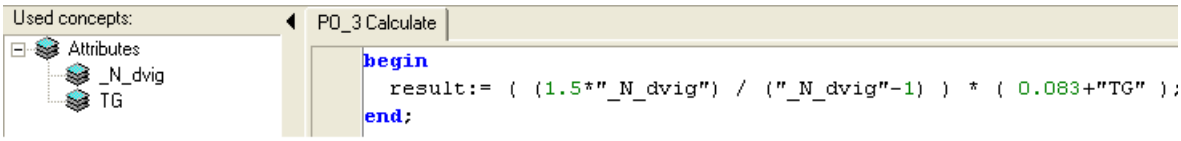

Figure 2.39 – Attributes and the script body of *P0_3 Calculate*

– ***Defining a script to calculate required thrust-to-weight ratio (by formula 2.17)***

Rename the created script to *PO_max Calculate* and connect it to the *PO_max* attribute. The *P0_1*, *P0_2*, *P0_3* attributes and auxiliary variable *Tem* are the parameters of the *PO_max Calculate* script that calculates required thrust-to-weight ratio. Write the script body (figure 2.40).

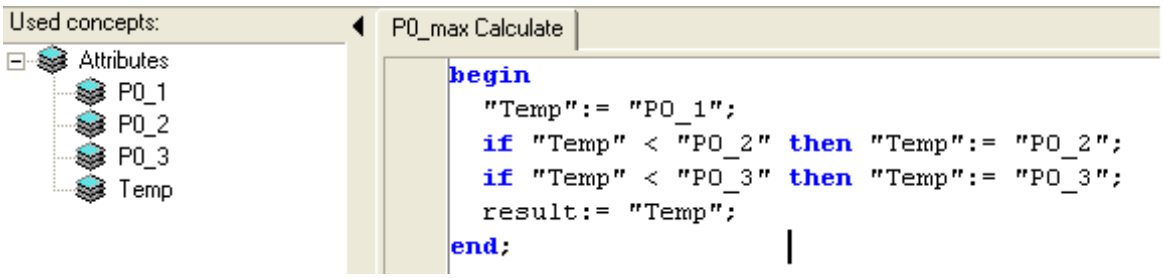

Figure 2.40 – Attributes and the script body of *PO_max Calculate*

Thus, the following scripts can be seen in the concept tree of the descriptive ontology of the Required thrust-to-weight ratio of an aircraft domain (figure 2.41).

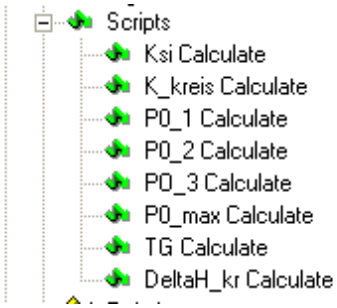

Fig 2.41 – Scripts of Descriptive ontology

2.2.2.4. Ontology representation in the Semantic Web

Descriptive ontology can be represented not just as a concept tree but also in the Semantic Web, that is a directed graph consisting of vertices, which represent concepts, and edges that represent relations among concepts. A user is able to move the concepts in the Semantic Web within the screen using the mouse.

To represent a descriptive ontology in the Semantic Web it is necessary to perform the following sequence of commands: *Tools -> Ontology as network* .

The descriptive ontology is represented as a concept tree on the left side of the *Ontology Network* pane, and on the right side it is shown in the Semantic Web (figure 2.42).

Closing the *Ontology Network* pane returns you to the ontology constructor.

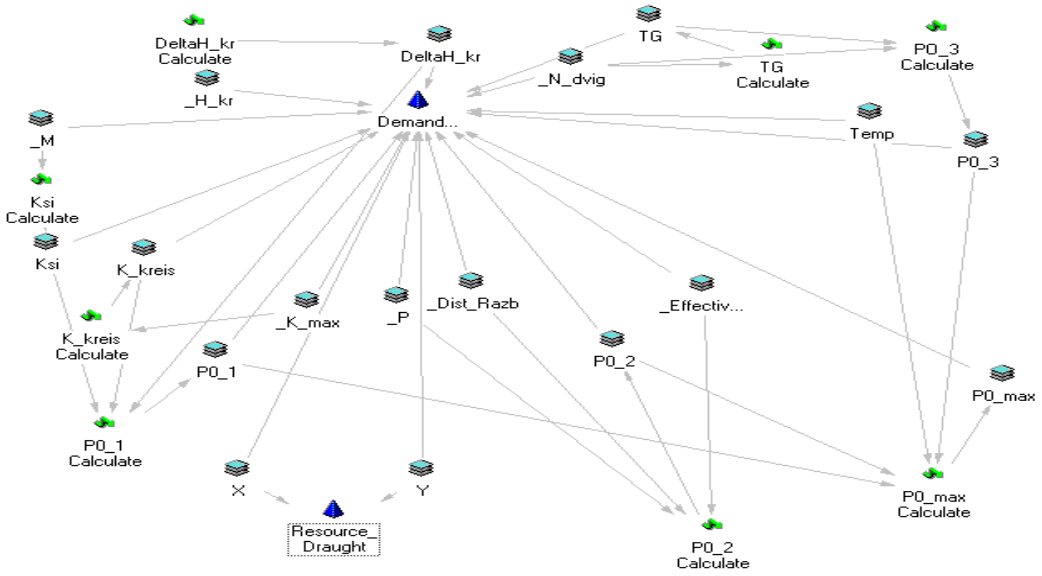

Figure 2.42 –Ontology representation of the Required thrust-to-weight ratio of an aircraft domain in the Semantic Web

2.2.3. **Modelling a world of demands and resources ontology**

2.2.3.1. Creating an ontology of demands and resources

To create an ontology of demands and resources (virtual world ontology) it is necessary to select an ontology library and click *New item -> Virtual world ontology* in the shortcut menu. A dialogue to create an ontology of demands and resources will appear on the right side of the screen. It allows selecting the objects that require creation

of demand or resource agents. If it is assumed that any object must have both a demand agent and a resource agent, tick the box from the left of this object. Agents are created automatically. If an object in the virtual world should correspond to either a demand agent or a resource agent, a box should not be ticked. Agents will be created later on an individual basis. Thus, in this example, object "designing aircraft" is a demand, so in the virtual world it should have only a demand agent. In turn, object-resource that is a resource should have only a resource agent. Clicking the <*OK*> button confirms the need to create an ontology of the virtual world (figure 2.43).

A pictograph of the virtual world ontology appears on the left side of the screen when clicking the <*OK*> button. A concept tree is opened by clicking <+>. It contains concepts of demand and resource agents as well as the relations among agents (figure 2.43).

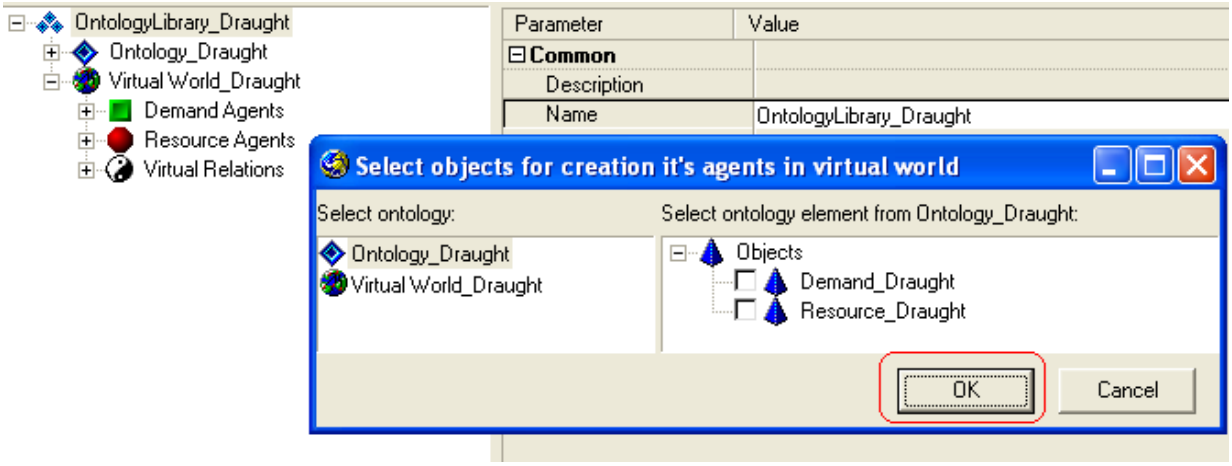

Figure 2.43 – Selecting objects for creation their agents

Create a virtual ontology for the Wing loading domain (*New Item -> Virtual World Ontology*). Type the name of the ontology as *Virtual World_Draught*. Disclose the concept tree of the virtual world ontology (figure 2.44).

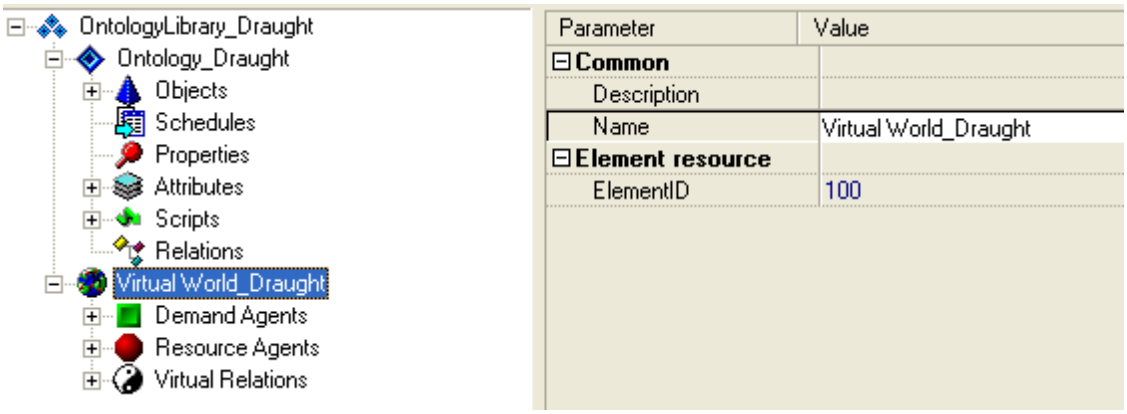

Figure 2.44 – The categories of the concepts of the virtual world ontology

2.2.3.2. Creating the Demand Agent concept

Create the Demand Agent concept for the "designing aircraft" concept (as it is active): select the *Demand Agents* category, click *New Item -> Demand agent* in the shortcut menu and select the *Demand_Draught* concept in the new dialogue. Then click <*OK*> and rename the created concept to *Demand_Draught* (figure 2.45). Set three

types of icons to the Demand Agent concept, which will be shown when working with the scene while modelling. Tick the *vaoAutoCreate* box (set by default).

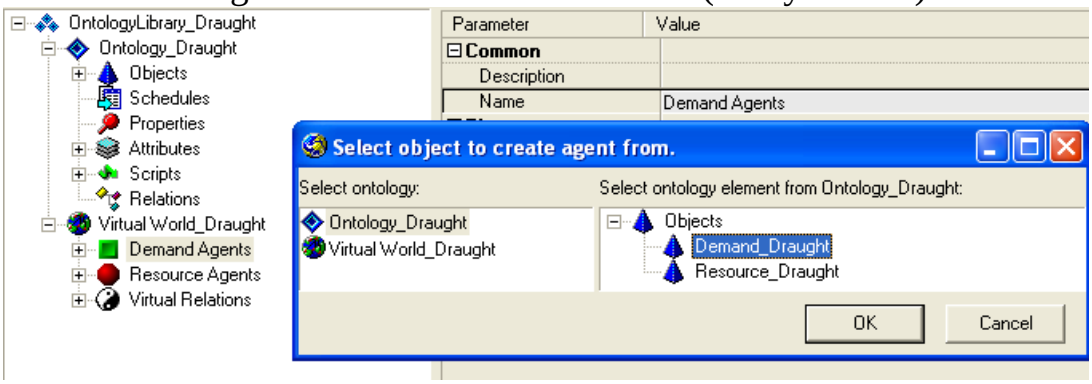

Figure 2.45 – Creating a demand agent for the *Demand_Draught* concept

2.2.3.3. Creating the Resource Agent concept

Create the Resource Agent concept for the "object-resource" concept: select the *Resource Agents* category, click *New Item -> Resource agent* in the shortcut menu and select the *Resource_Draught* concept in the new dialogue. Then click <*OK*> and rename the created concept to *Resource_Draught* (figure 2.46). Set three types of icons to the Resource Agent concept, which will be shown when working with the scene while modelling. Tick the *vaoAutoCreate* box, and do not tick the *raoActive* box.

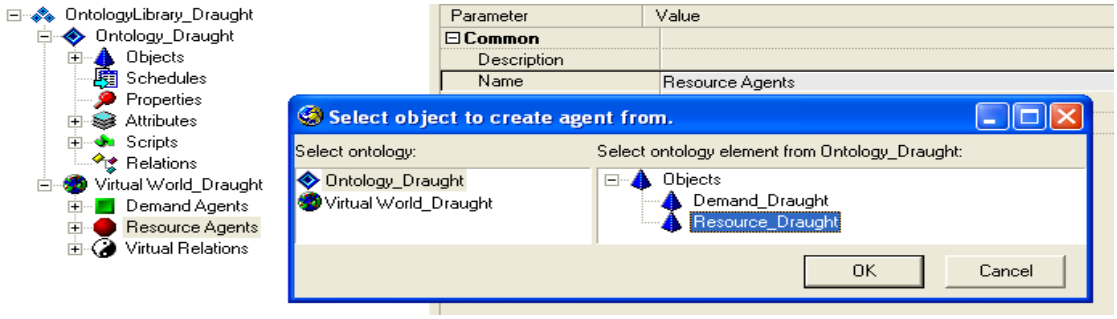

Figure 2.46 – Creating a resource agent for the *Resource_Draught* concept

2.2.3.4. Virtual relations. Matching relation

Matching relation is a utility class of relations in the virtual world. It connects demand and resources concepts. Matching relation shows the *possibility* of matching among agents, the concepts of which are connected by this relation in the ontology. In other words, matching is possible but it does not necessarily take place, because the agents may not agree by some reasons (there may be a more beneficial offer, this offer does not suit the partner/agent, etc.)

A Matching relation is only possible between the agents of demand and resource. For example, matching between agents of demand is impossible. A Matching relation is the subject-object relation. Subject is the initiator of matching. A demand agent and a resource agent can set matching relation in the scene. Either demand agent or resource agent can be the matching subject (if *raoActive* is set).

Set unilateral matching relation between the concepts of the *Demand_Draught* demand agent and the *Resource_Draught* resource agent. Select the *Matching relation* concept of the *Virtual Relations* category and click *Establish relation* in the shortcut

menu. On the right window disclose the tree of agents, and then its categories *DemandAgents* and *ResourceAgents*. Select *Demand_Draught* as the *Matching subject* and *Resource_Draught* as the *Matching object* (figure 2.47).

In the *Used by* tab of the Virtual relations: matching relation pane, it can be seen that the *Demand_Draught. Resource_Draught* matching relation is set (figure 2.47).

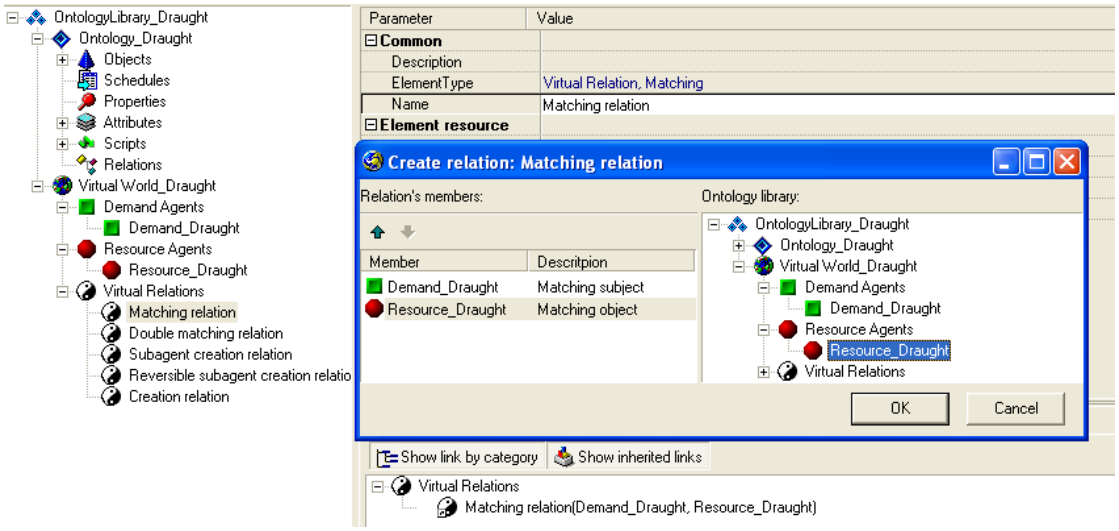

Figure 2.47 – Creating matching relation between *Demand_Draught* and *Resource_Draught*

2.2.3.5. Matching condition

Navigate to the *Used by* tab (*Virtual relations -> Matching relation -> Used by*) and select *Matching relation (Demand_Draught, Resource_Draught)*. Click *Edit virtual relation properties* in the shortcut menu (figure 2.48). The Edit matching condition pane appears (figure 2.49).

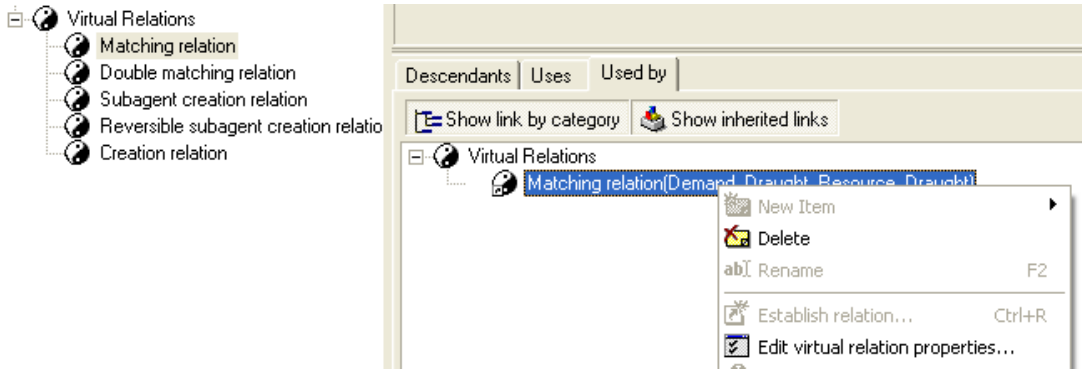

Figure 2.48 – Edit virtual relation properties

The Edit matching condition pane contains the following tabs:

– *Matching conditions:* creation and editing the matching conditions. Sign and script are described above. *Name* is a condition type (recorded automatically). *Checking agent* is an agent that checks the condition of matching, e.g. agent-subject (recorded automatically).

– *Decision Making Machine conditions:* creating and editing the criteria, based on which in matching a decision to reserve a resource agent by a demand agent is taken.
– *Tasks:* forming tasks to calculate additional attributes for a matcher (is not used in this laboratory work).
– *Events:* event handler which is run when it is required to change a value of an agent attribute depending on the attribute values in matcher (is not used in this laboratory work).

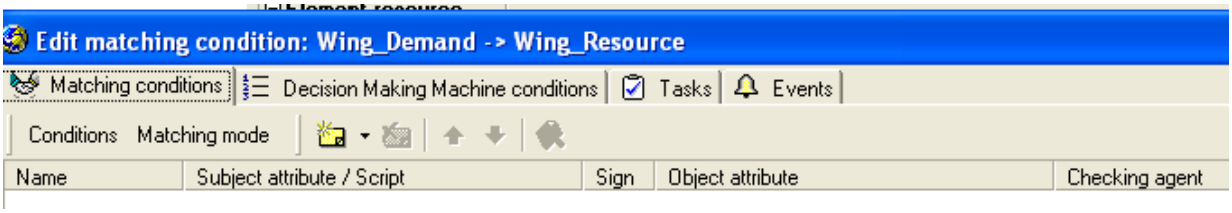

Figure 2.49 – Edit matching condition

In this task matching is needed just to run scripts that perform calculations, so special matching conditions are not required.

2.2.3.6. Decision Making Machine conditions

Decision Making Machine condition allows an agent to select an offer (matching) from other partners. It is created in the *Decision Making Machine conditions* tab of the *Edit Matching Conditions* pane by using the button. It is necessary to identify a condition attribute, optimisation (maximum/minimum) and weight coefficient that defines the importance of this condition.

2.2.3.7. Creating a decision making condition: choosing a maximum value of thrust-to-weight ratio

The maximum from one of the three values should be taken as the calculated value of the required thrust-to-weight ratio of an aircraft (formula 2.17): thrust-to-weight ratio from the condition of ensuring level flight at cruise speed and planned altitude (formula 2.12), thrust-to-weight ratio from the condition of the given runway length (formula 2.15) and thrust-to-weight ratio from the condition of engine failure on take-off (formula 2.16).

Create Decision Making Machine condition 1 for the *Wing_Demand -> Wing_Resource* matching: click the button in the *Decision Making Machine conditions* tab. Set the following parameters (figure2.50):

– Attribute = ‘Demand_Draught.P0_max’;
– Order = ‘Max’;
– Weight = ‘100’.

Tick the *Active* box to activate the decision making condition.

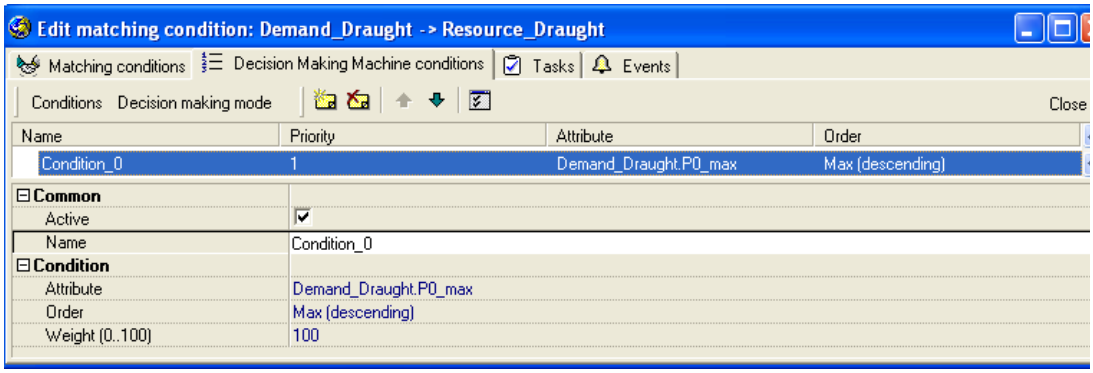

Figure 2.50 – Decision Making Machine condition for *Demand_Draught - Resource_Draught*

2.2.3.8. Interface options

The Interface options pane (figure 2.51) identifies the behaviours of an instance of the Object concept.

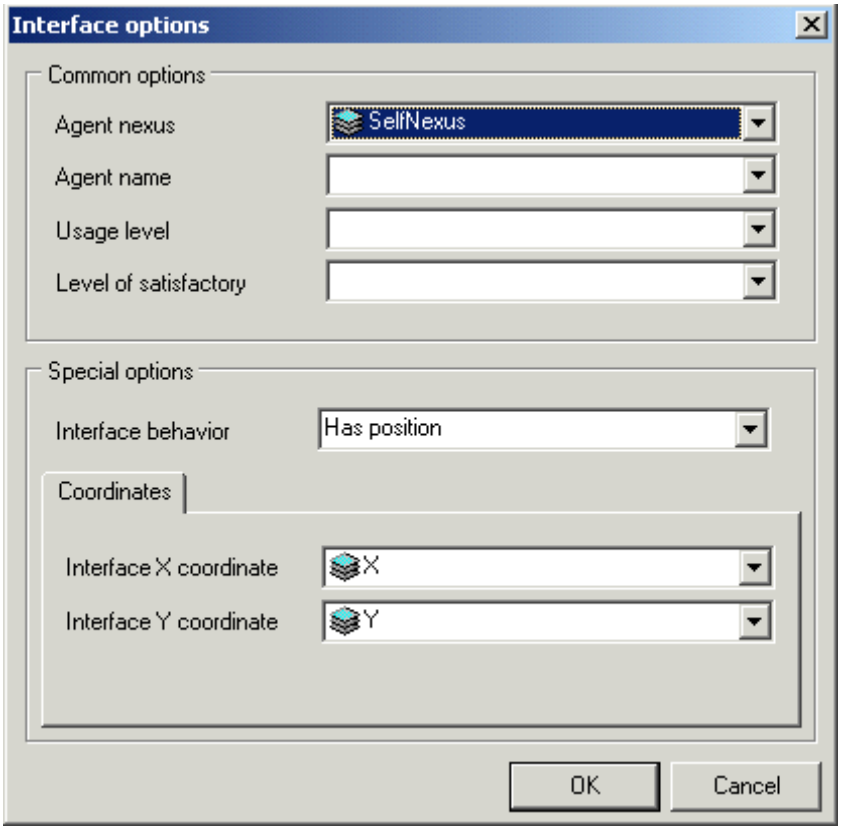

Figure 2.51 –Interface options

Connect the *X, Y* attributes with the agent position of the *Demand_Draught* object in the scene. To do so, open the *Interface behaviour* dialogue in the *Demand_Draught* properties, select the *Has position* parameter from the list. Then, select *X* and *Y* in the *Interface X coordinate* and *Interface Y coordinate* combo boxes respectively. Click <OK>.

Connect the *X, Y* attributes with the agent position of the *Resource_Draught* object in the scene. To do so, open the *Interface behaviour* dialogue in the *Resource_Draught*

properties, select the *Has position* parameter from the list. Then, select *X* and *Y* in the *Interface X coordinate* and *Interface Y coordinate* combo boxes respectively. Click <OK>.

2.2.3.9. Saving an ontology of the Required thrust-to-weight ratio of an aircraft domain

The concept tree of the descriptive ontology and the ontology of the virtual world of the Required thrust-to-weight ratio of an aircraft domain is shown on figure 2.52.

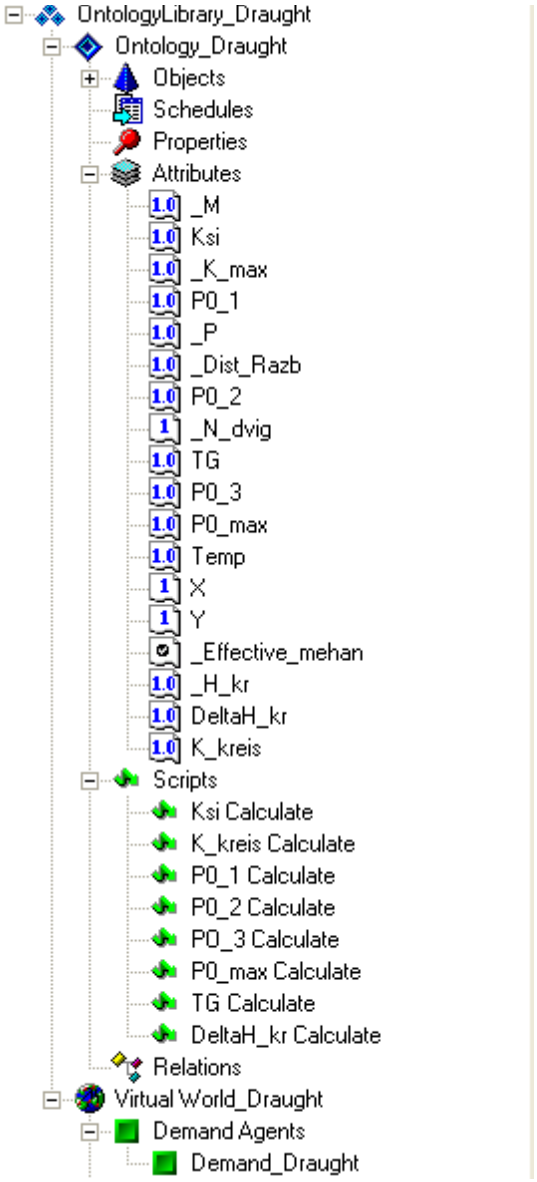

Figure 2.52 – The concept tree of the descriptive ontology and the ontology of the virtual world of the Required thrust-to-weight ratio of an aircraft domain

Save the created ontologies (descriptive ontology and demand and resource ontology) as *Ontology_Draught* by using the button. The *.ocl* extension is added automatically. The file is located in the *Ontology Samples* category by default. Finish the work with the ontology constructor (*File -> Close*).

2.2.4. **Creating an ontological scene**

Modify some settings on your computer: Start -> Control panel -> Regional and Language options -> Customise this format -> select a dot in the Decimal symbol drop-down list (figure 2.53).

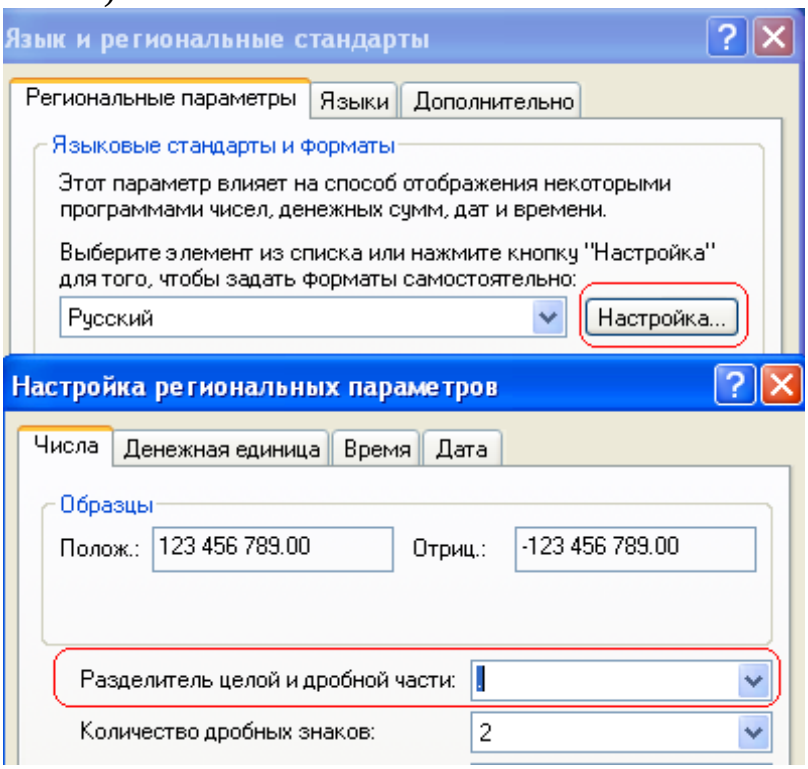

Figure 2.53 – Customising computer settings

Run the UniIntf.exe file that is located in the *OntConsUniIntf* folder. Create a new ontological scene (*File → New scene -> Load ontology*, select *Ontology_Draught.ocl).* In the Physical world pane create *Demand_Draught_1*, an agent of the designing aircraft, and *Resource_Draught_1*, an object-resource agent to perform calculations (figure 2.54). Set the following values of the attributes for agents (for example, for Tu-154 aircraft) using Agent inspectors.

Attributes	***Tu-154***	***Il-86***	***Yak-40***
_Effective_mehan	+	-	+
_P	**583,3**	**531,1**	**301,7**
_Dist_razb	1400	2000	750
_N_dvig	3	4	3
_K_max	**16,84**	**16,47**	**17,72**
_M	0,88	0,88	0,47
_H_kr	10100	10000	8100

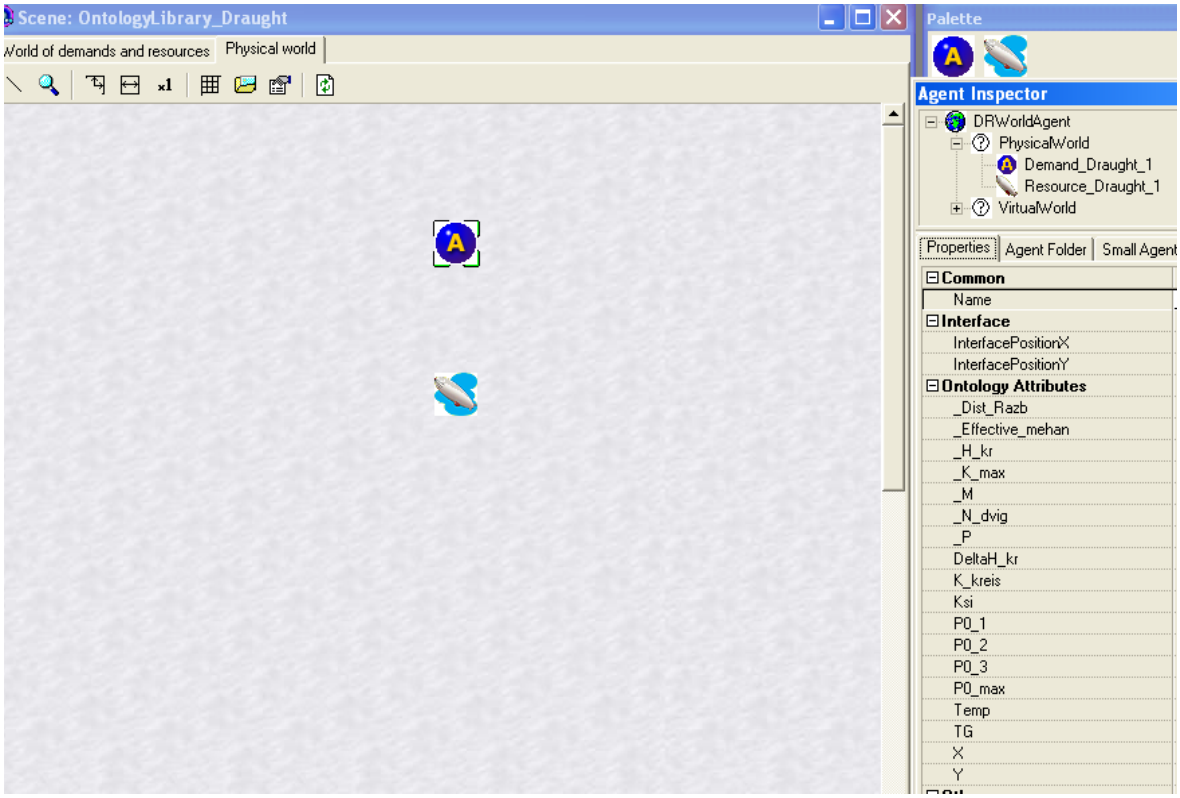

Figure 2.54 – Creating a scene of the Required thrust-to-weight ratio of an aircraft domain

2.2.5. **Modelling a virtual world scene**

2.2.5.1. Running the modelling scene for Tu-154

Navigate to the *Worlds of demands and resources* tab. Run the modelling scene (i.e. matching process) using the ⏩ button. Observe the matching process between the agents of the designing aircraft and the aircraft prototype agents in the database.

In the matching process the active agent of the designing aircraft runs scripts required to calculate the attribute values. Then the agent builds a decision making table, in which the only reserved object-resource agent is indicated. The table where the agent structure is presented (the Matchers tab) can be opened by double clicking the icon of the *Demand_Draught_1* agent in the physical world tab or the tab of world of demands and resources (or selecting *Actions -> Show agent structure* in the shortcut menu). The list of resources (partners), i.e. the agents that are reserved by the active agent, is indicated in the table (figure 2.55). When a partner is selected the attributes of the active agent are shown on the right side of the screen:

- *Simple:* the values are set in the scene;
- *Scripted:* the values are calculated using scripts during the matching process.

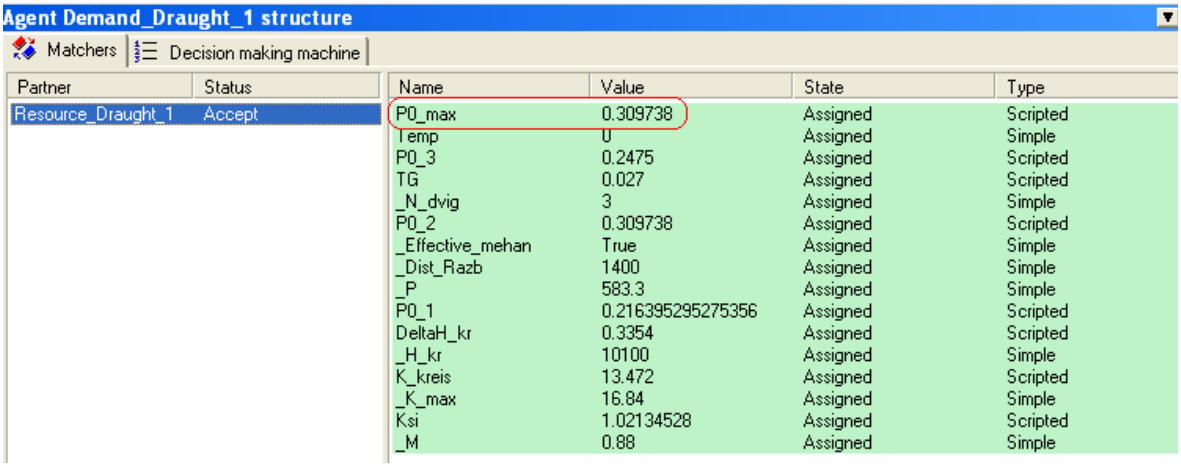

Figure 2.55 – Agent *Demand_Draught _1* structure for Tu-154 aircraft

The basic parameters that are calculated using scripts are shown in the *Decision Making Machine* tab (figure 2.56).

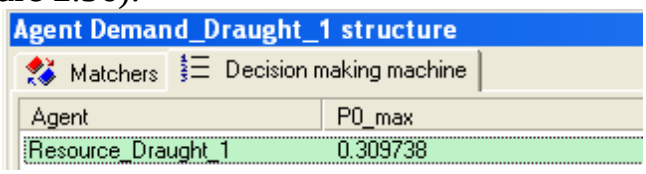

Figure 2.56 – The Decision Making Machine tab of the *Demand_Draught_1* agent for Tu-154 aircraft

In this task the agent-resource does not have attributes, that is why the table that contains its structure, is empty.

The final matching results can be seen on figure 2.57. As a result the following reservation operation was performed:

– *Demand_Draught_1 – Resource_Draught_1*.

Fig 2.57 – Matching results

2.2.5.2. Saving a virtual world scene

A scene can be saved by using the  button or by selecting *Tools - > Save ontology scene…* Save the scene as *Scene_Draught_TY-154*. The*.osf* extension is added automatically. The file is located in the *Ontology Samples* category by default. Finish the work with the system (*File -> Close*).

2.2.5.3. Loading an ontology scene

To load a scene click the button or select *Tools - > Load ontology scene…* Load the *Scene_Draught_Y-154.osf* file.

2.2.5.4. Modelling a scene for Il-86 aircraft

Clear the results of modelling by selecting *Tools -> Clear results*. Change the attribute values of *Demand_Draught_1* to Il-86 aircraft. Navigate to the *Worlds of demands and resources* tab. Run the modelling scene (i.e. matching process) by clicking the button. Observe the matching process between the agents of the designing aircraft and the aircraft prototype agents. The matching results are shown on figure 2.58, and the Decision making machine table is on figure 2.59.

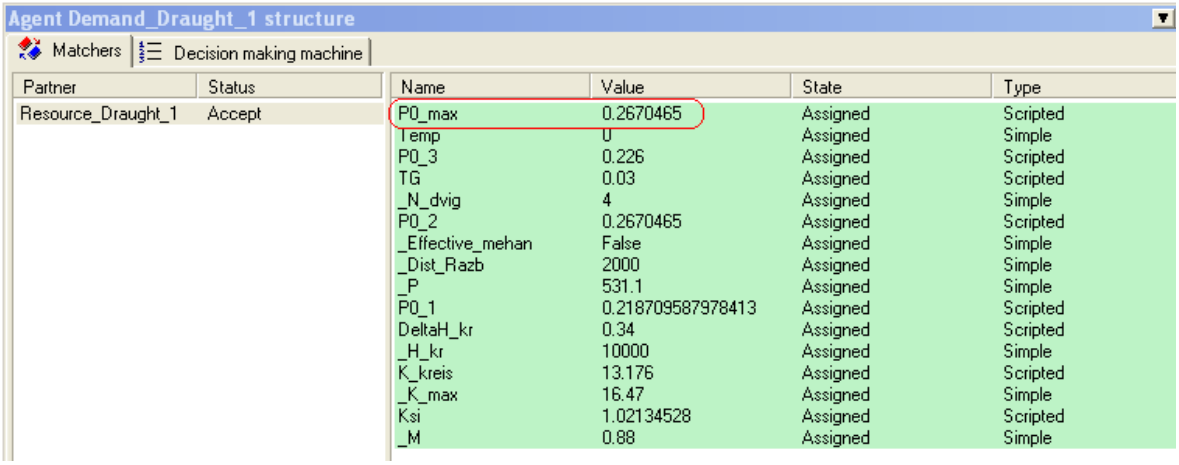

Figure 2.58 –Agent *Demand_Draught_1* structure for Il-86

The basic parameters that are calculated using scripts are shown in the *Decision Making Machine* tab (figure 2.59).

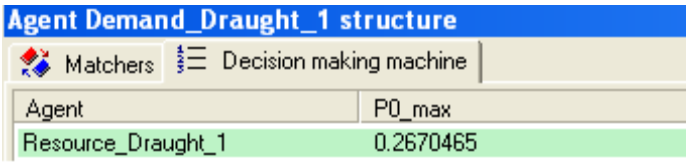

Figure 2.59 – The Decision Making Machine tab of the *Demand_Draught_1* for agent Il-86 aircraft

In this task the agent-resource does not have attributes, that is why the table that contains its structure, is empty. Save the scene as *Scene_Draught_ИЛ-86*. The*.osf* extension is added automatically. The file is located in the *Ontology Samples* category by default. Finish the work with the system (*File -> Close*).

2.2.5.5. Modelling a scene for Yak-40 aircraft

Load *Scene_Draught_TY-154.osf*. Clear the results of modelling by selecting *Tools -> Clear results*. Change the attribute values of *Demand_Draught_1* to Yak-40 aircraft. Navigate to the *Worlds of demands and resources* tab. Run the modelling scene (i.e. matching process) by clicking the button. Observe the matching process between the agents of the designing aircraft and the aircraft prototype agents. The

matching results are shown on figure 2.60, and the Decision making machine table is on figure 2.61.

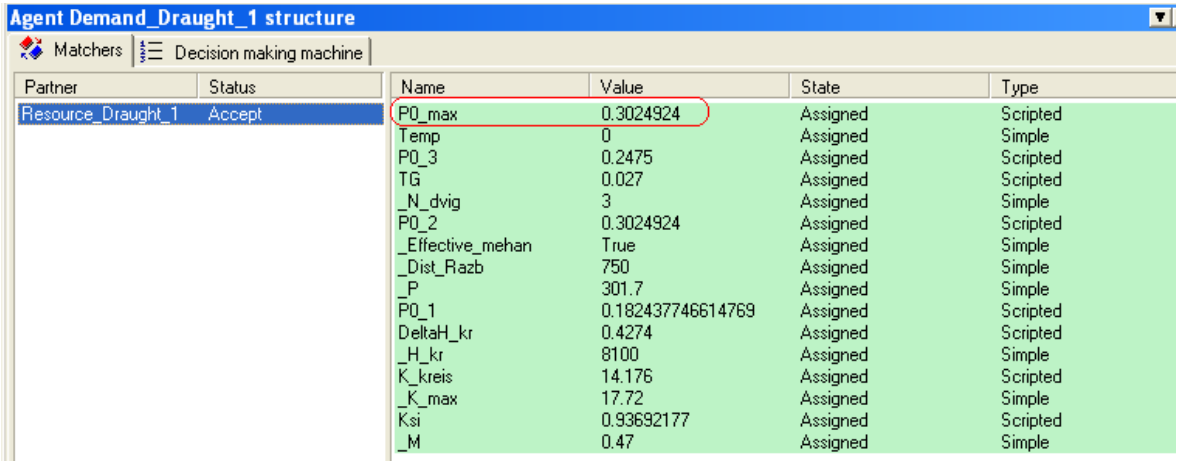

Figure 2.60 – Agent *Demand_Draught_1* structure for Yak-40

The basic parameters that are calculated using scripts are shown in the *Decision Making Machine* tab (figure 2.61).

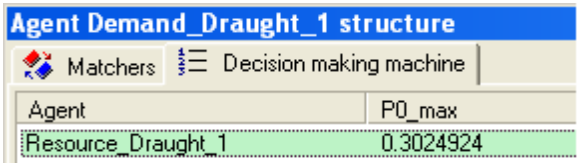

Figure 2.61 – The Decision Making Machine tab of the *Demand_Draught _1* agent for Yak-40

In this task the agent-resource does not have attributes, that is why the table that contains its structure, is empty. Save the scene as *Scene_Draught_ЯК-40*. The.*osf* extension is added automatically. The file is located in the *Ontology Samples* category by default. Finish the work with the system (*File -> Close*).

2.2.6. A QUESTION CHECKLIST

1. Define the following notions: an ontology, a descriptive ontology, ontology of the world of demands and resources, a concept, category of concepts.
2. What are the main functions of an ontology constructor?
3. Describe the structure of ontology library. Which main concept categories are presented in it?
4. What is the category and concept tree of ontology constructor for?
5. How to create a concept of descriptive ontology and to set its properties (by the example of the Object and Attribute concepts)? How to connect concepts?
6. How to define a script to calculate an attribute value? What are the functions of the Script editor?
7. How to check syntax? How to save a script?

8. How to create an ontology of the world of demands and resources and to set its properties? (by the example of the Demand and Resource concepts)?
9. How to set matching relation among the concepts of the world of demands and resources?
10. How to create an ontological scene with the use of instruments provided by the support system?
11. How to analyse the results obtained from executing the scripts? Consider the structure of agents.
12. How to set or to modify the attribute values using Agent inspector?
13. What should be done when modelling an ontological scene?

LABORATORY WORK 3

INTRODUCTION

The aircraft domain is considered as an example of ontological analysis with **Protégé 4.1 beta** ontology editor. This version allows building an OWL ontology in the Semantic Web. The OWL formal semantics specifies how to derive its logical consequences. i.e. facts not literally present in the ontology but entailed by the semantics.

OWL (Web Ontology Language) is ontology language for the Semantic Web. It is intended to describe the classes and relations between them that are inherent in Web documents and applications. OWL is based on OIL and DAML+OIL languages. It is endorsed by the World Wide Web Consortium (W3C).

Reality representation in the "object – property" data model is at the core of the language. OWL is used to describe not only Web pages but also any objects in reality. Each element of this language (including properties that connect objects) is associated with URI (Uniform Resource Identifier). First of all, it is, of course, the Internet resources and World Wide Web. URI provides a simple and extensible means for identifying resources.

3.1. GOALS AND OBJECTIVES OF LABORATORY WORK 3

Goals:

1. To master ontological analysis of the aircraft domain with the use of **Protégé 4.1 beta** ontology editor.
2. Consolidate skills in creating an ontology.

***Objectives*:**

1. Create a part of the aircraft ontology.
2. Develop the ontology according to the task.
3. Add values of instances to the ontology.
4. Present the result in the Semantic Web.

3.2. Creating an ontology

3.2.1. Creating a project

The work with Protege 4.1 beta as in Protege 3.3 starts with creating a new ontology or loading an existing one.

To create a new project click “Create new OWL ontology” (figure 3.1).

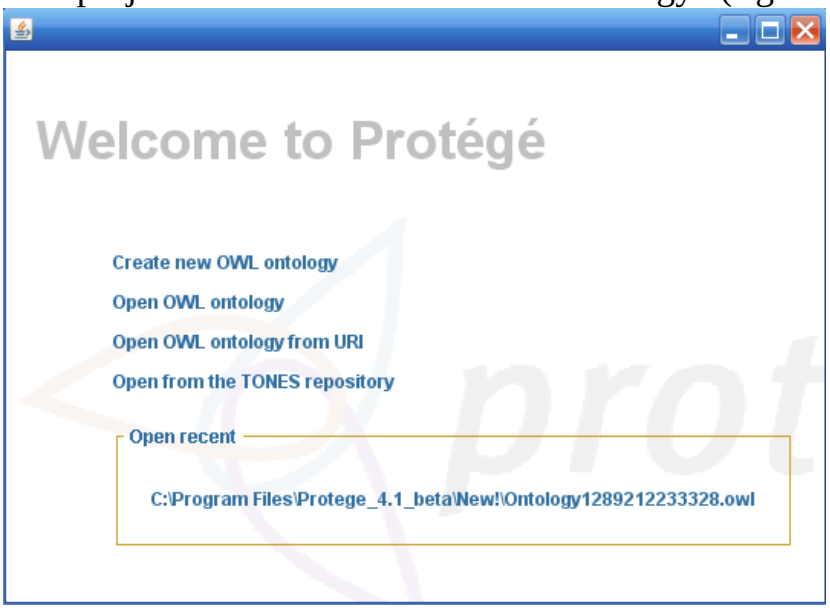

Figure 3.1 – Welcome window

Click the “Continue” button twice after the new window "Create ontology wizard" appears (Figure 3.2).

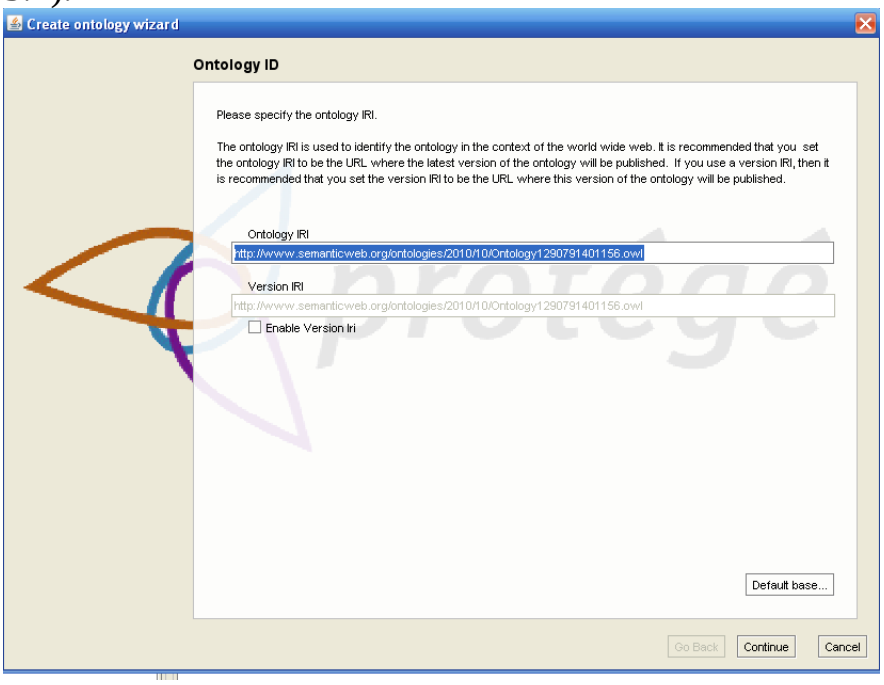

Figure 3.2 – “Create ontology wizard” window

Then select OWL/XML format and click the “Finish” button (Figure 3.3).

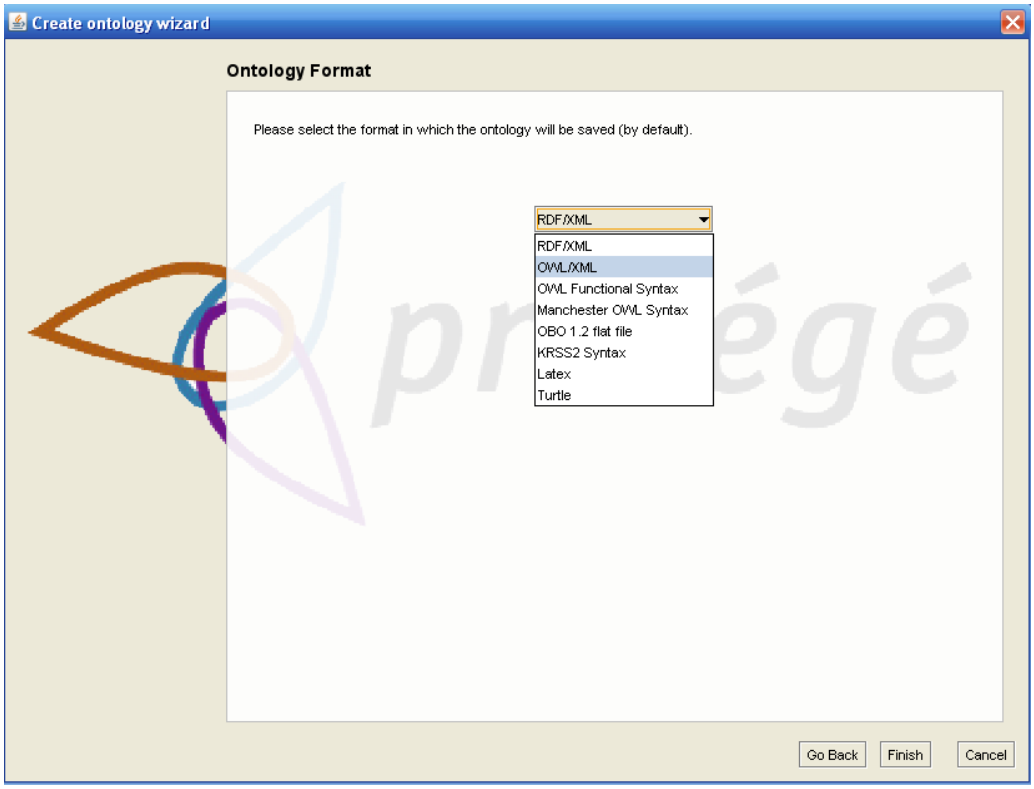

Figure 3.3 – Selecting the format

A Protégé project window will open (Figure 3.4).

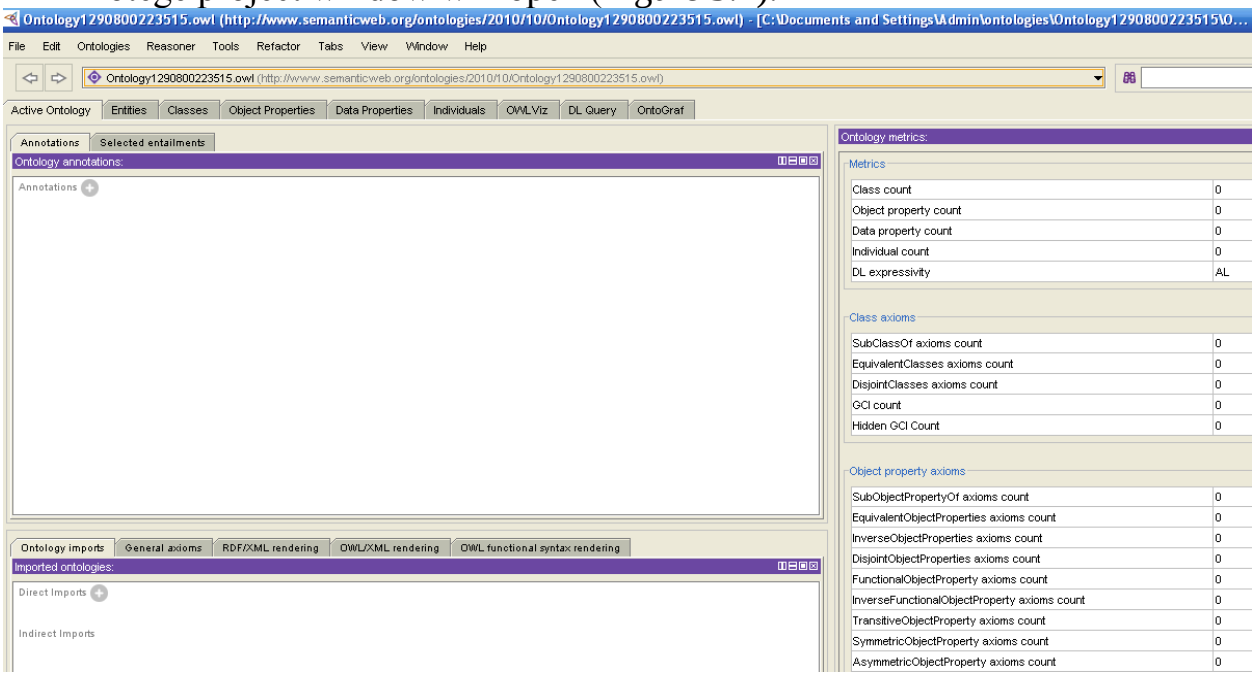

Figure 3.4 – A project window

3.2.2. Saving an ontology

A project can be saved in several ways:

1. Selecting “Save as…” from the File menu (Figure 3.5),
2. Clicking the Save button in the shortcut menu,
3. By using the keystroke sequence Ctrl+Shift+S.

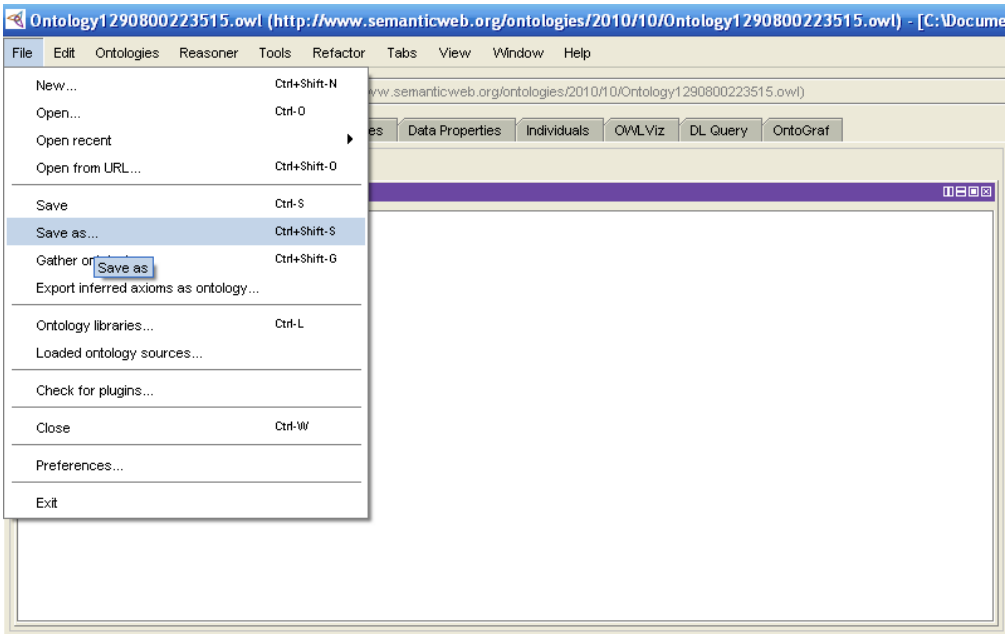

Figure 3.5 – Saving a project

Then select an ontology format and click OK (Figure 3.6).

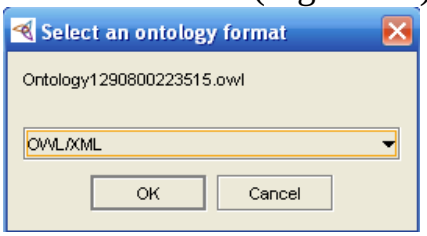

Figure 3.6 – Selecting an ontology format

Then a window for selecting a file name appears (Figure 3.7). Change the default name to *«LR3_varX»,* where X is your variant number. Select a directory to save the project and click OK.

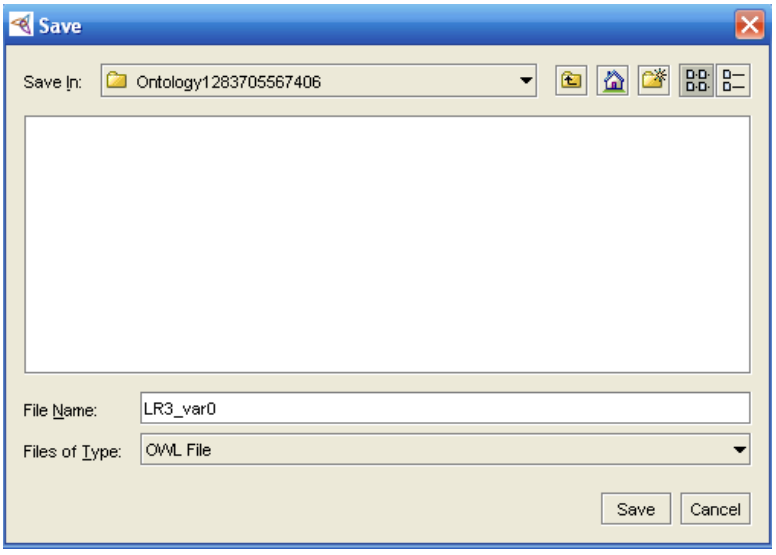

Figure 3.7 – File name window

3.2.3. Creating classes

The main Protégé window (Figure 3.8) consists of tabs that represent different knowledge models. The Classes tab is the most important when creating a new

ontology. The classes correspond to objects or types of objects in a domain. In the laboratory work, the Aircraft class only is created. The other classes are its subclasses.

The classes in Protégé are presented as a Class hierarchy which is located on the left of the Classes tab. On the right, in the Annotations and Usage tabs one can find information about the class.

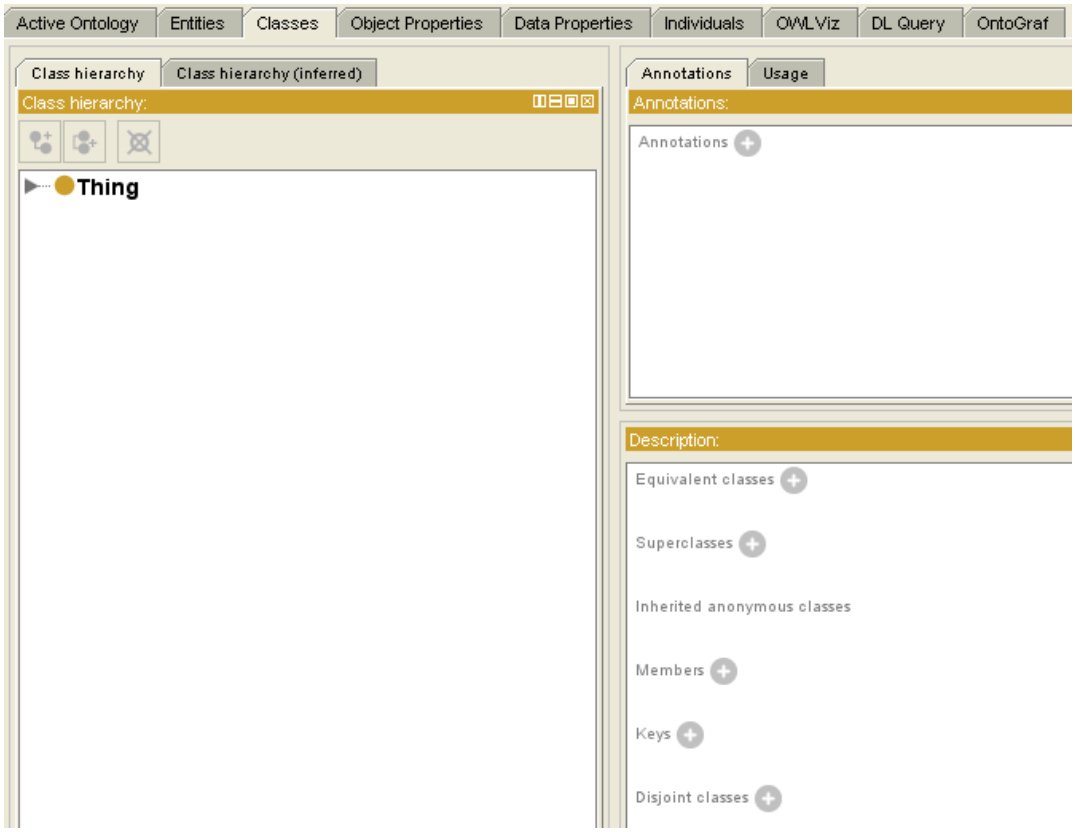

Figure 3.8 – The main Protégé window

3.2.4. Creating the Aircraft class

To create a new class select the Thing class in the Class Browser. All new created classes should be at a lower level of this system class.

Click the Add subclass button at the right corner of the Class Browser. Type the name of the class "*Самолет*" (Aircraft) in the appeared window "Create a new OWL Class" and click on OK (Figure 3.9).

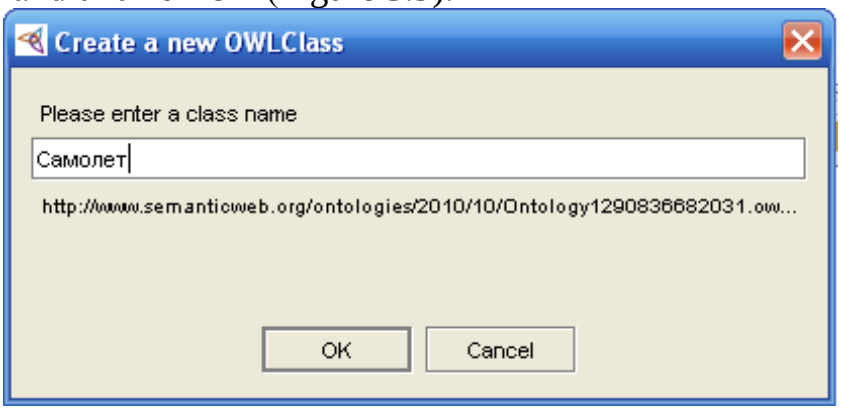

Figure 3.9 – Entering a class name

The new created class appears in the Class Browser. One can see, that the name of the new class is highlighted to indicate that the class is now selected (Figure 3.9).

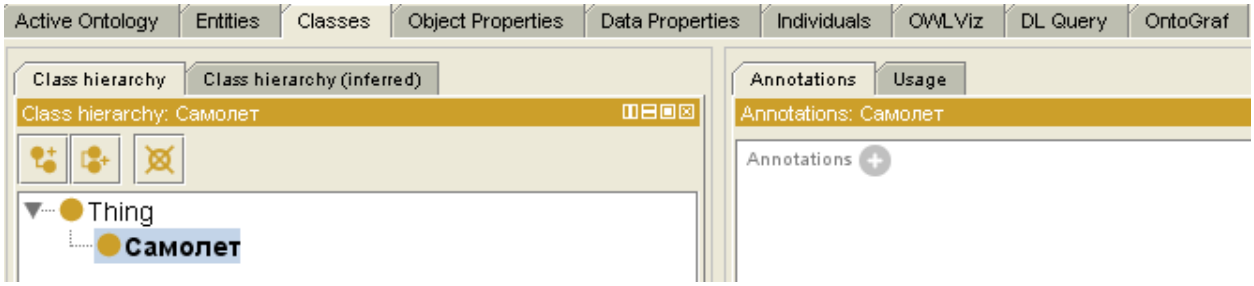

Figure 3.9 – The "Самолет" (Aircraft) class

3.2.5. Creating subclasses

The process of creating subclasses is similar to classes. Note the difference between the Add subclass button and the Add sibling class button. The button is for deleting classes.

To create a subclass select the "Самолет" (Aircraft) class. Click on the Add subclass button. In the appeared window "Create a new OWL Class" enter the name of the class and click OK.

Create a fragment of aircraft ontology [4, p.49-50]. Note that every element in ontology is a class. The classes that are at lower level than the "Самолет" (Aircraft) are highlighted, the other classes are separated by parentheses. The element that is before the parentheses is the class that is at a higher level than the element in the parentheses. Be careful, keeping the hierarchy. The phrase "and so on" means that there is just a part of elements of the class and that the ontology can be expanded.

Aircraft
(crew,
payload
(type *(*passengers, load, armament, equipment, etc*),*
character *(*jettisonable, off-load, non-removable*),*
parameters *(*quantity, mass, location, etc.*),*
transportation conditions
(comfort *(*number of stewards, number of on-board food stations, passenger cabin capacity, distance between the seats, aisle width, etc),
Security, etc.*),*
Location
(outside *(removable nacelle* *(on fuselage, on wing)),*
inside *(*in fuselage, in wing*)),*
fixation *(*cargo floor, seats, suspension units*),*
loading_unloading
(means
(airdrome mechanisation *(*power ladder, carrier, crane*),*
board_mechanisation *(*folding ladder, airstairs, hoist*),*
emergency funds *(*parachute, catapult*),*
character *(*loading time, unloading time*),*
places
(passenger doors *(*number, size, location*),*
Cargo doors, hatches*),*
And so on.),

Main load-bearing surfaces,
Power plant,
Stabilizer system,
Landing gear,
Connecting structure, etc.),

Only one class, “Payload”, is expanded in the hierarchy. It contains seven subclasses, which, in turn, have two to four classes, and not less than three levels in the hierarchy. Subclasses for other classes are not shown. It is required to similarly expand the ontology start with the class that corresponds to the task variant (see table 3.1) guided by the example [4] and the textbook [5]. The created ontology fragment is shown on figure 3.10.

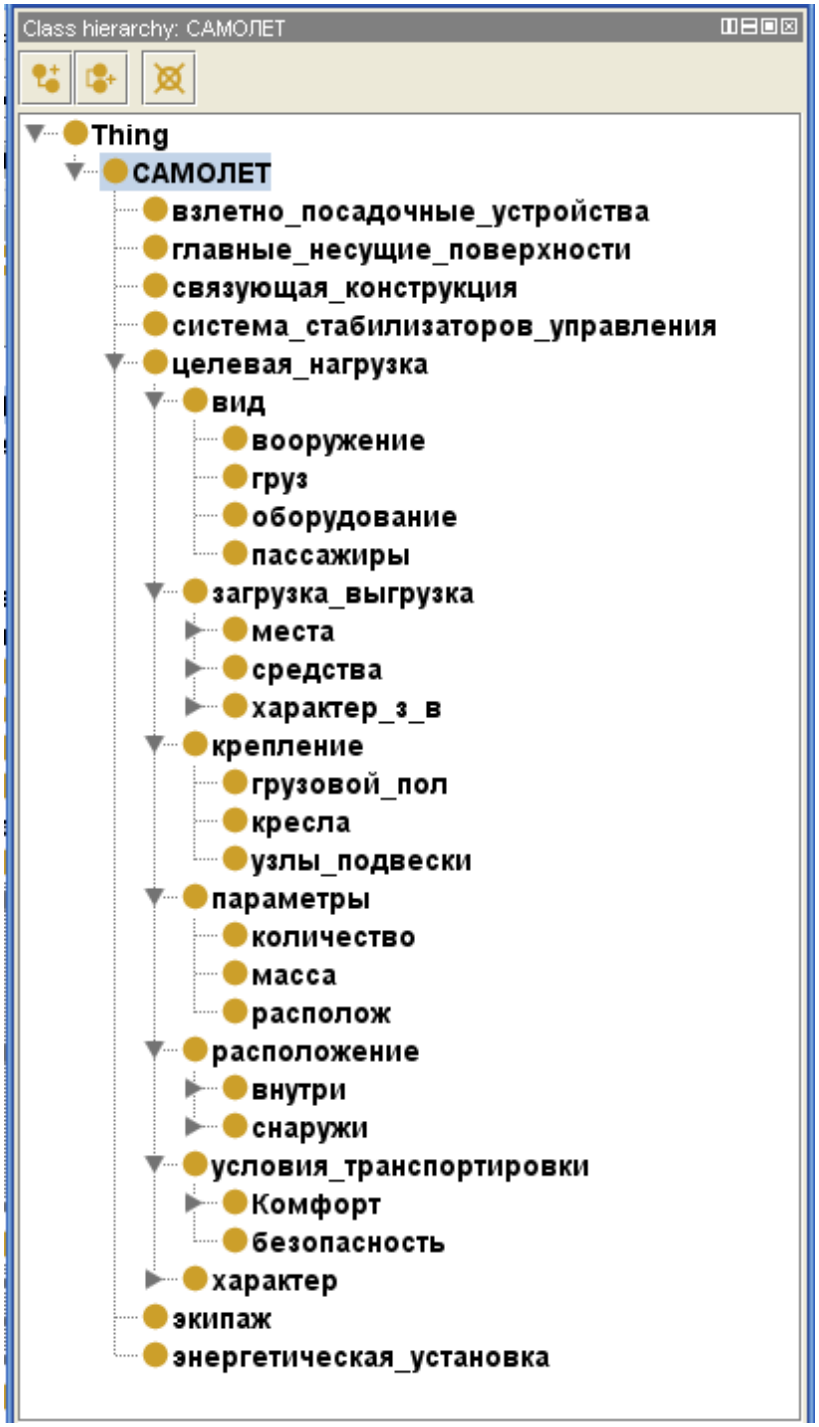

Figure 3.10 – Fragment of aircraft ontology

3.2.6. Creating superclasses

Creating a superclass is required in case of repetition of subclasses for several classes. The “Size” class is a universal parameter; it can be used to define dimensions (length, width, height, etc, see 3.2.7) of different parts of an aircraft, for example, a *door size*, or a *hatch size*. In the example this subclass should be in the “Cargo doors”, “Passenger doors”, and “Hatches” classes.

When the “Size” class is created in one of the mentioned above classes, for example, in the “Cargo doors” class (figure 3.11), a subclass with the same name cannot be created for other classes.

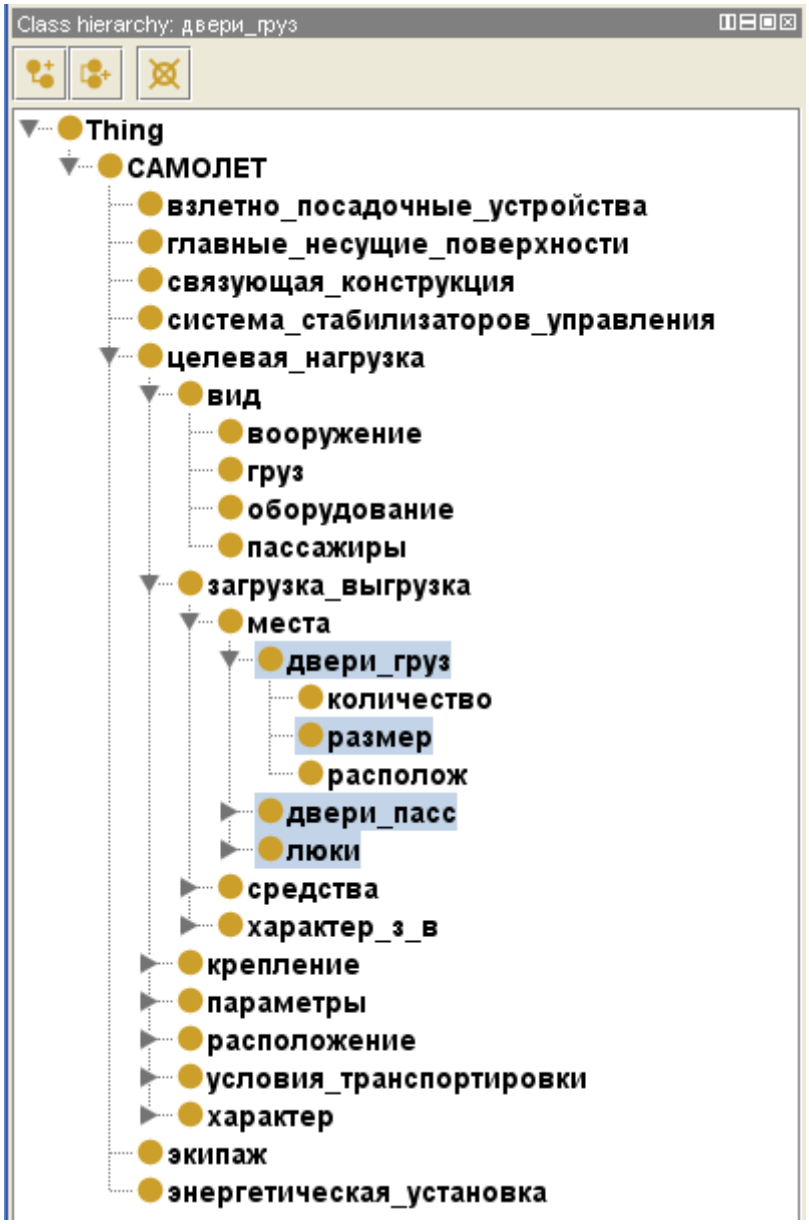

Figure 3.11 – The “Cargo doors” class containing the “Size” subclass

In order for the “Passenger doors” and “Hatches” classes to have the “Size” subclass, select “Size” in the hierarchy. In the Description pane click the button on the Superclasses bar. Then select the “Class hierarchy” tab and select the “Cargo doors”, “Passenger doors”, and “Hatches” classes holding the Ctrl key. Click on OK (Figure 3.12).

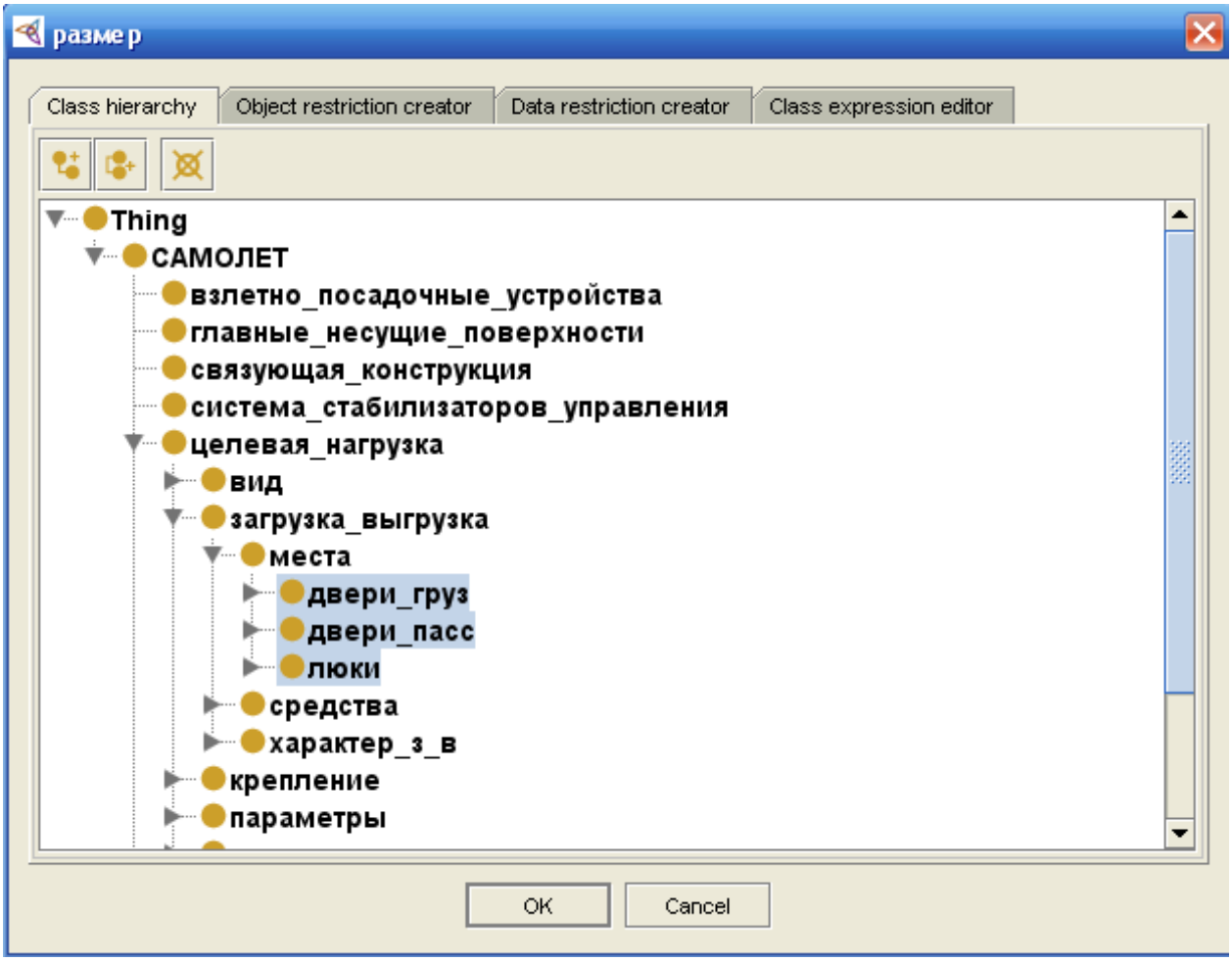

Figure 3.12 – Selecting superclasses

The classes that contain the “Size” subclass appear in the Superclasses bar (Figure 3.13).

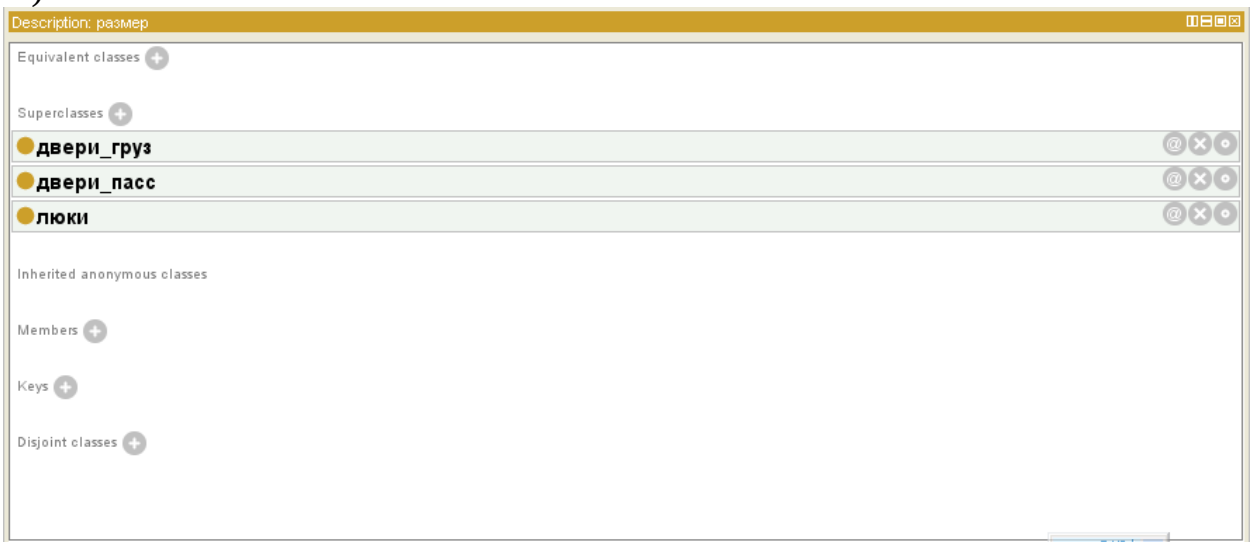

Figure 3.13 – Superclasses of the “Size” subclass

The “Quantity” and “Location” subclasses (figure 3.14) should also be included in the “Cargo doors”, “Passenger doors”, and “Hatches” classes.

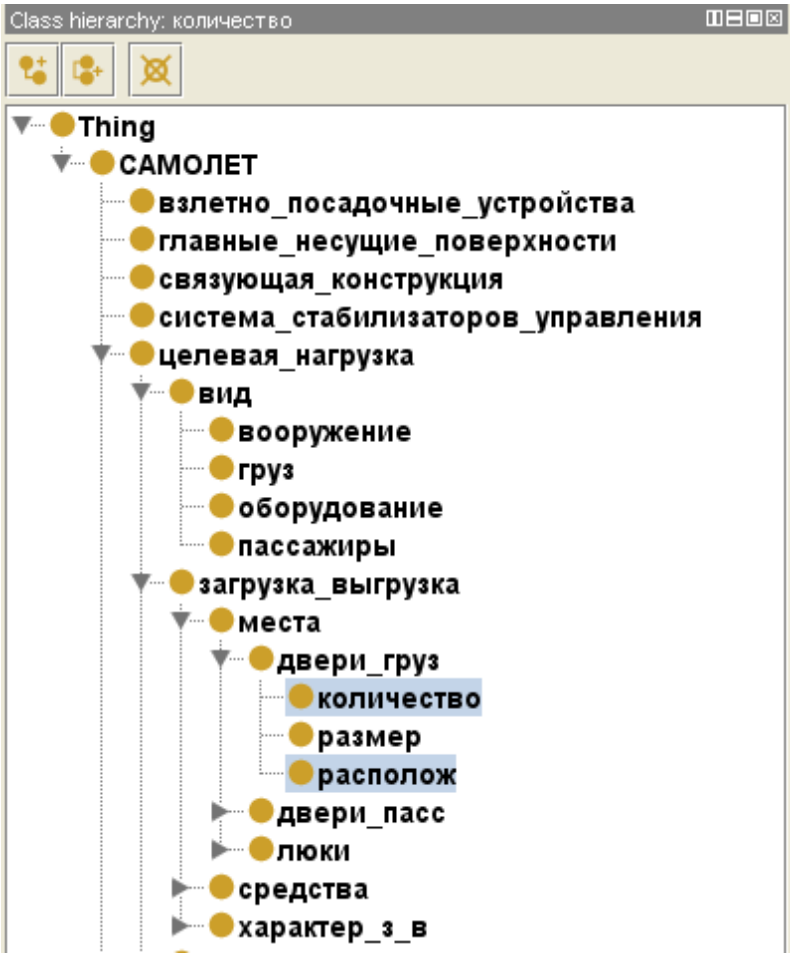

Figure 3.14 – The “Quantity” and “Location” subclasses

Similarly include the “Quantity” and “Location” subclasses in the “Cargo doors”, “Passenger doors”, and “Hatches” classes.

3.2.7. Creating object properties and data properties

It is similar to creating classes.

To add object properties click the  button in the “Object Properties” tab (Figure 3.15).

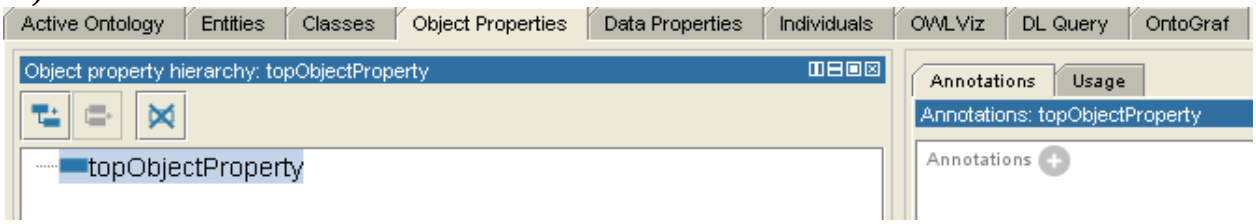

Figure 3.15 – The “Object Properties” tab

To add data properties click the  button in the “Data Properties” tab (Figure 3.16).

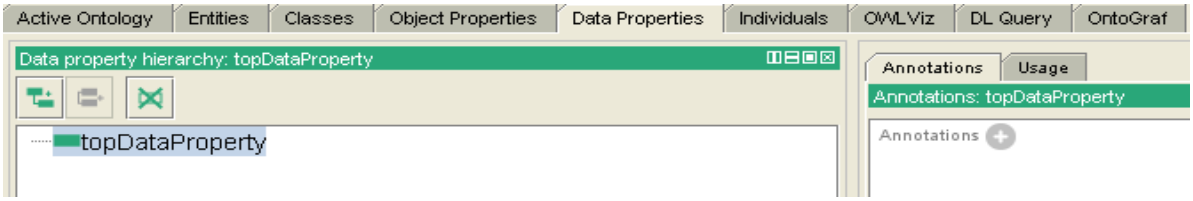

Figure 3.16 – The “Data Properties” tab

It is necessary to indicate sizes when creating instances. As an example for such objects as doors and hatches width and height parameters can be set. Create properties “hatch width”, “passenger door width”, “hatch height”, and “passenger door height” by clicking the  button in the “Data Properties” tab (Figure 3.17).

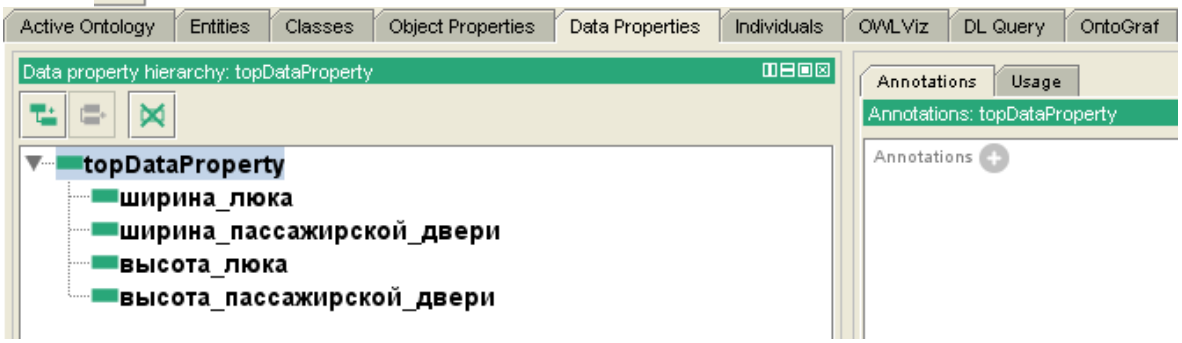

Figure 3.17 – Creating the “width” and “height” properties

3.2.8. Creating individuals

Creating individuals in Protege 4.1 beta is similar to Protege 3.3.

Navigate to the “Individuals” tab (Figure 3.18).

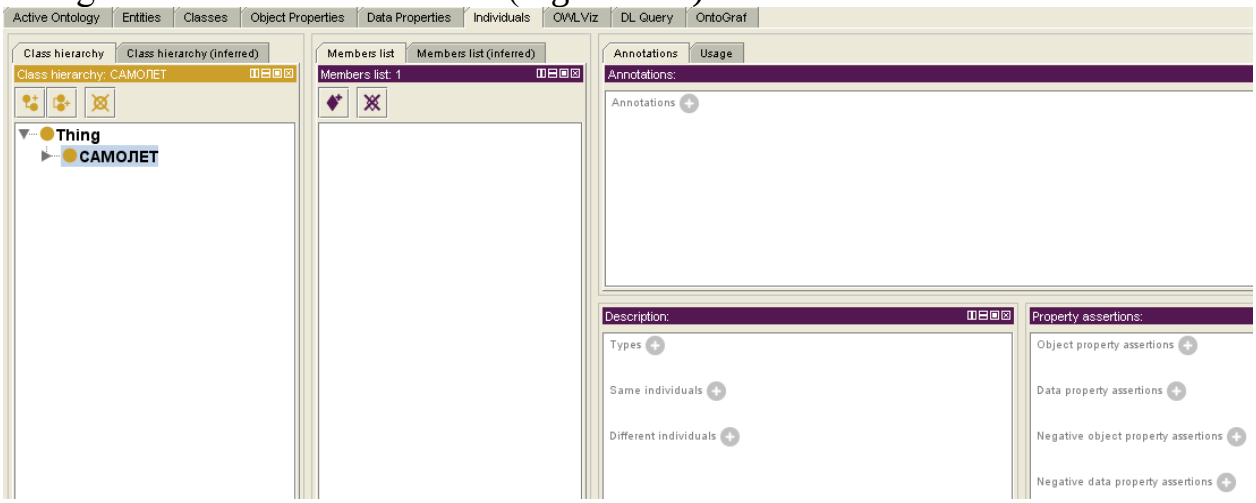

Figure 3.18 – Creating instances

Select the “САМОЛЕТ” (Aircraft) class in the hierarchy and click the  button. In the appeared window type the name «Ил-76тд» and click OK (Figure 3.19).

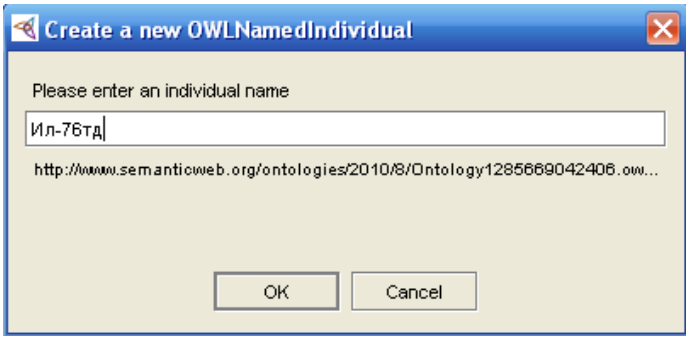

Figure 3.19 – Entering an individual name

Ontology describes the entire domain. However, the created instances may not be in all classes. For example, there are “Armament”, “Cargo”, “Equipment”, “Passengers” classes, and a cargo aircraft is created as an instance. In this case it is not included in the “Passengers” class.

The created instance Ил-76тд should be included in the selected classes. As an example the “Places” class and its subclasses “Loading_unloading” and “Payload” are considered.

Navigate to the Classes tab to select instances for a class. In the Description pane select the “Places” class on the Members bar and click the button. In the appeared window select Ил-76тд (Figure 3.20).

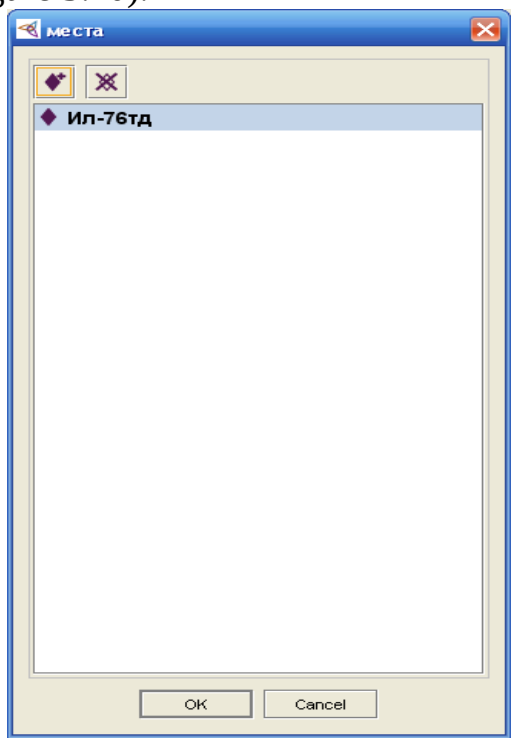

Figure 3.20 – Instance for the “Places” class

The name of the selected instance appears in the Members bar (Figure 3.21).

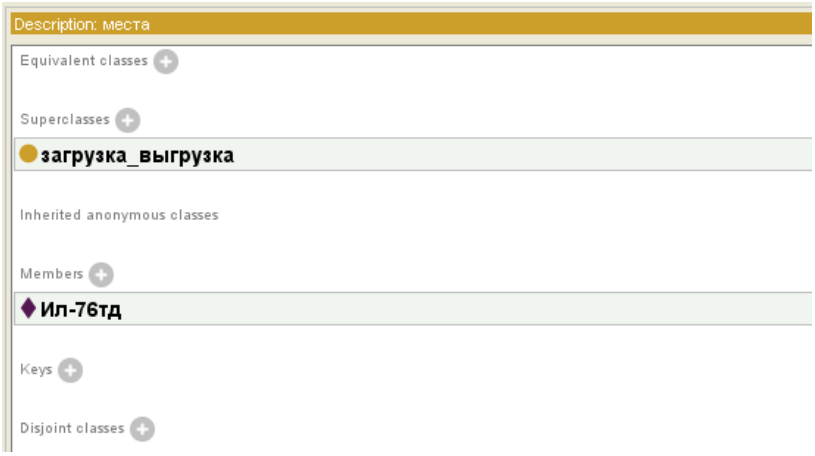

Figure 3.21 –Description pane: places

3.2.9. Setting values for instances

To set values for the instance, select the “Size” subclass of the “Passenger doors” class in the hierarchy. In the Description pane click the button on the Superclasses bar. In the appeared window navigate to the “Data restriction creator" tab. Select *«высота_пассажирской_двери»* on the left side of the screen and “integer” on the right side as a value type. Select *«Exactly»* from the down drop list and enter the value of passenger door height equal to *1900*. Click OK (Figure 3.22).

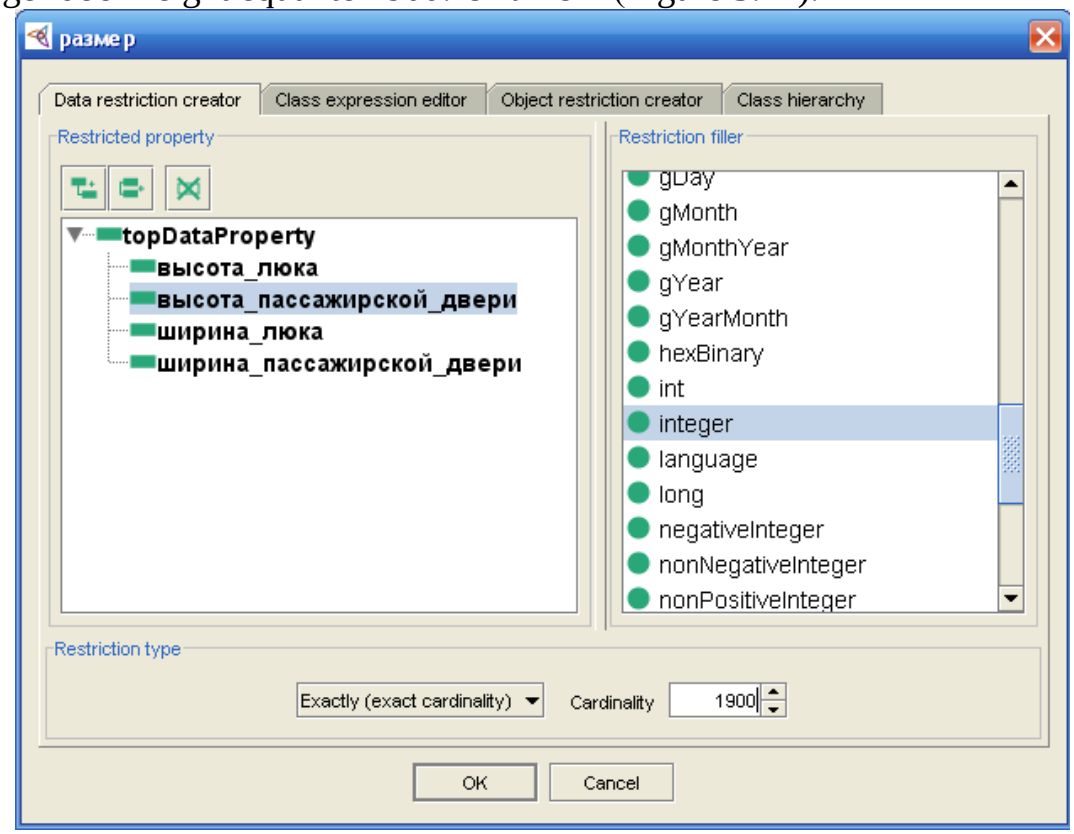

Figure 3.22 –Entering the value of passenger door height for Ил-76тд

Here, value types (integer, name, boolean, etc) and restrictions (Min, Max, etc.) can be chosen.

Similarly enter the value of passenger door width for Ил-76тд equal to 860 (Figure 3.23).

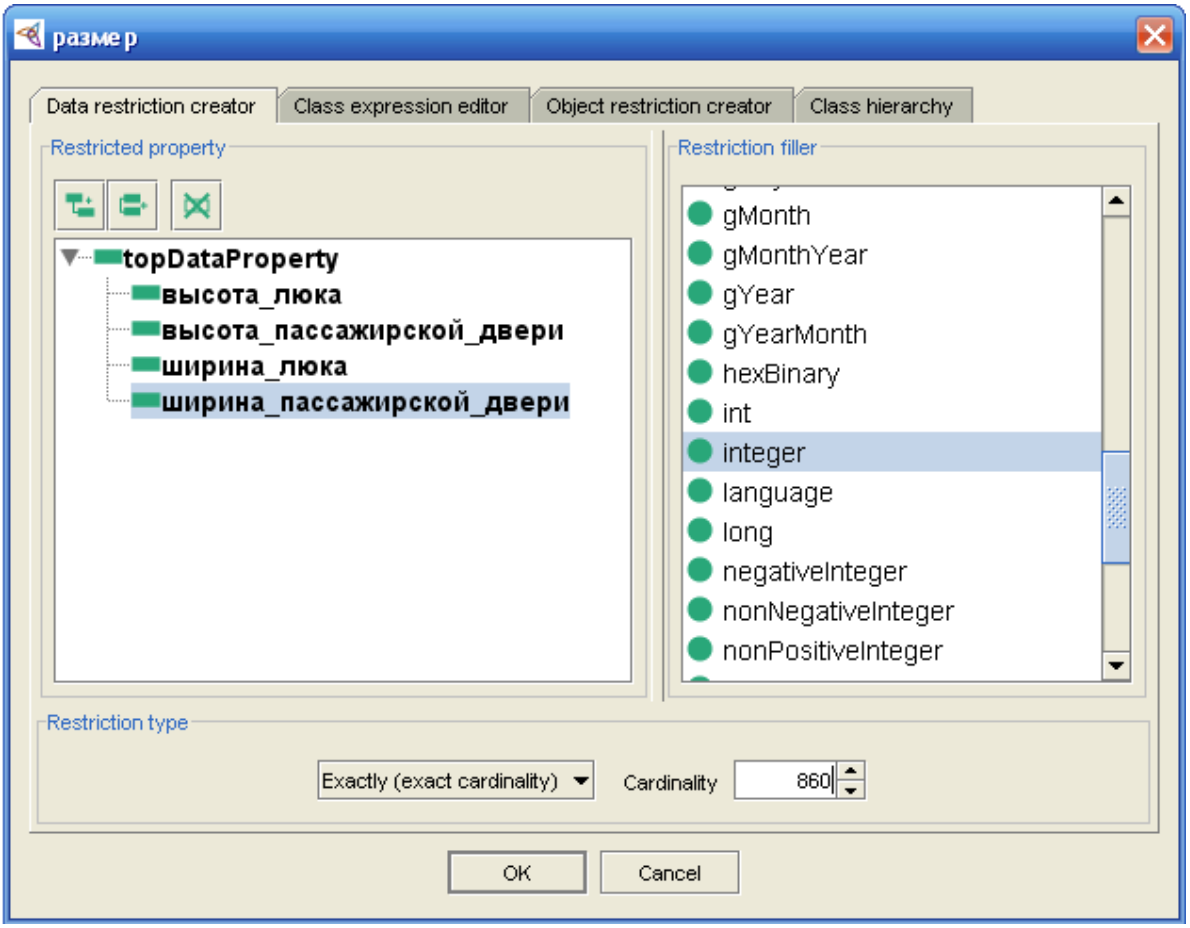

Figure 3.23 – Entering the value of passenger door width for Ил-76тд

After that, the Description: size pane will appear as shown on figure 3.24.

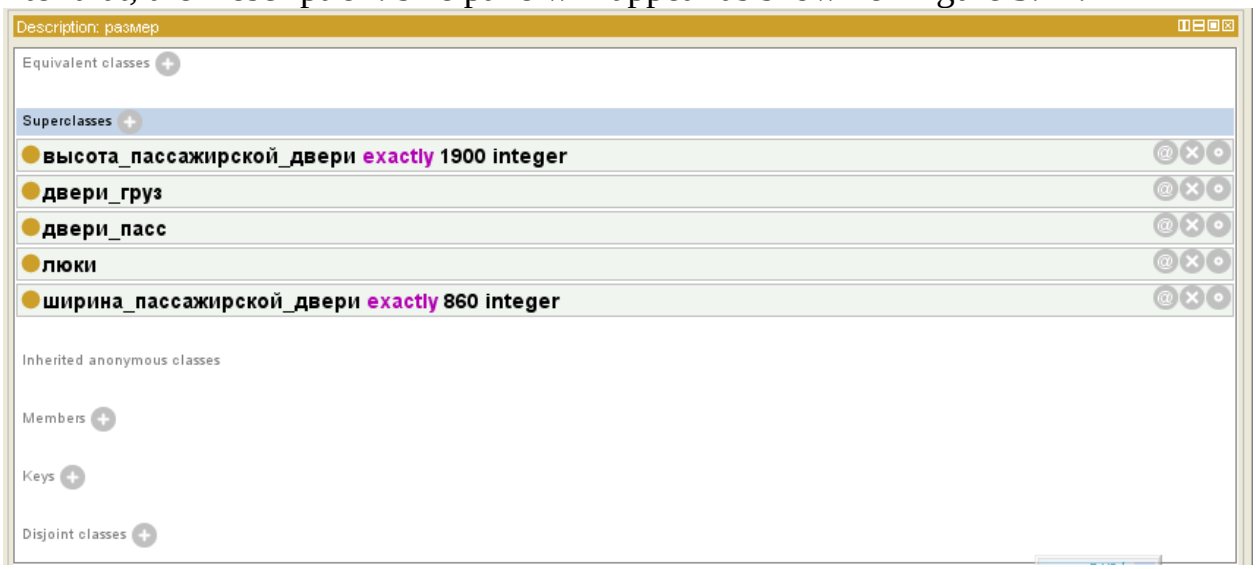

Figure 3.24 –Description: size pane

3.2.10. Representation in the Semantic Web

Navigate to the OntoGraf tab and select “САМОЛЕТ” (Aircraft) in the hierarchy. The «Самолет» (Aircraft) button appears on the right side (Figure 3.25).

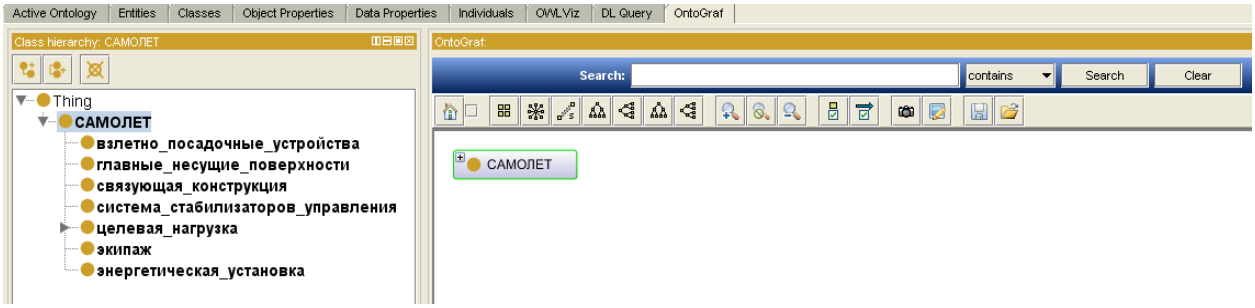

Figure 3.25 – Creating Semantic Representation

The «+» icon at the upper left corner of the “САМОЛЕТ” (Aircraft) button means that the «Самолет» (Aircraft) class contains subclasses or is a subclass. Double-click the “САМОЛЕТ” (Aircraft) button to disclose the structure. A fragment of Semantic Web is presented (Figure 3.26).

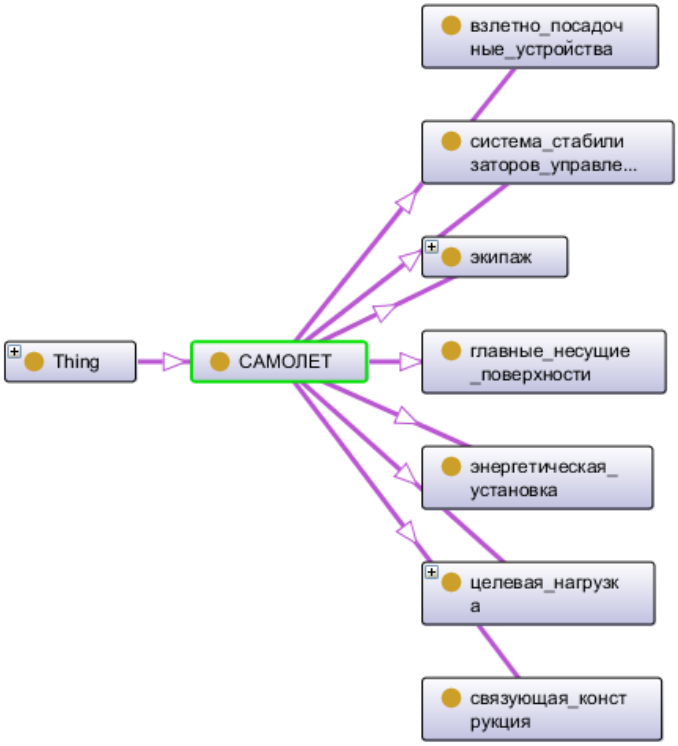

Figure 3.26 – A Semantic Web fragment

To disclose the web, you can also click the object of interest in the hierarchy. Note that the arrows between classes are of different colour (Figure 3.27).

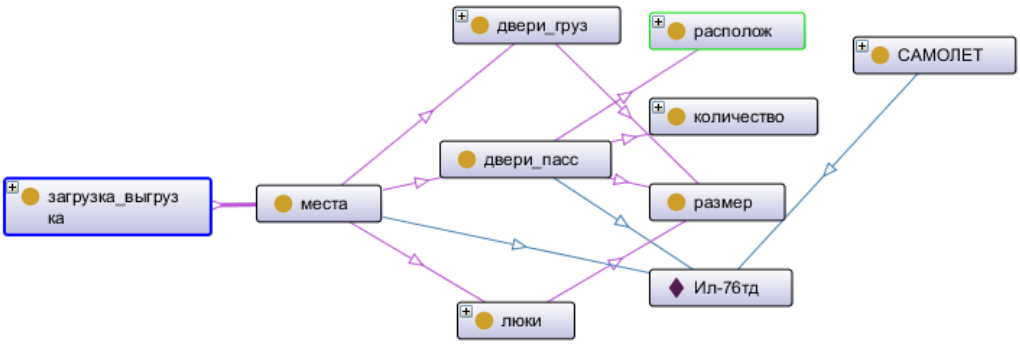

Figure 3.27 – A Semantic Web fragment that contains classes and instances

3.2.11. Saving a Semantic Web

To save the Semantic Web in JPEG format click the  button in the OntoGraf pane. In the appeared window select the folder, format and enter a name. Then click OK. The Semantic Web is shown on figure 3.28.

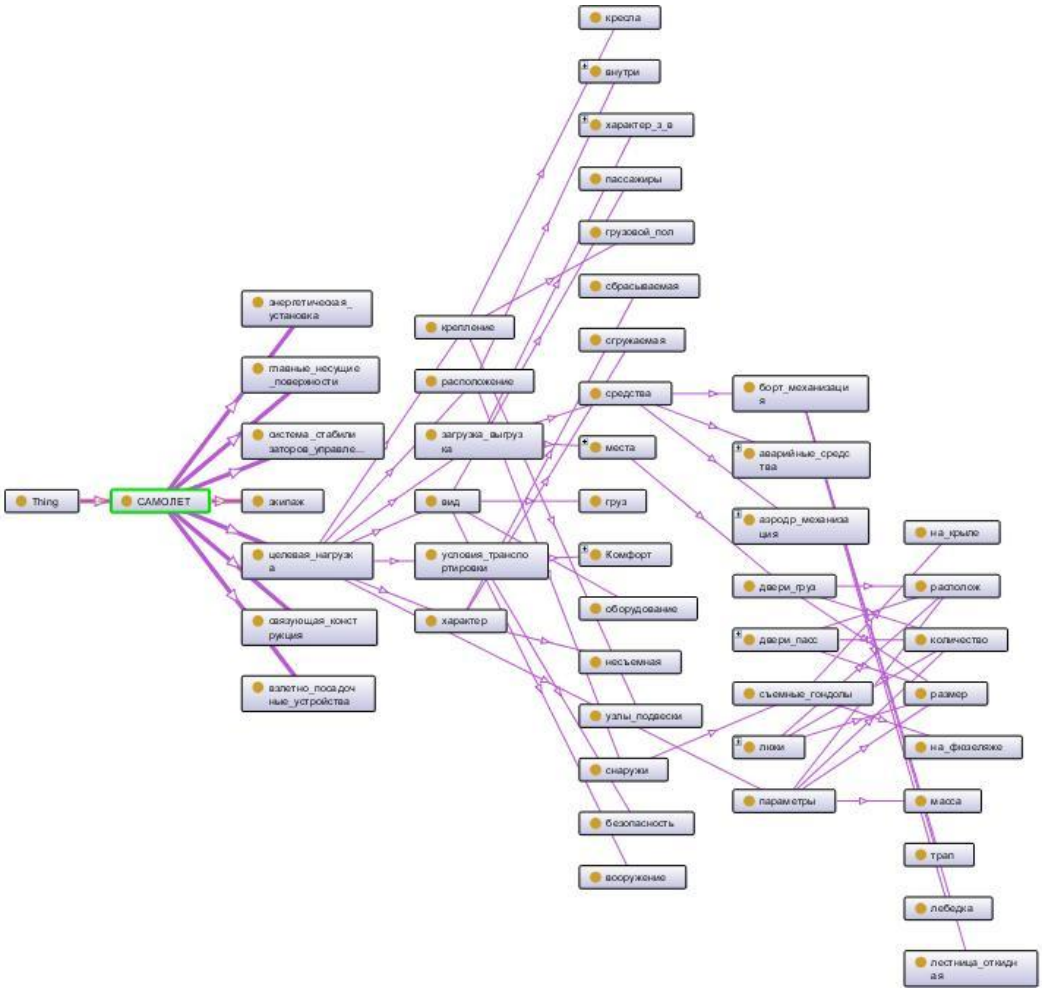

Figure 3.28 – Semantic Web

In **Protégé** there is an important option of OWL functional syntax rendering. The programme code can be used in other OWL ontology editors.

Navigate to the Active Ontology tab and select the OWL functional syntax rendering tab (Figure 3.29). Copy and save the programme code.

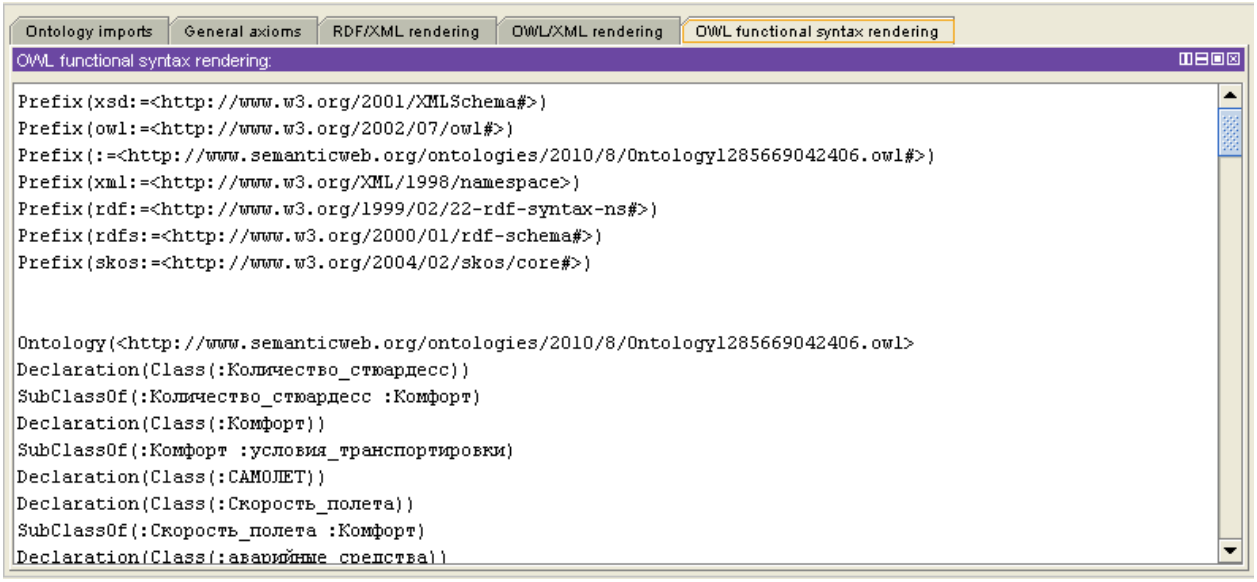

Figure 3.29 – OWL functional syntax rendering

3.3. SELF-STUDY TASKS

1. Expand the ontology in accordance with the task variant (Table 3.1).
– Create at least three levels in the hierarchy starting with class by the task,
– Create at least five subclasses for each level.
2. Supplement the ontology with the values of instances (3-5 instances).
3. Represent the result in the Semantic Web.
4. Include the functional syntax rendering in the report.

Table 3.1 – Task variants

№№	Aircraft element
1	Landing gear
2	Main load-bearing surfaces
3	Stabilizer system
4	Connecting structure
5	Power plant
6	Armament
7	Equipment
8	Payload type
9	Transportation conditions
10	Hatches
11	Loading and unloading means
12	Payload fixation
13	Payload location
14	Location of land-bearing surfaces
15	Power plant type

References

1. Боргест Н.М. Онтология проектирования: теоретические основы: Учеб. пособие. Самара: СГАУ, 2010 – 88 с.
2. Боргест Н.М. Антология онтологии: подборка научных статей. Самара: СГАУ, 2010 – 88 с.
3. Боргест Н.М., Симонова Е.В. Основы построения мультиагентных систем, использующих онтологию: Учеб. пособие. Самара: СГАУ, 2009 – 80 с.
4. Боргест Н.М. Автоматизация предварительного проектирования самолета: Учеб. пособие. Самара: Самар. авиац. ин-т, 1992, 92с.
5. Проектирование самолетов: учебник для вузов/С.М. Егер, В.Ф. Мишин, Н.К. Лисейцев и др. Под ред. С.М. Егера. - М.: Логос, 2005. - 648 с.