

documentation:



Competence Center for Case-based reasoning,  
DFKI, Kaiserslautern

Centre for model-based software engineering  
and explanation-aware computing,  
University of West London, UK

# SDK for building and integrating CBR systems

## Table of Contents

- ✱ What is CBR
- ✱ Knowledge formalisation in CBR
- ✱ The CBR cycle (the four 'R')
- ✱ CBR areas of application
- ✱ What's myCBR
- ✱ Architecture diagram
- ✱ myCBR sources and documentation
- ✱ Prerequisites
- ✱ Prerequisites for development
- ✱ Information's for developers
- ✱ How to install myCBR (getting started)
- ✱ How to install within Eclipse (for Developers)
- ✱ What and How to start
- ✱ myCBR Application design
- ✱ myCBR SDK integration
- ✱ myCBR the workbench GUI explained
- ✱ myCBR getting started: modelling your domain knowledge
- ✱ myCBR getting started: refining your knowledge model

# What is CBR





# What is CBR

## *The knowledge formalisation for CBR: Knowledge Containers*

### **Similarity Measures**

The retrieval of similar cases is based upon the use of similarity functions (or measures) to compute the distance or similarity of two cases.

### **Case base**

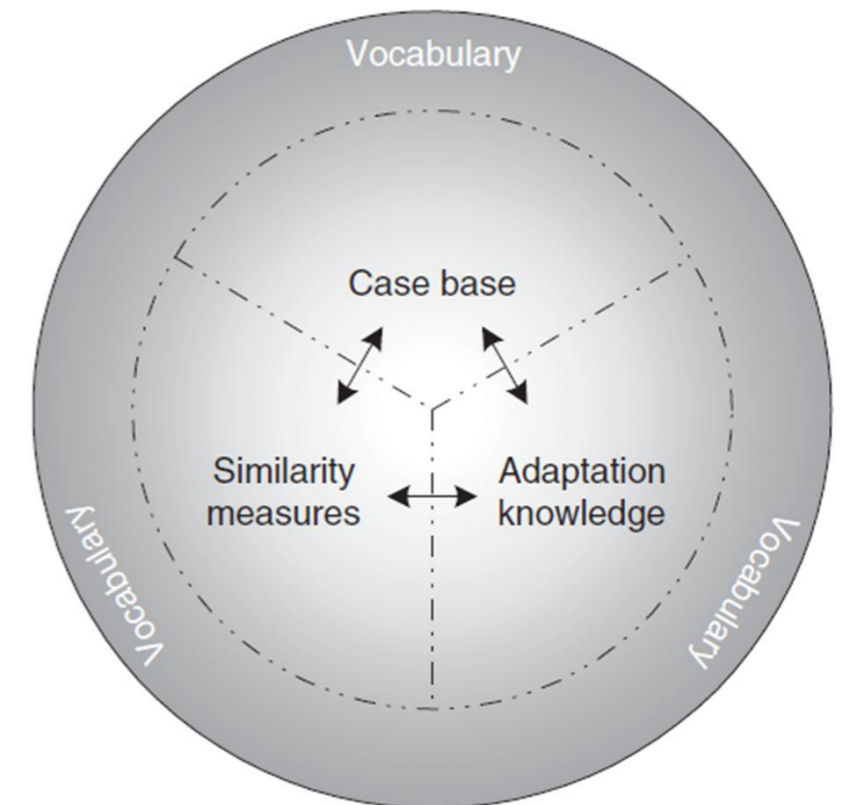
The systems experience is stored as cases within the case base which can be seen as a special form of a data base.

### **Vocabulary**

The cases themselves, the similarity measures and the adaptation knowledge are composed upon a vocabulary that contains the objects of interests (terms, attributes, concepts).

### **Adaptation knowledge**

Adaptation knowledge is used whenever a retrieved case's solution has to be adapted to be suitable to solve the presented problem. An example for this kind of knowledge is given by adaptation rules like "If X is not available use Y instead."

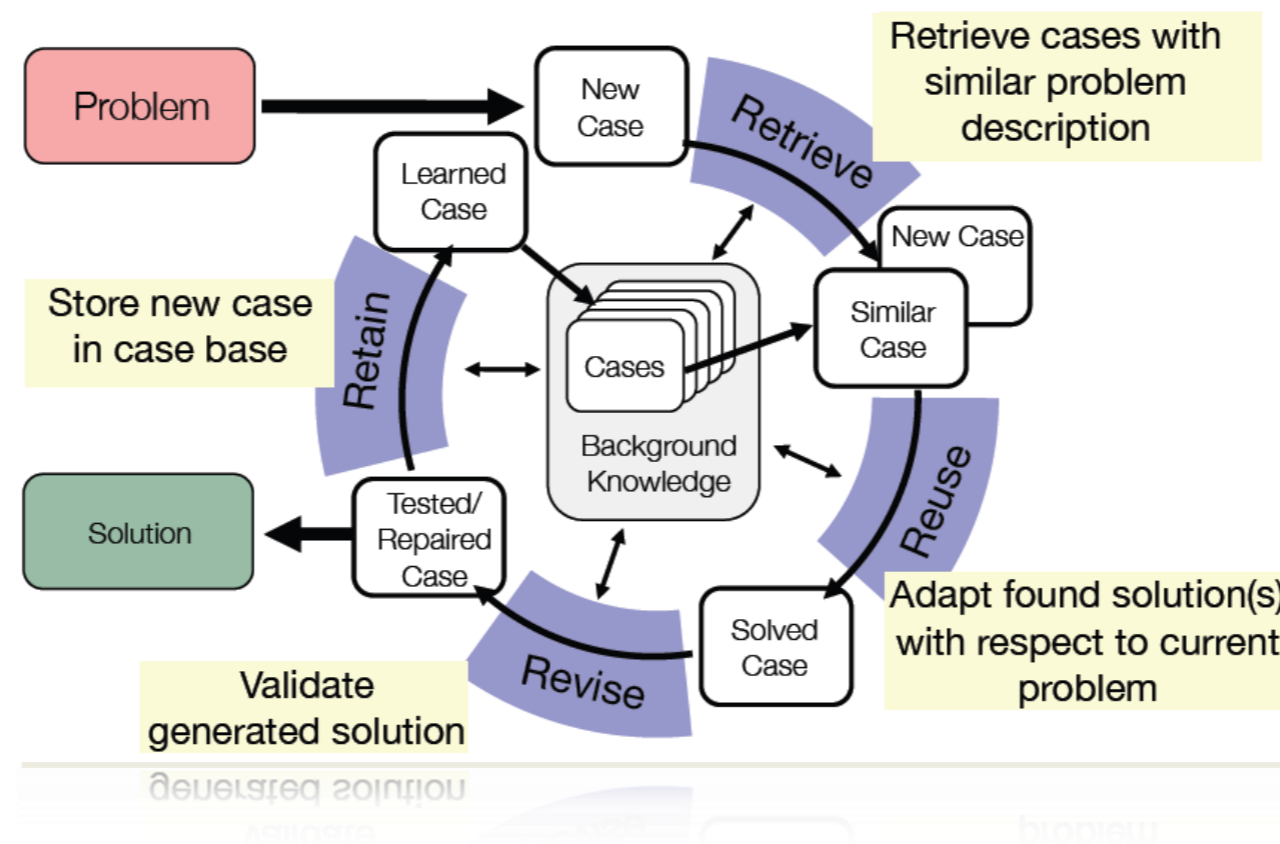


# Examples of CBR in human reasoning



- ✿ A medical doctor remembers the case history of another Patient
- ✿ A lawyer argues with similar original precedence
- ✿ An architect studies the construction of existing building to base his new designs on it
- ✿ A work scheduler remembers the construction steps of a similar work piece
- ✿ A mathematician tries to transfers a known proof to a new problem
- ✿ A service technician remembers a similar defect at another device
- ✿ A salesperson recommends similar products to similar customers

# The four steps of the CBR cycle: The 4 R's



**Retrieve:** the most similar case or cases: The case(s) with the most similar problem description (s)

**Reuse:** the information/experience stored in the solution descriptions of the retrieved case(s) to solve the presented problem

**Revise:** the retrieved solution if it is necessary to solve the presented problem in a satisfying way

**Retain:** the tested adapted new solution/experience as a new case, consisting of the presented problem description and the adapted solution description as a new experience in the case base

# Applications of CBR



CBR is capable of **automating** the tasks of **planning, diagnosis, design** and **recommending**.

It is used in a wide variety of successful business solutions. At the current time CBR is one of the most used AI methodologies within commercial applications.

Application possibilities/analogies for CBR:

- ✿ A medical doctor remembers the case history of another Patient
- ✿ A lawyer argues with similar original precedence
- ✿ An architect studies the construction of existing building
- ✿ A work scheduler remembers the construction steps of a similar work piece
- ✿ A mathematician tries to transfers a known proof to a new problem
- ✿ A service technician remembers a similar defect at another device

# What is myCBR



# SDK for building and integrating CBR systems

***myCBR is an open-source case-based reasoning tool hosted at [DFKI](http://mycbr-project.net)***

myCBR enables you to build CBR systems and their knowledge and to use them in your applications

Its aims are:

- ✱ to be easy to use
- ✱ to enable fast prototyping
- ✱ to be extendable and adaptable
- ✱ to integrate state-of-the-art CBR functionality

myCBR supports the teaching and research of the CBR approach by offering an easy way to prototype CBR engines

***You can download the latest version here: <http://mycbr-project.net/download.html>***

myCBR is developed as open source software currently by these Institutions:

- Competence Center for Case-based reasoning at the German Research Center for Artificial Intelligence
- Centre for model-based software engineering and explanation-aware computing at the University of West London

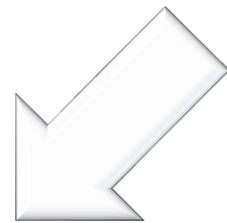


# SDK for building and integrating CBR systems

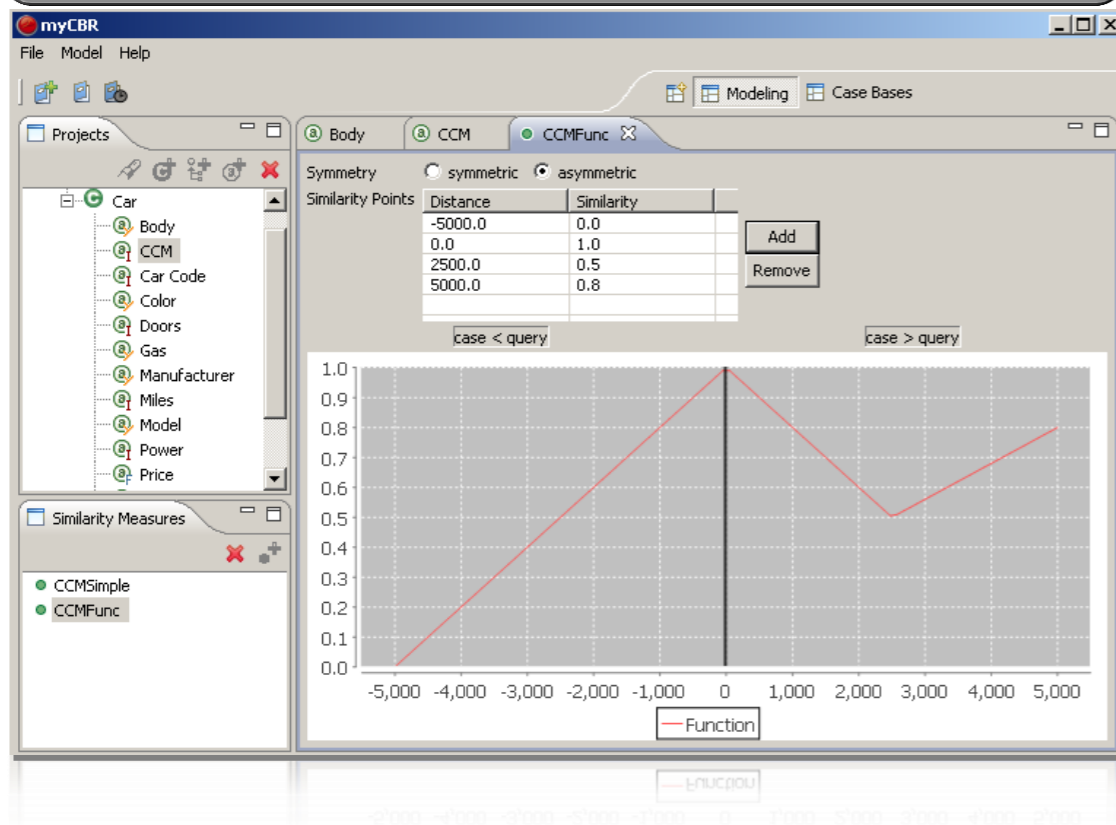
## Current Features of myCBR

- ✿ Powerful GUIs for modelling knowledge-intensive similarity measures
- ✿ Similarity-based retrieval functionality
- ✿ Export of domain model (including similarity measures) in XML
- ✿ Extension to structured object-oriented case representations, including helpful taxonomy editors
- ✿ Powerful textual similarity modelling
- ✿ Scriptable similarity measures using Jython
- ✿ Rapid prototyping via CSV
- ✿ Improved scalability
- ✿ Simple data model (applications can easily be build on top)
- ✿ Fast retrieval results
- ✿ Rapid loading of large case bases

# myCBR: Workbench and SDK [API]



## Workbench



## SDK [API]

```
// add a casse base to the project
```

```
DefaultCaseBase newcasebase = project.createDefaultCB("myCaseBase");
```

```
// add a case to the case base
```

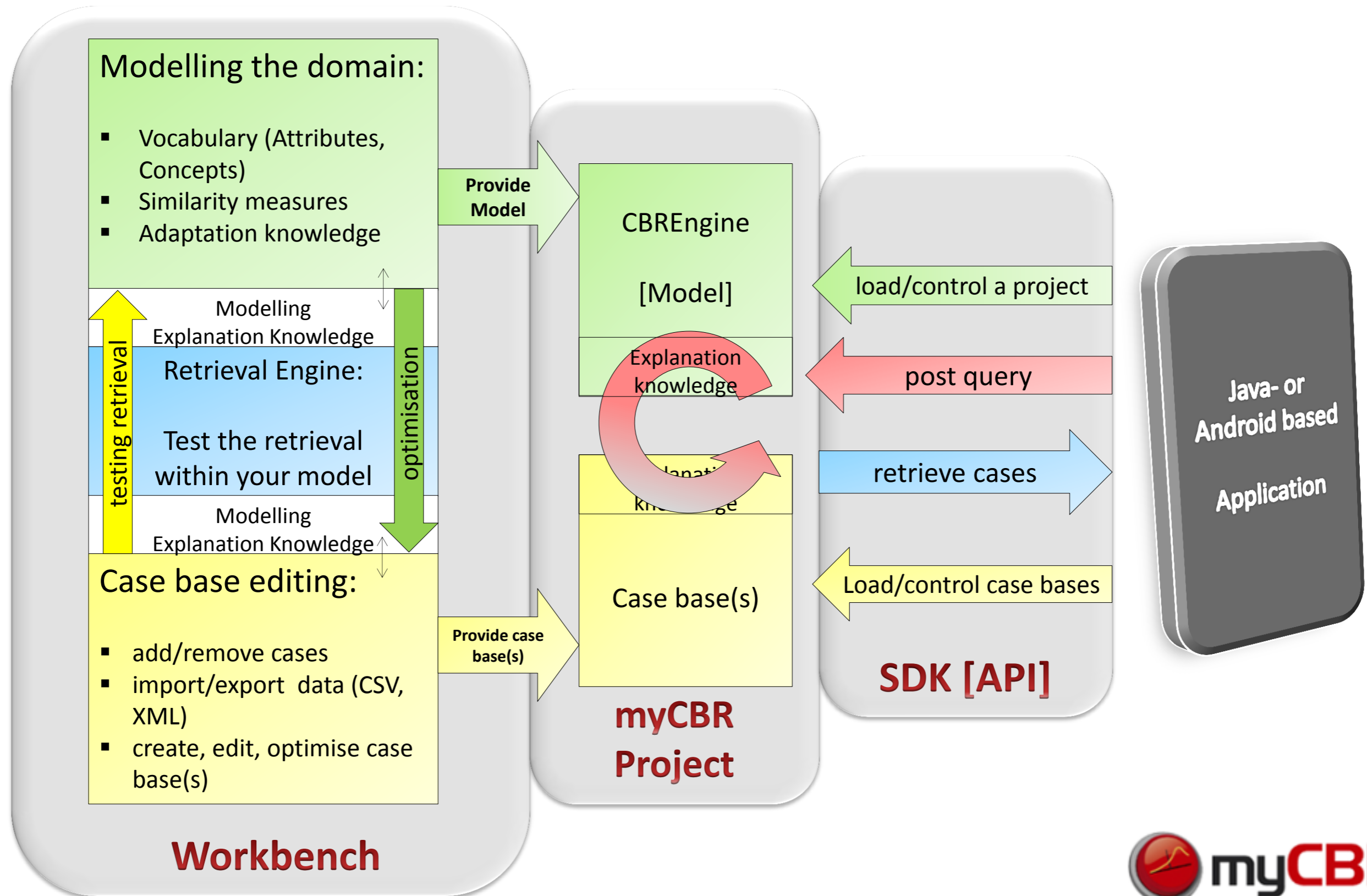
```
Instance instance = car.addInstance("car1");
instance.addAttribute(manufacturerDesc,manufacturerDesc.getAttribute("BMW"));
newcasebase.addCase(i, "car1");
```

```
// set up query, define a query instance and do a retrieval
```

```
Retrieval results = new Retrieval(car);
Instance query = results.getQuery();
query.addAttribute(manufacturerDesc.getName(),manufacturerDesc.getAttribute("Audi"));
```



# SDK for building and API for integrating CBR systems in your Application


























# Info's on myCBR



# Modularization of the project

The project is well modularized which should offer potential developers easy access to the code structure and class hierarchy of myCBR. The **java documentation** is available here: <http://mycbr-project.net/doc/index.html>

<a href="#">de.dfki.mycbr.core</a>	Contains all classes that represent core functionality of a CBR application such as the domain model, case bases, similarity functions and retrieval algorithms.
<a href="#">de.dfki.mycbr.core.action</a>	Defines classes for specifying actions that operate on Observable objects.
<a href="#">de.dfki.mycbr.core.casebase</a>	Contains classes for the basic definition of DefaultCaseBase objects.
<a href="#">de.dfki.mycbr.core.explanation</a>	Explanations provide additional information on all myCBR concepts.
<a href="#">de.dfki.mycbr.core.model</a>	Contains classes for the basic definition of the project's model.
<a href="#">de.dfki.mycbr.core.retrieval</a>	All retrieval algorithms extend the abstract class RetrievalEngine and can be used within Retrieval objects to obtain the retrieval results (possibly ordered pairs of case and corresponding similarity).
<a href="#">de.dfki.mycbr.core.similarity</a>	Contains standard classes to maintain similarity functions for attribute descriptions (local similarity functions) and concepts (amalgamation functions).
<a href="#">de.dfki.mycbr.core.similarity.config</a>	Contains various enumerations specifying configurations for the corresponding similarity function.
<a href="#">de.dfki.mycbr.io</a>	Contains classes that handle import and export of relevant CBR application data.
<a href="#">de.dfki.mycbr.util</a>	Contains utility classes that are useful but do not have a special meaning for case-based reasoning applications.

- ▷  de.dfki.mycbr.gui
- ▷  de.dfki.mycbr.gui.attribute.editor
- ▷  de.dfki.mycbr.gui.attribute.editor.type
- ▷  de.dfki.mycbr.gui.attribute.wizard
- ▷  de.dfki.mycbr.gui.casebase.editor
- ▷  de.dfki.mycbr.gui.casebase.list
- ▷  de.dfki.mycbr.gui.commands
- ▷  de.dfki.mycbr.gui.concept.editor
- ▷  de.dfki.mycbr.gui.csvimporter.wizard
- ▷  de.dfki.mycbr.gui.filter
- ▷  de.dfki.mycbr.gui.function.editor
- ▷  de.dfki.mycbr.gui.function.editor.function2d
- ▷  de.dfki.mycbr.gui.function.list
- ▷  de.dfki.mycbr.gui.function.wizard
- ▷  de.dfki.mycbr.gui.ie
- ▷  de.dfki.mycbr.gui.ie.lod.wizard
- ▷  de.dfki.mycbr.gui.ie.wizard
- ▷  de.dfki.mycbr.gui.instance.list
- ▷  de.dfki.mycbr.gui.misc
- ▷  de.dfki.mycbr.gui.perspectives
- ▷  de.dfki.mycbr.gui.plugin
- ▷  de.dfki.mycbr.gui.project.list
- ▷  de.dfki.mycbr.gui.retrieval.editor

## Web Sources for myCBR

Your ***web source*** for myCBR:

<https://git.opendfki.de/public>

# System Requirements

The **minimum system requirements** to run myCBR are:

- ✿ Essentially any PC that is able to run an Java Runtime Environment with reasonable performance is ok to use myCBR.
- ✿ CBR engines developed with myCBR are slim/efficient enough to be run on recent smartphones without limitations

**Software requirements** to run the myCBR SDK standalone:

- ✿ Java Runtime Environment (JRE) preferable in its latest version but the minimum version required is: 1.6

You can download the **latest version of JRE** here: <http://www.java.com/en/download/>

# Prerequisites for development

If you plan to **develop the myCBR SDK** further you need these additional prerequisites to be able to do so:

The **Java Development Kit (JDK)** with minimum version 1.6. however the most recent version recommended. You can find the JDK here:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

A **Java-Development Environment (JDE)**, we recommend Eclipse, which you can find here:

<http://www.eclipse.org/downloads/>

If you want to integrate **repository access** to the **mycbr.opendfki.de project** into you development environment we recommend to use a plugin to do so, for example to integrate the subversion repository into the Eclipse JDE we recommend the **Subclipse Plug-In** which you can find here: <http://subclipse.tigris.org/>

# Information's for developers

To sign up for the open source development community of myCBR, get an opendfki account here:

<http://www.opendfki.de/>

Repository access to the project (opendfki account required):

<https://mycbr.opendfki.de/repos/mycbr-gui>

During your account registration you can chose which project you like to contribute to. To contribute chose myCBR.

There is currently only the java doc available as a basis for reading into the projects source. This will be amended shortly by additional available material for developers.

As you are invited to add to, refactor and optimise the myCBR SDK we are still following a strict policy with the acceptance of new versions of the SDK. This policies require the passing of a series of j-unit tests that are available with the source of the SDK also. The last decision of new versions/features of the SDK still lies with the two Centres currently leading the development of the SDK at the DFKI and the UWL.

# Information's for developers

The DFKI and UWL maintain a ***Wiki for all information about the current development*** of the myCBR SDK. However being able to access this Wiki depends on getting a user account for which you can contact christian.sauer@uwl.ac.uk.

You can find the Wiki here: <http://mycbr.opendfki.de/>

To get signed up to the ***mailing list of myCBR developers*** and get updates about the latest developments you can contact: [cbr@dfki.uni-kl.de](mailto:cbr@dfki.uni-kl.de) .

If you want to get a ***broader view and the latest developments in CBR*** you may also consider signing up for the CBR Wiki, to be found here: [http://cbrwiki.fdi.ucm.es/mediawiki/index.php/Main\\_Page](http://cbrwiki.fdi.ucm.es/mediawiki/index.php/Main_Page)

# myCBR Installation



# How to install the myCBR SDK on a Windows 7 PC


1. Download the zip archive (binaries) from the myCBR download website:

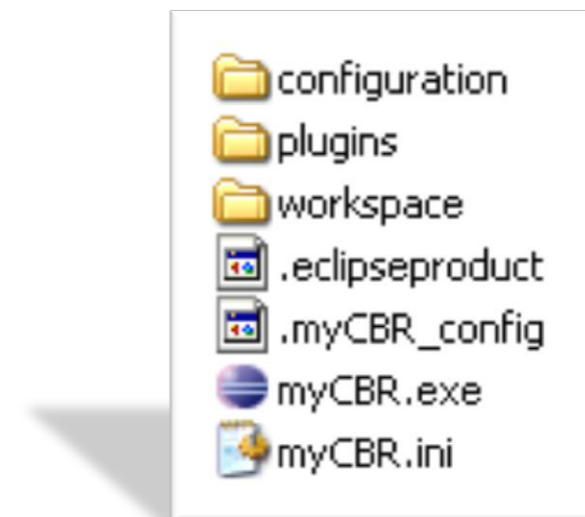
<http://mycbr-project.net/download.html>

*This archive contains all files for the 'product' version of the myCBR SDK*

2. Unzip the contents of the mycbr.zip in a single folder.

*The contents of your new myCBR folder should look like pictured here.*

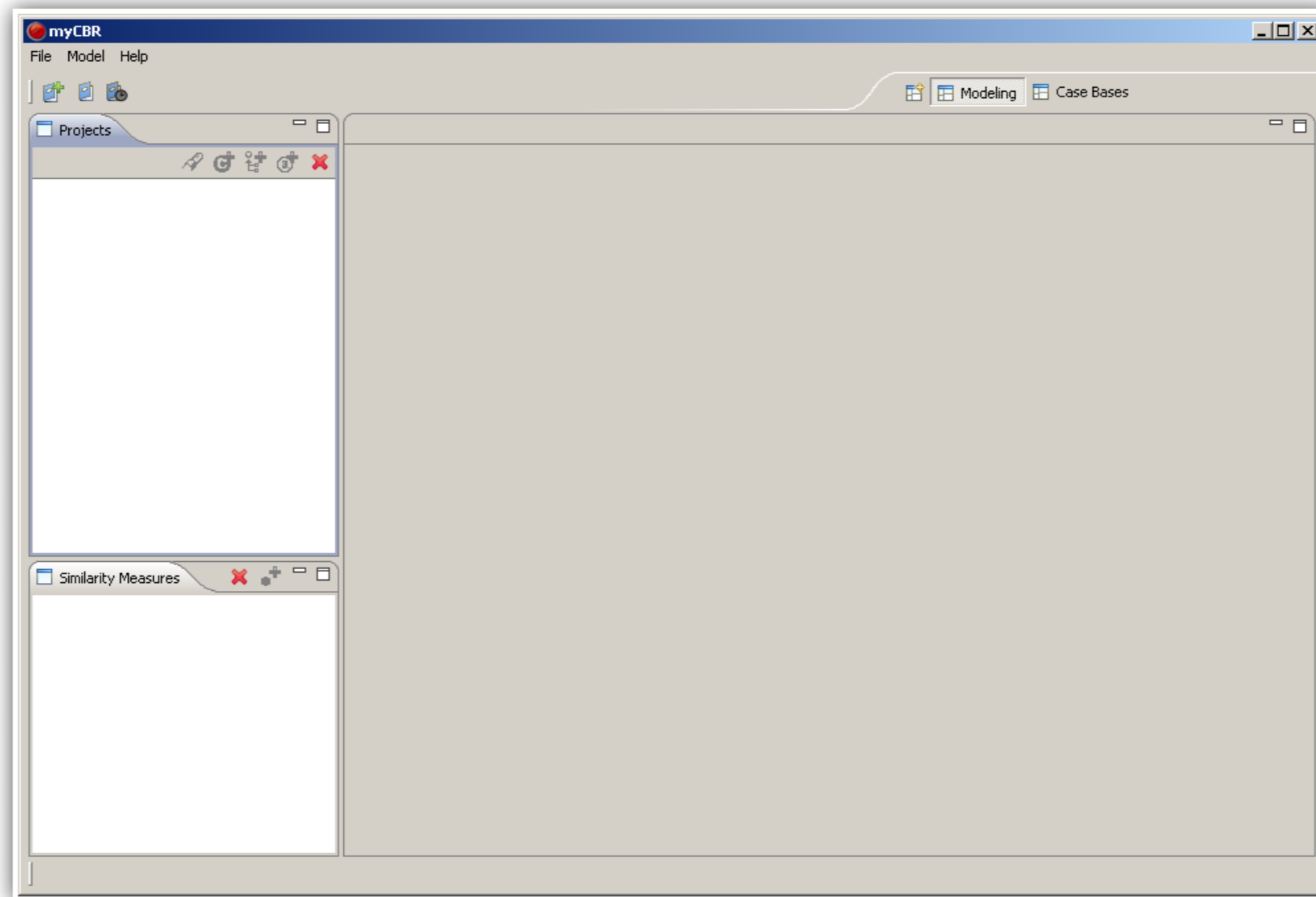
3. To Start myCBR simply run (double-click) the  myCBR.exe file



*The start-up may take a while...*

# SDK for building and integrating CBR systems

If everything went correct (be patient as it is a java application the start of myCBR might take up to 30 seconds) your first Impression of myCBR should look like this:



# myCBR installation for developers

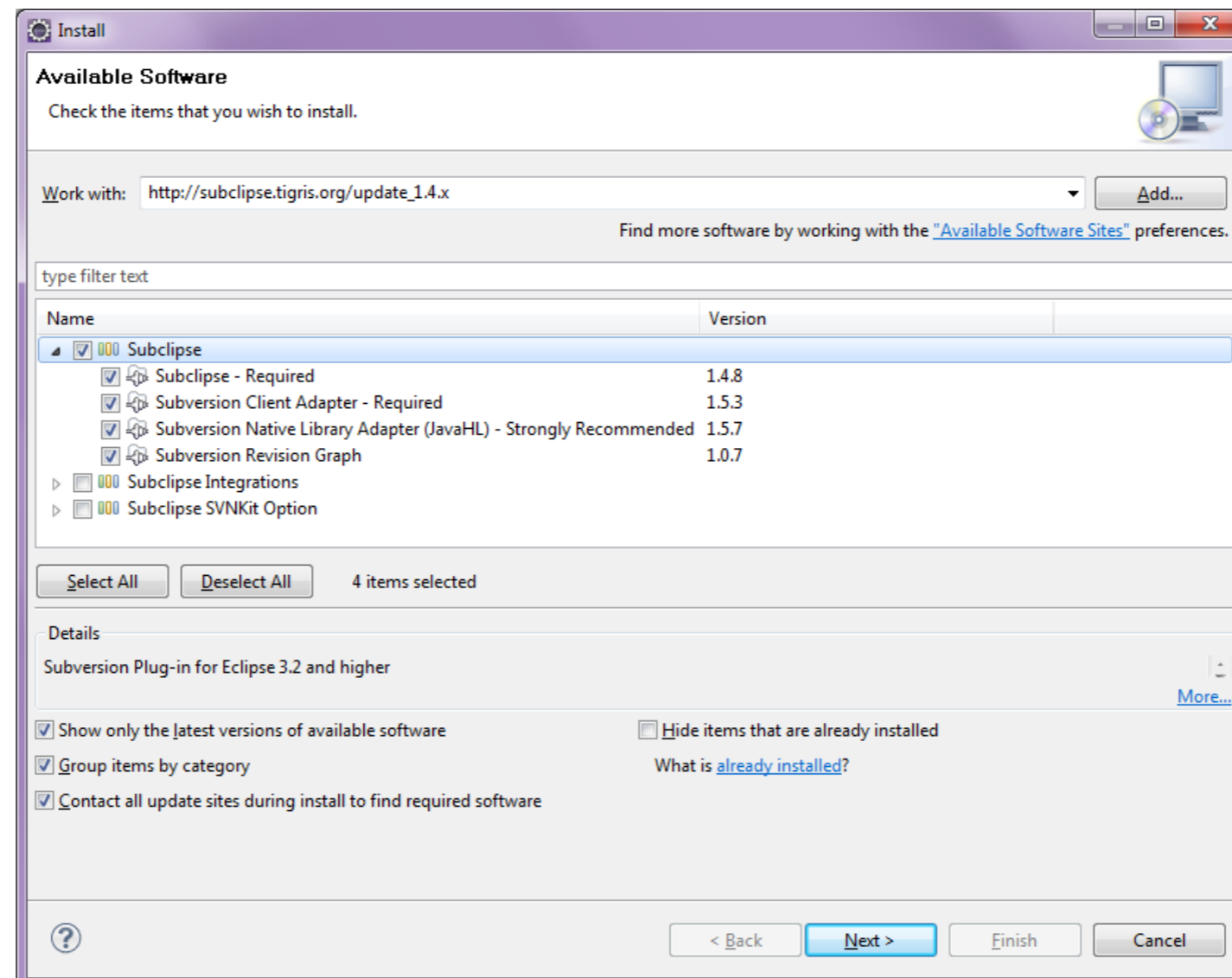


# Installation of the Development Environment



**Prerequisite 1:** Subversion integrated into **Eclipse: Subclipse**

Installation: In Eclipse open the Help menu → Install new software → Enter the path at “Work with:” [http://subclipse.tigris.org/update\\_1.4.x](http://subclipse.tigris.org/update_1.4.x) → Choose Subclipse and all of its components



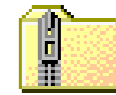
# Installation of the Development Environment



**For older versions of the JDE (Eclipse)** or if you want to work with older versions then version 3.0 of myCBR you might need the Eclipse: **Standard Widget Toolkit (SWT)**

This shouldn't be necessary with most recent Eclipse/myCBR versions but if you have to include SWT into your Eclipse JDE make sure you do so **before** you move on to configure your myCBR project.

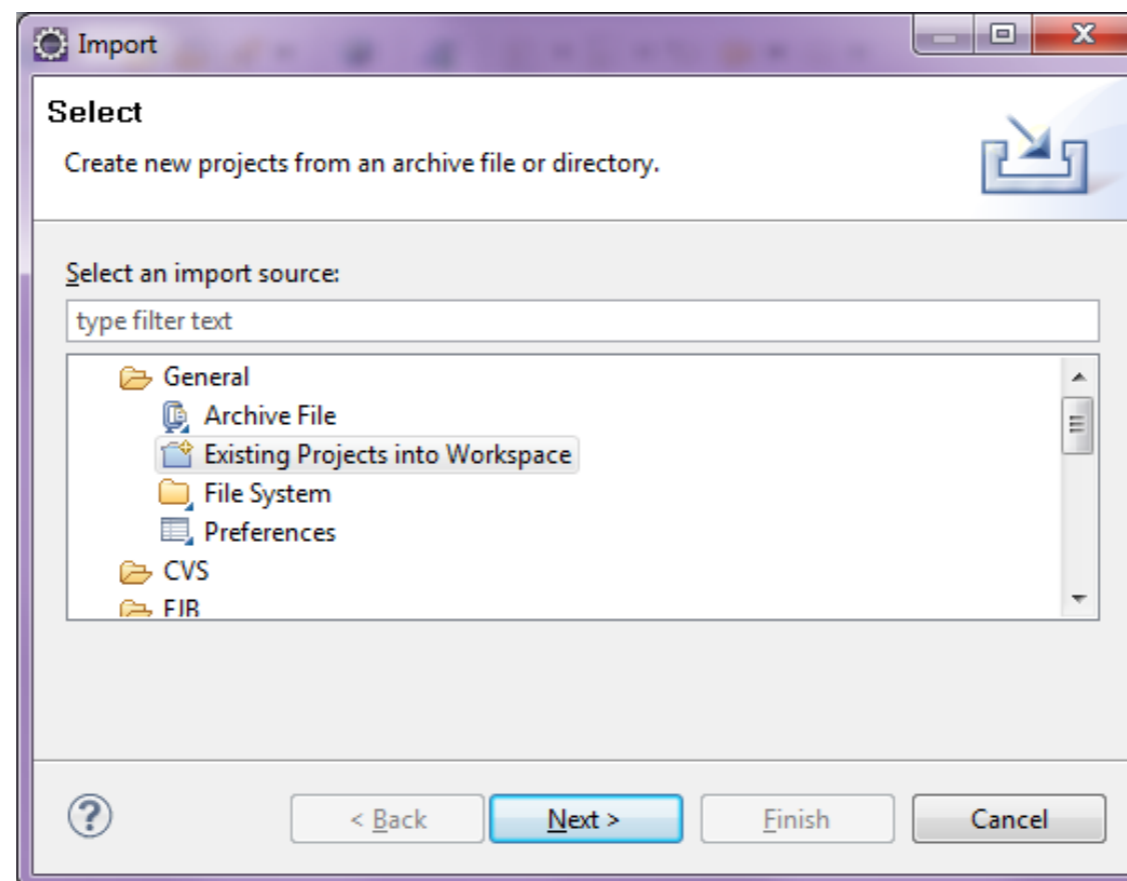
You can download the SWT here: <http://www.eclipse.org/swt/>



swt-3.7.2-win  
32-win32-x86.  
zip

Installation: Include SWT as a project (might be needed on older version of Eclipse)

Import → Existing Projects into Workspace

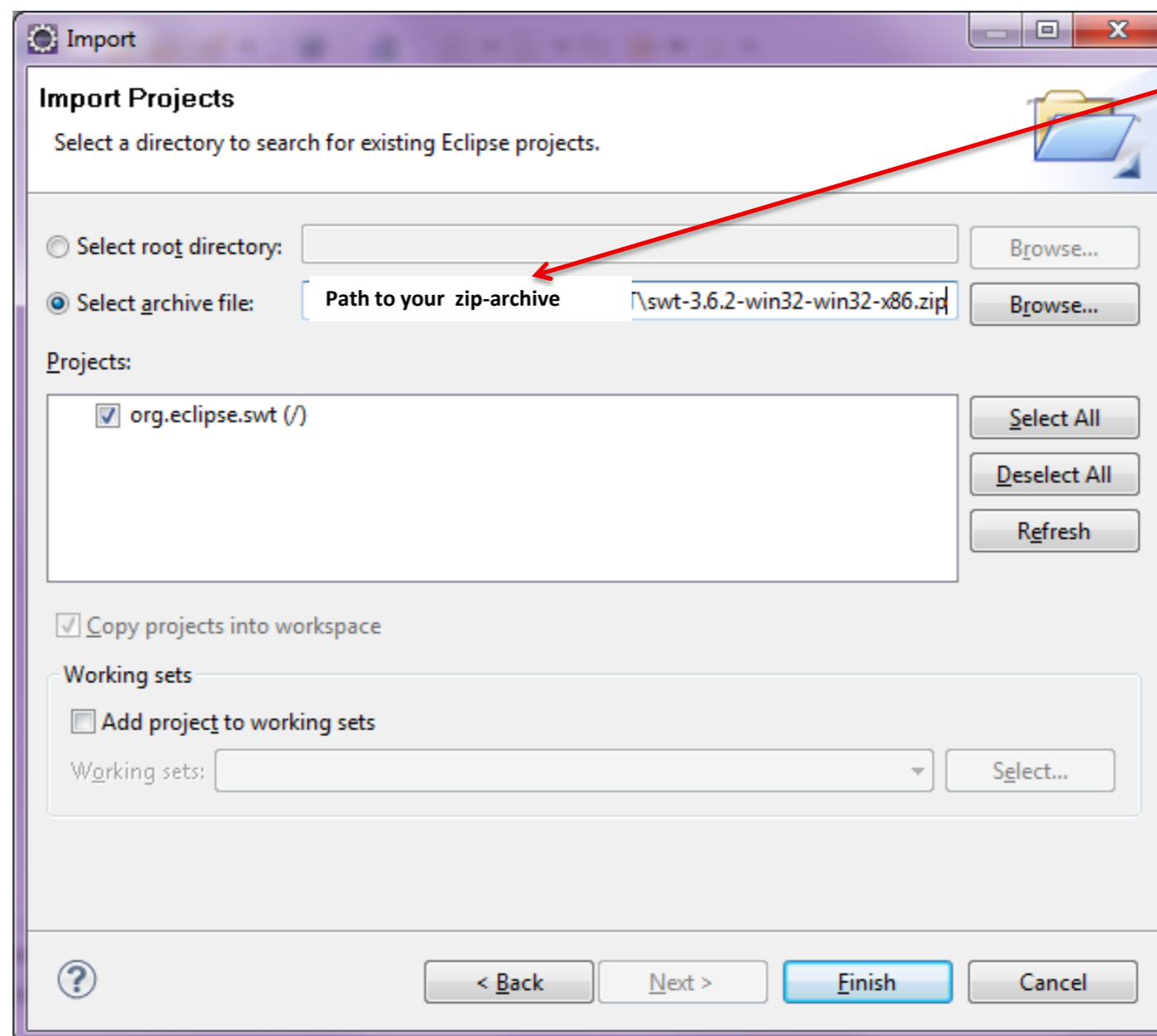


# Installation of the Development Environment



Eclipse: Standard Widget Toolkit installation (continued)

Chose the path to your downloaded SWT zip archive → Select the project (org.eclipse.swt) → Finish



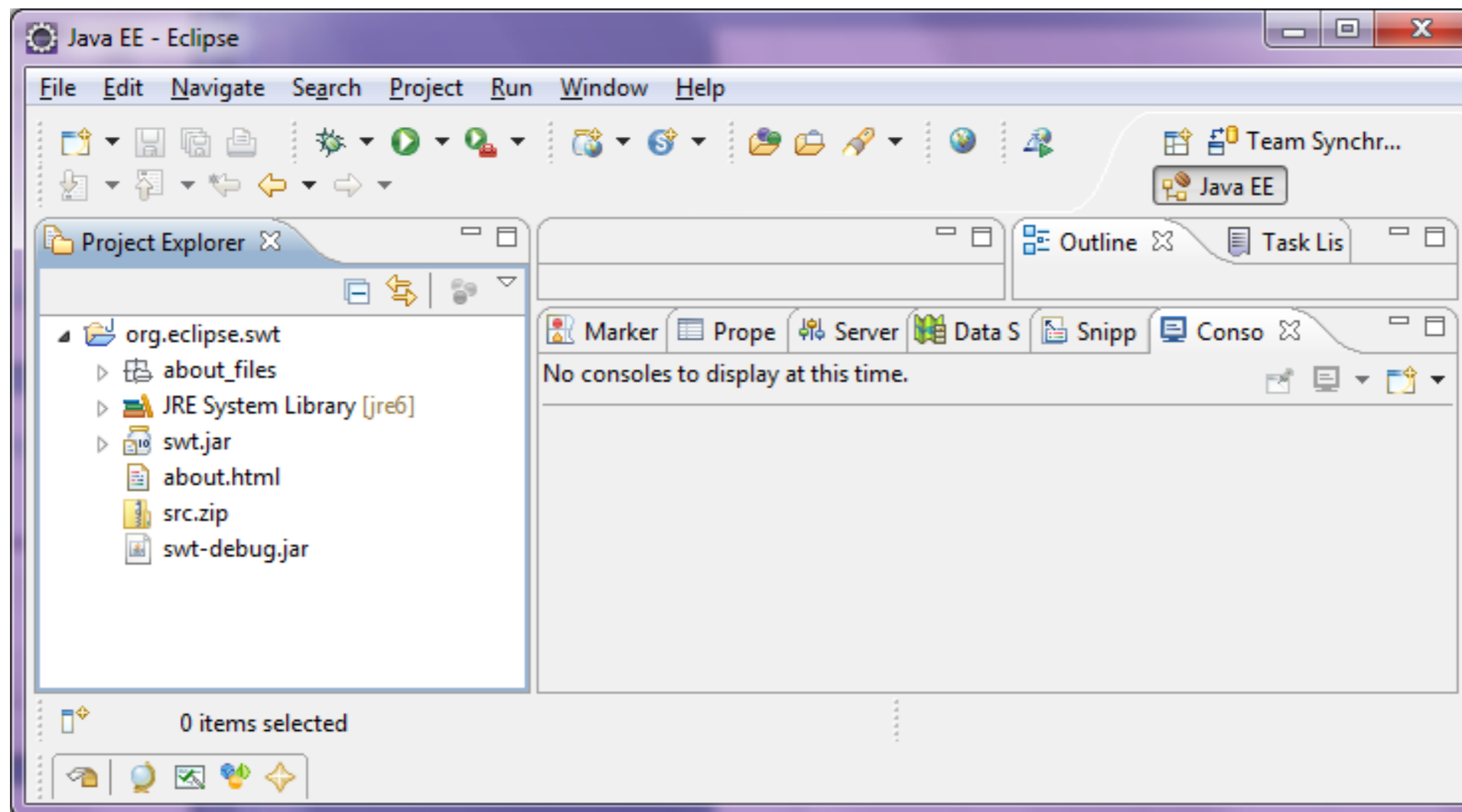
swt-3.7.2-win  
32-win32-x86.  
zip

# Installation of the Development Environment



Eclipse: Standard Widget Toolkit installation (continued)

If your SWT installation was successful, you should have a project like this:



# Installation of the Development Environment

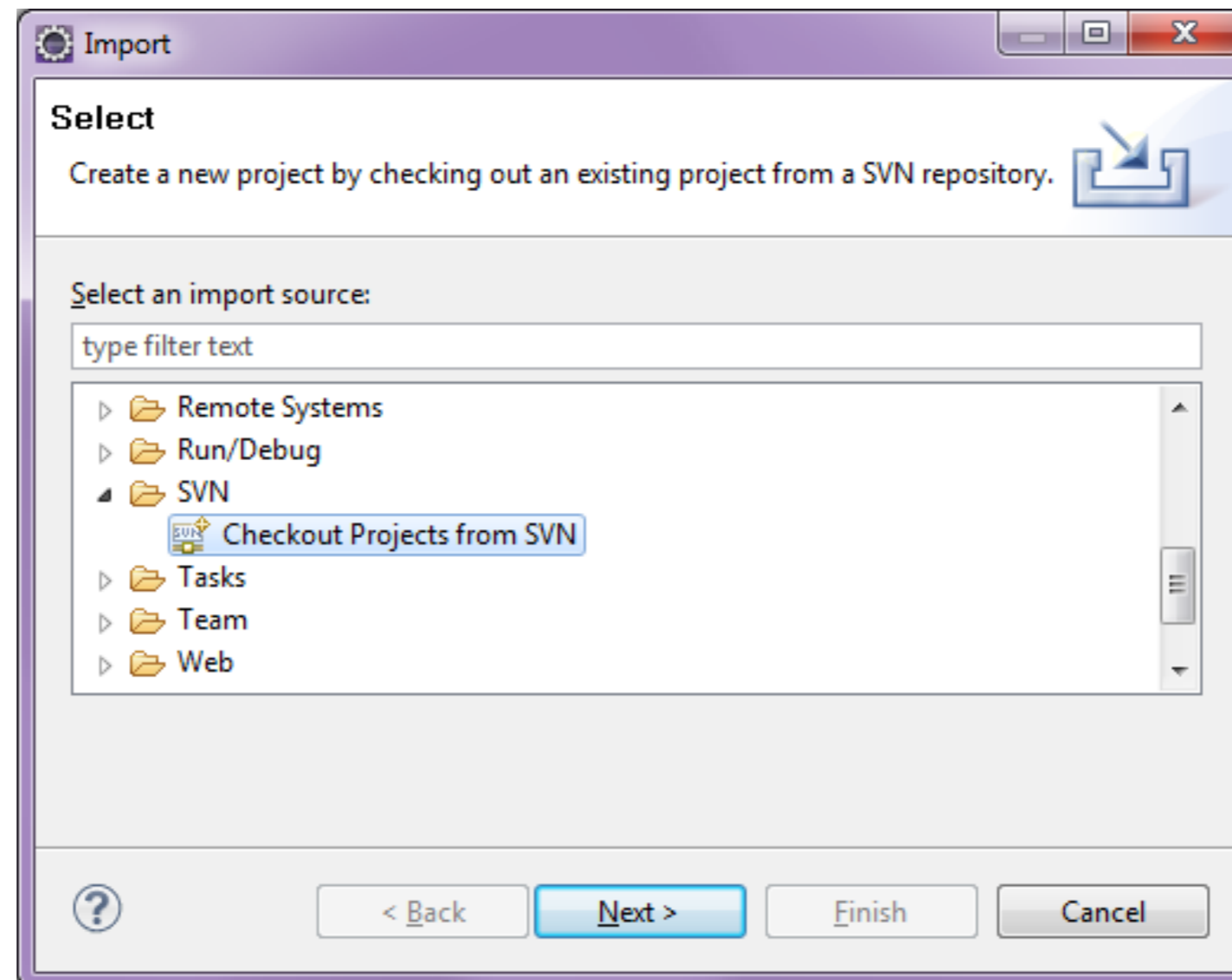


**Prerequisite 2: Import the myCBR project** from the repository at the DFKI

File → Import → SVN → Checkout Projects from SVN → Next →

Create a new repository location: URL: <https://mycbr.opendfki.de/repos/mycbr-gui/trunk>

→ Next

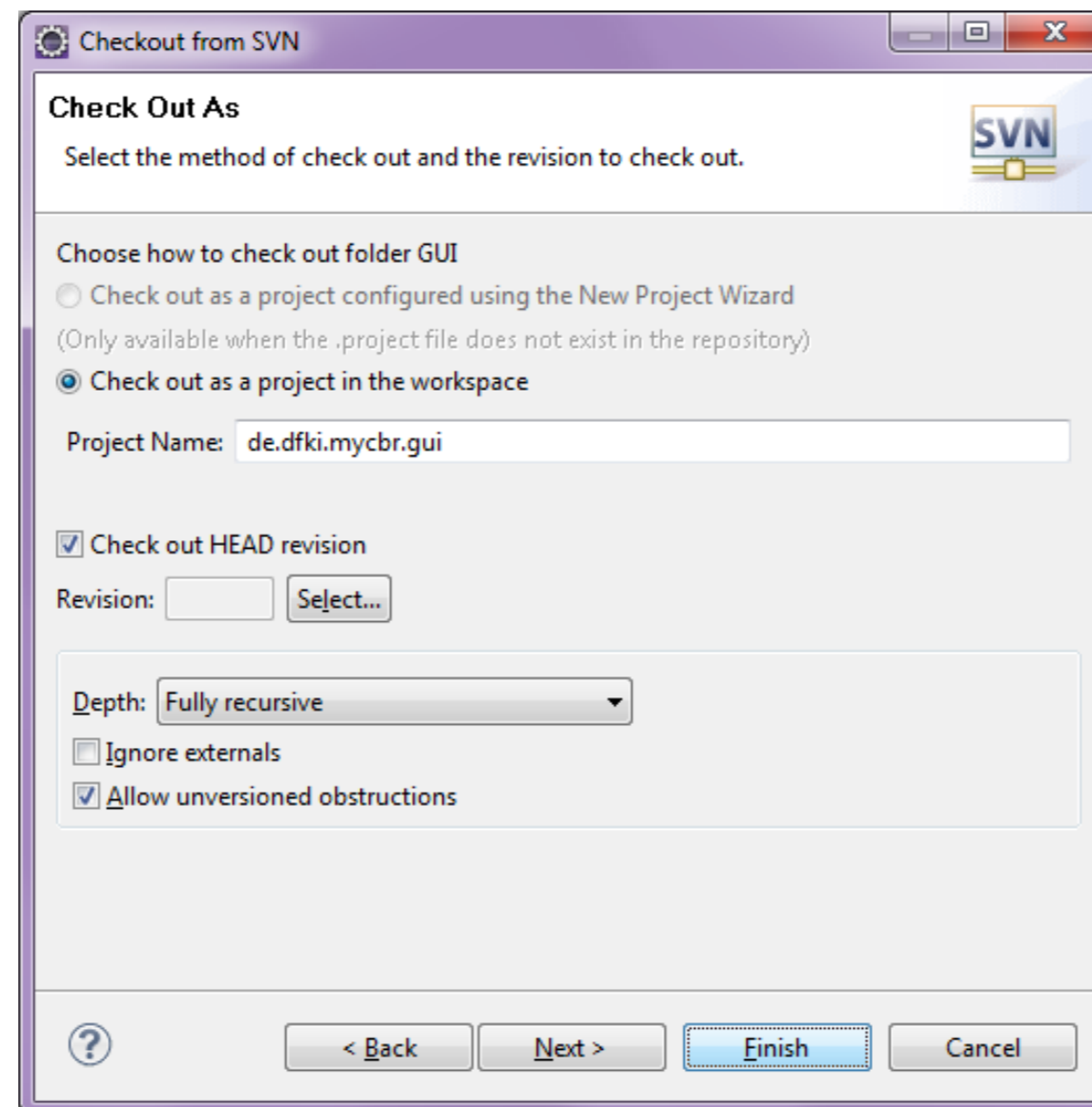


# Installation of the Development Environment



Prerequisite 2: **Import the myCBR project** from the repository at the DFKI (continued)

Select the project as shown below, make sure to tick the checkbox '*Check out as a project in the workspace*'. *Check out the HEAD revision* and select fully recursive and *allow for unversioned obstructions* → Finish

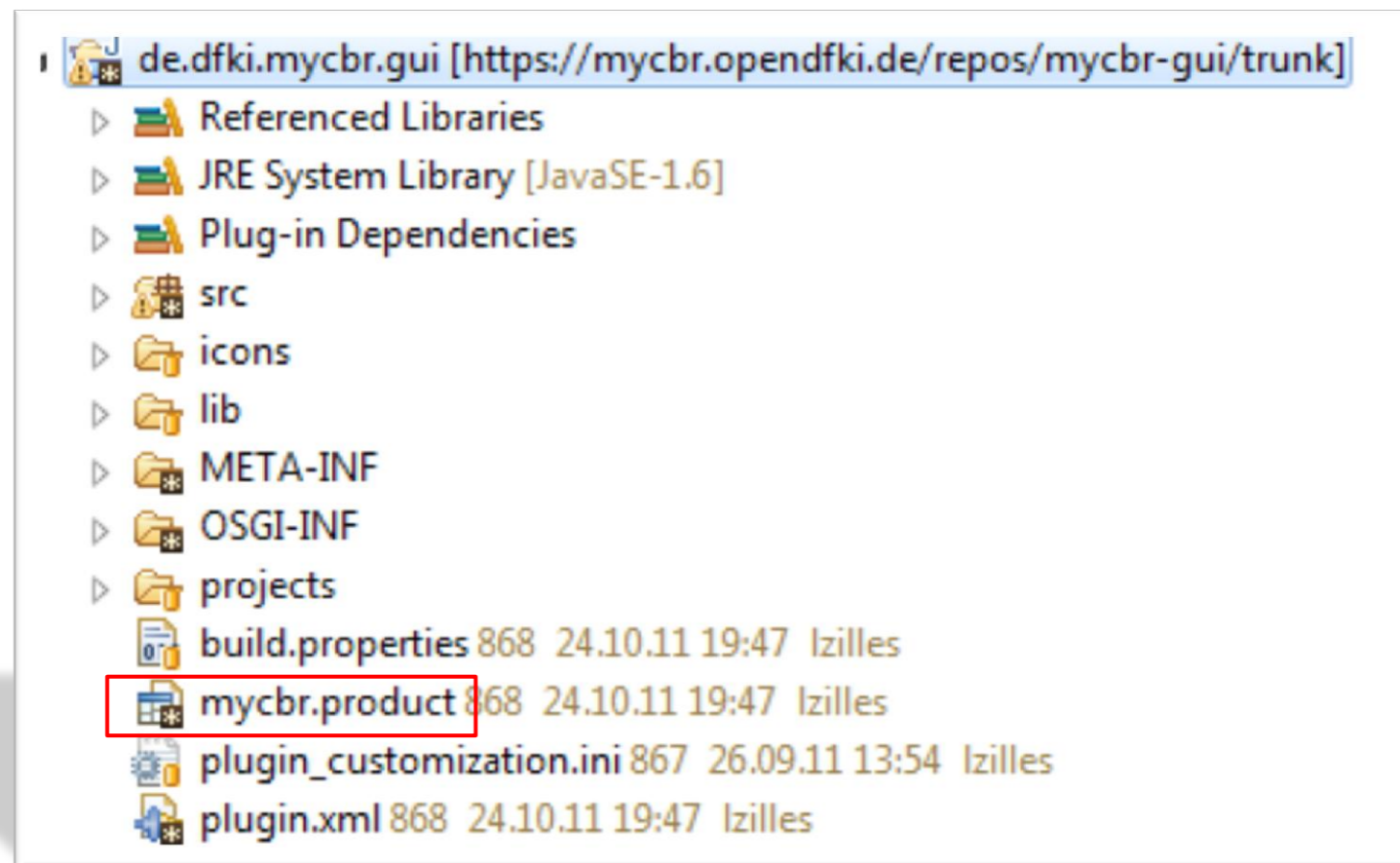


# Installation of the Development Environment



Prerequisite 2: **Import the myCBR project** from the repository at the DFKI (continued)

If all went okay you should have a project like this in your Eclipse IDE:

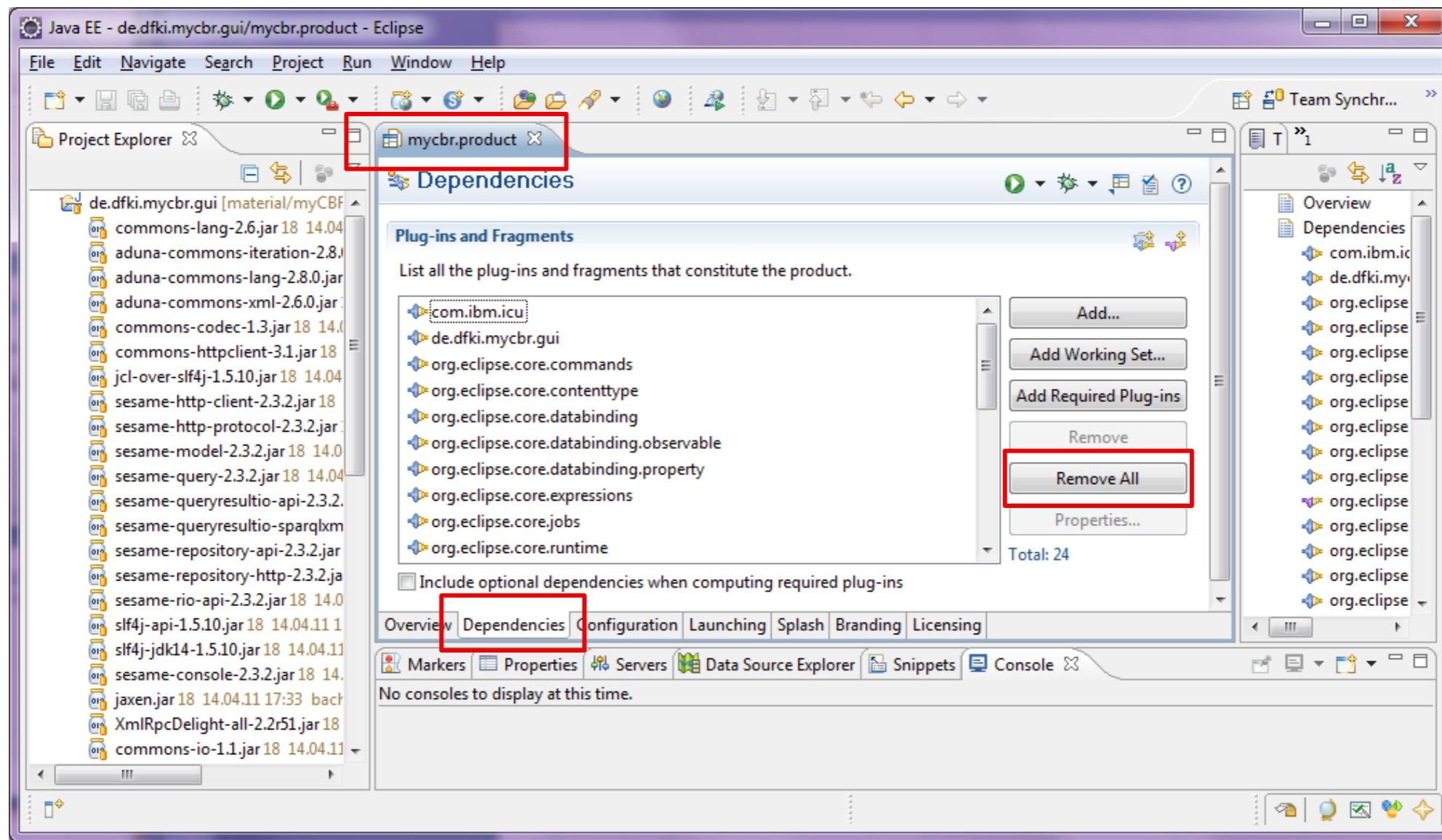


Have a look at *mycbr.product*, this is what you need to configure to launch myCBR from your JDE



# Launching myCBR from the JDE

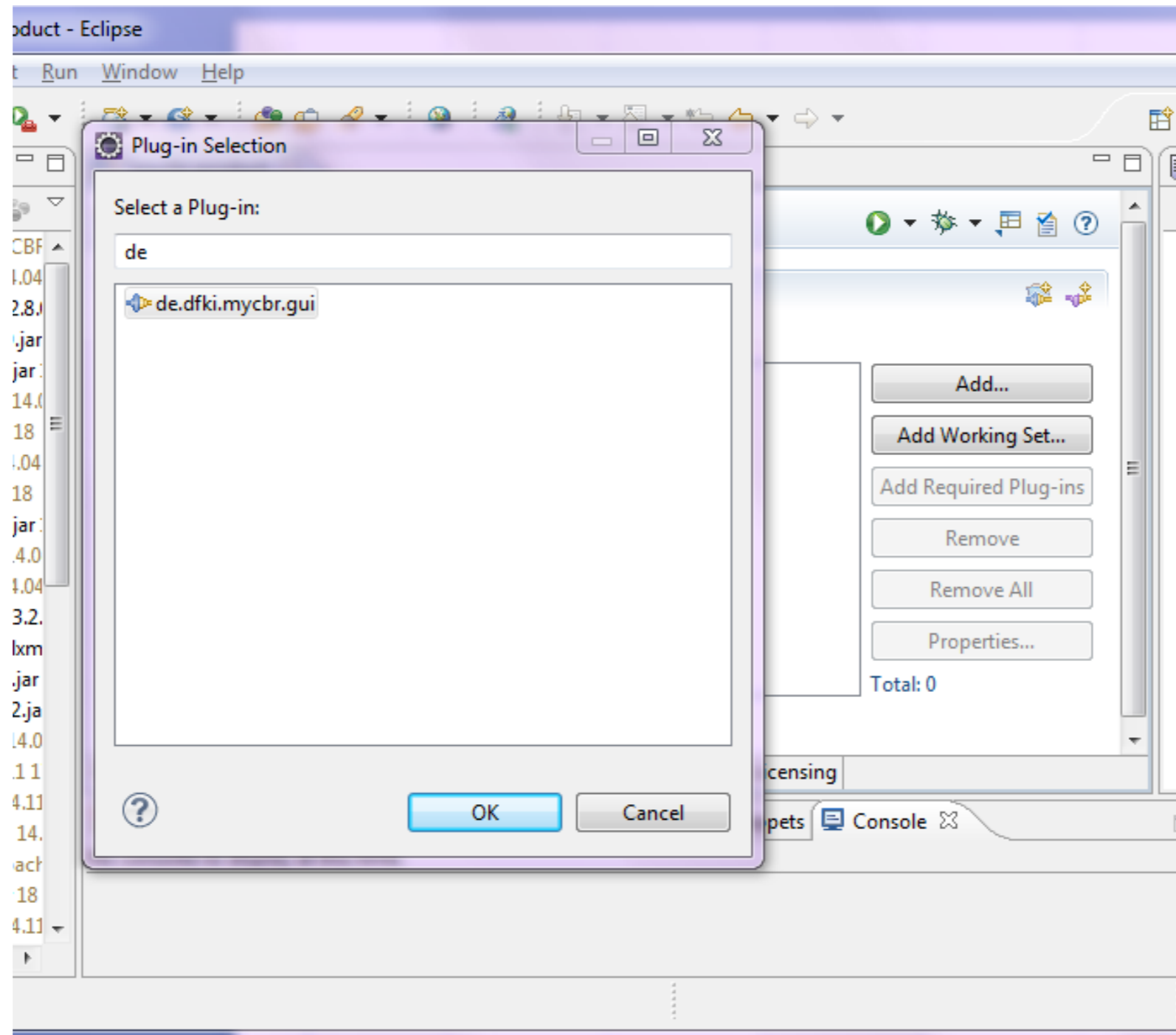
Double-click 'mycbr.product' in your project tree: Select the Dependencies Tab → Remove All



# Launching myCBR from the JDE



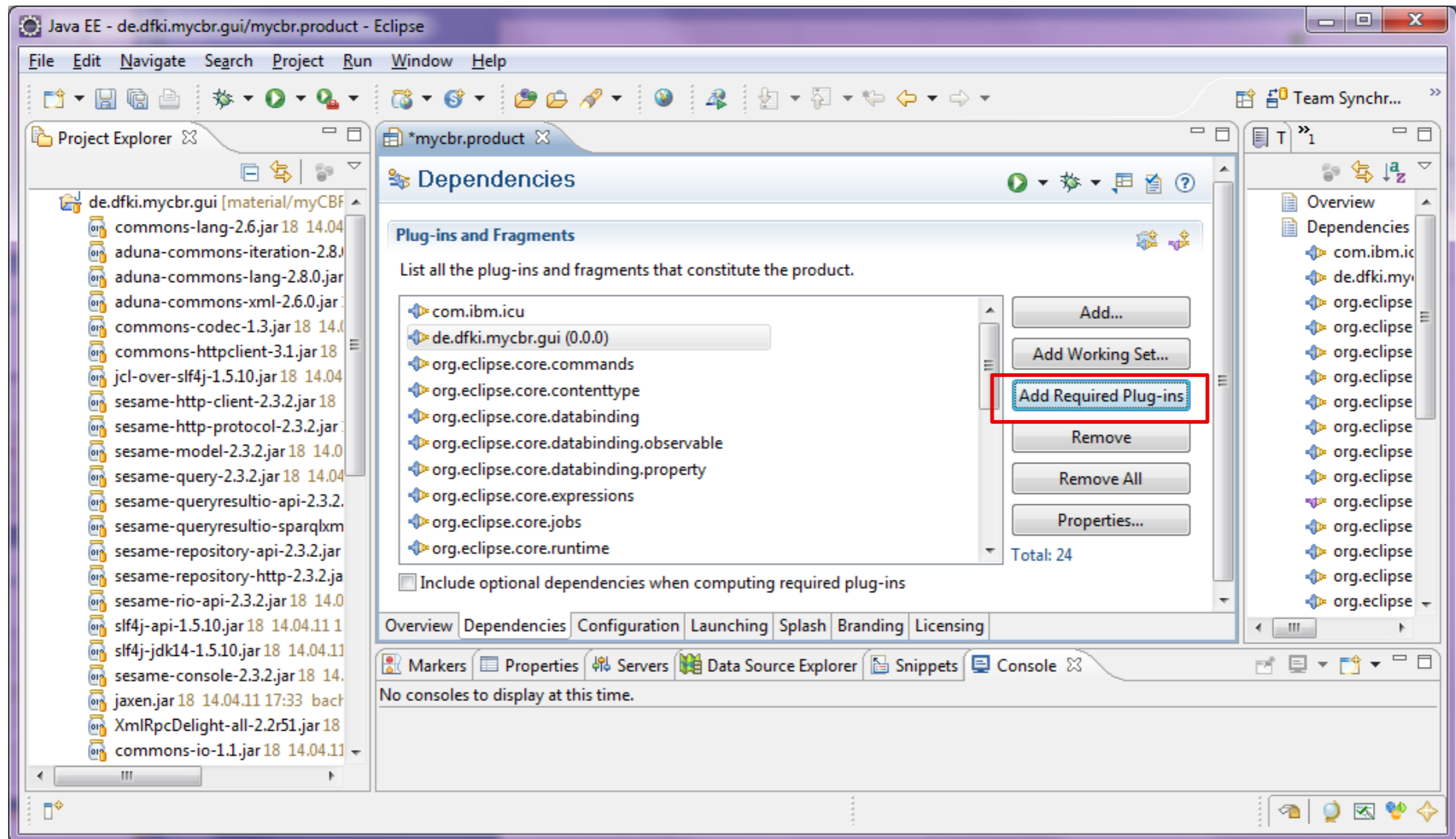
Click on 'Add...' → (and then select) *de.dfki.mycbr.gui* → OK





# Launching myCBR from the JDE

Click on '*Add required Plug-Ins*' to let Eclipse integrate all necessary Plug-Ins for myCBR to run

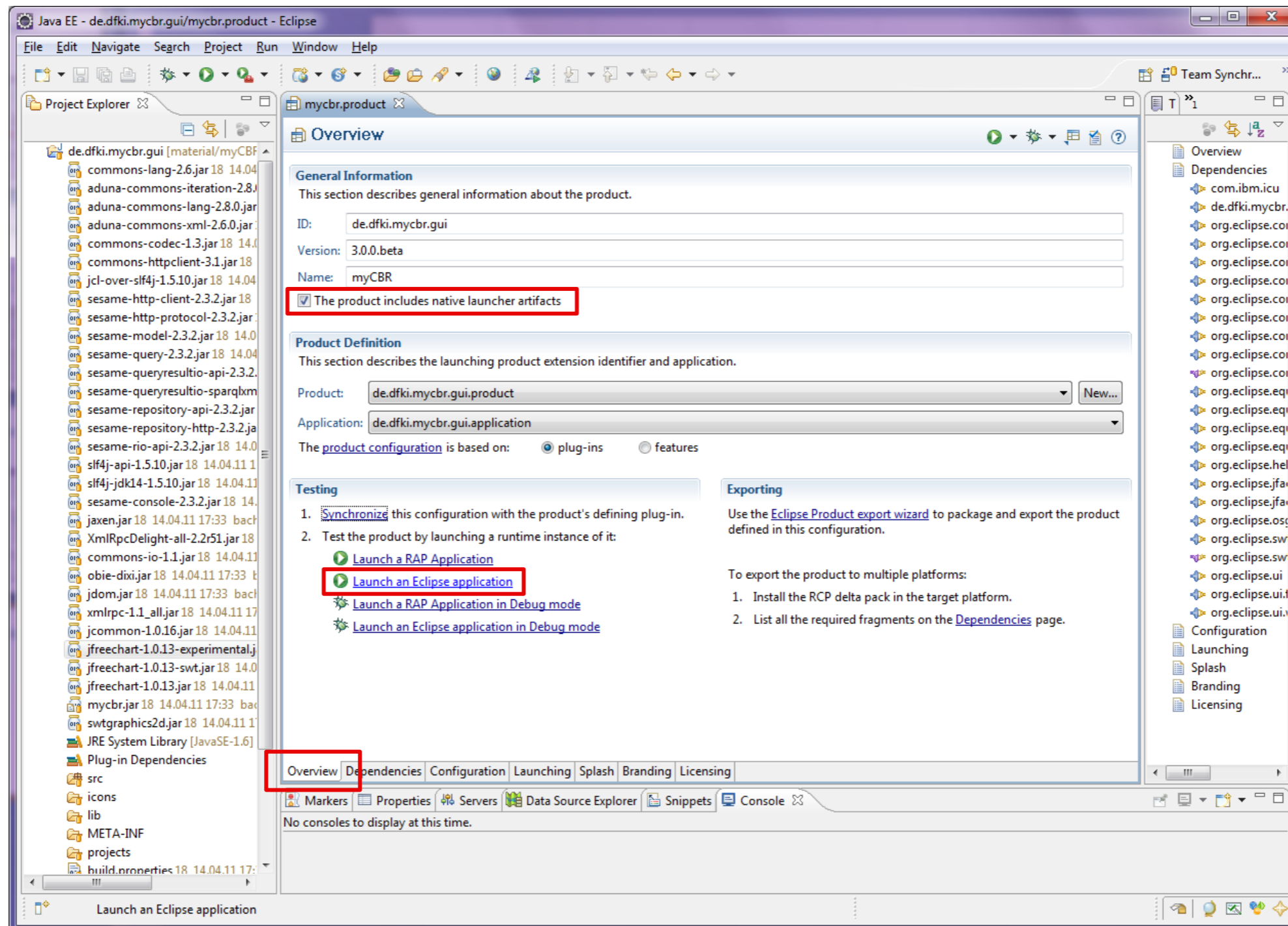




# Launching myCBR from the JDE

Back in the 'Overview' Tab make sure that the checkbox 'The product includes native launcher artifacts' is ticked.

You can now select 'Launch an Eclipse application' to start the myCBR SDK



# myCBR Application design and API use



# myCBR Application design



Integrating a myCBR engine in your project can be achieved in different ways:



Direct myCBR integration into an application:

You can integrate the myCBR library directly into your java program and load myCBR engines and case-bases into your program to execute queries on them



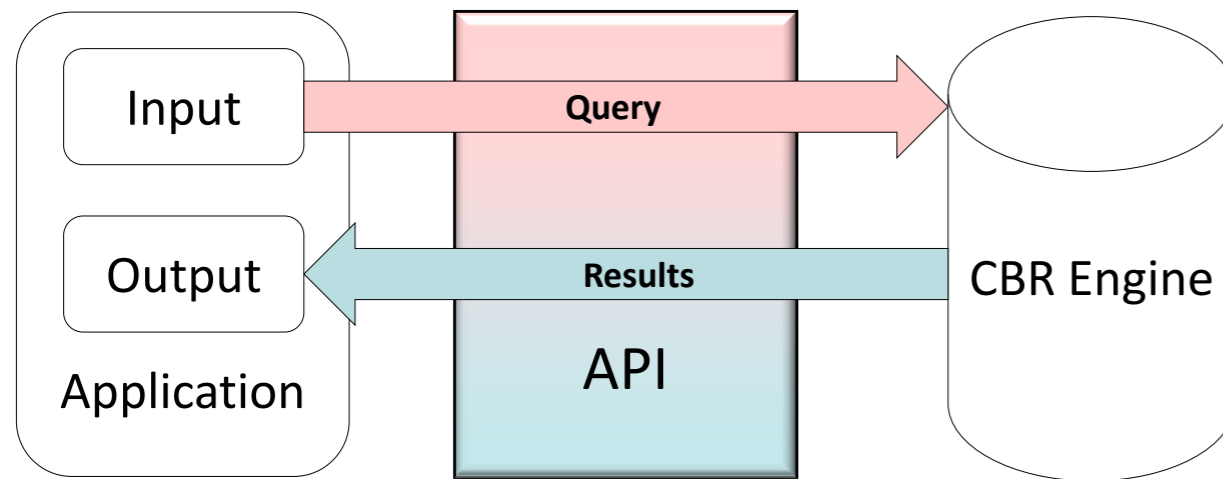
Central myCBR engine server and several clients: (alpha stadium development)

You can run a central server which integrates the myCBR library into a server-based java program and then process queries, deliver query-results to thin clients

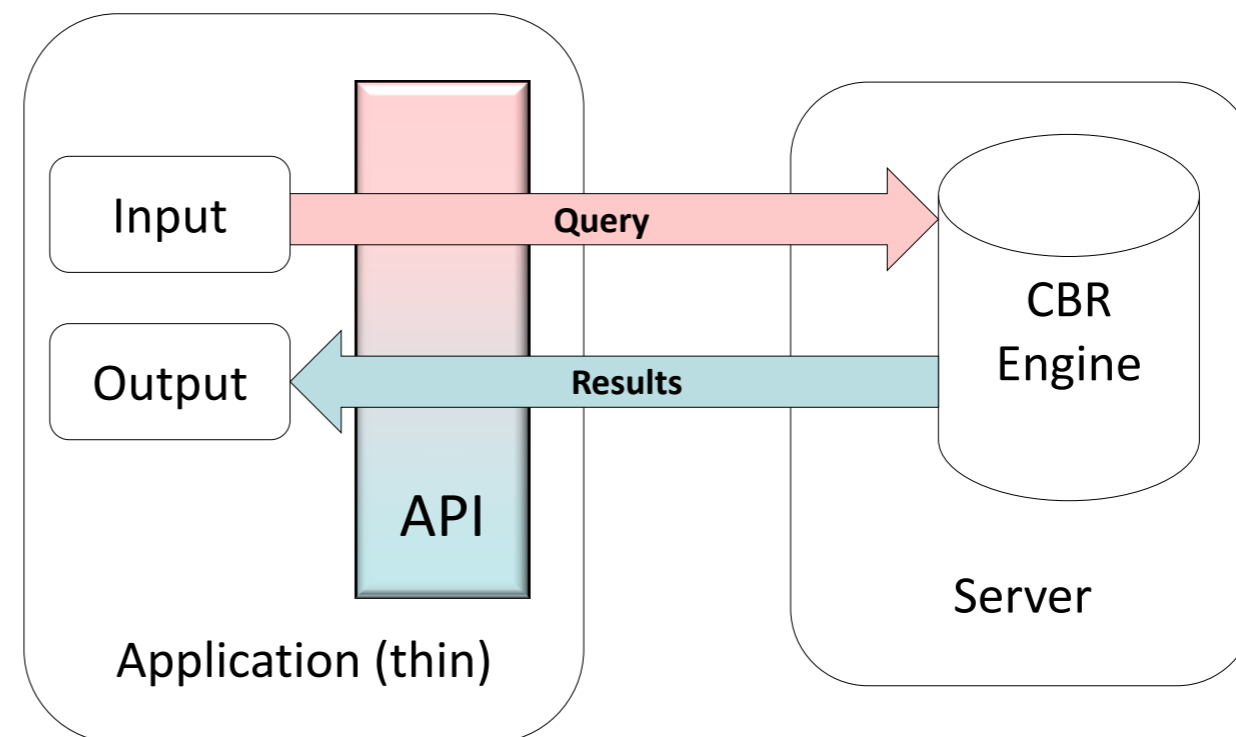
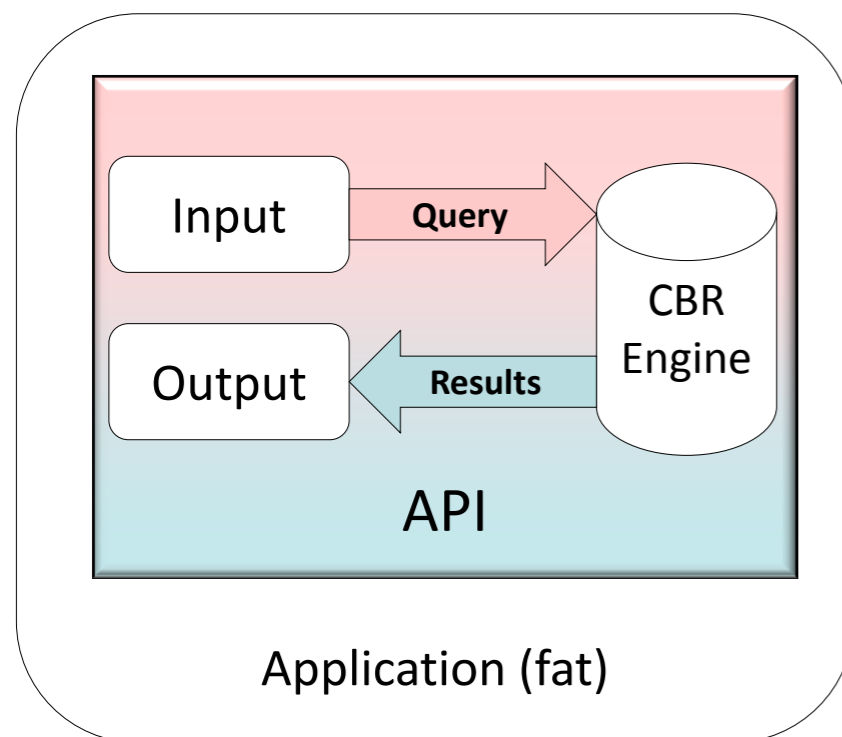
Additionally: mobile application (alpha stadium development)

There is currently a mobile centered version of myCBR under development which targets particularly the needs of Android-based development. This version of myCBR is available to developers yet.

# myCBR Application design



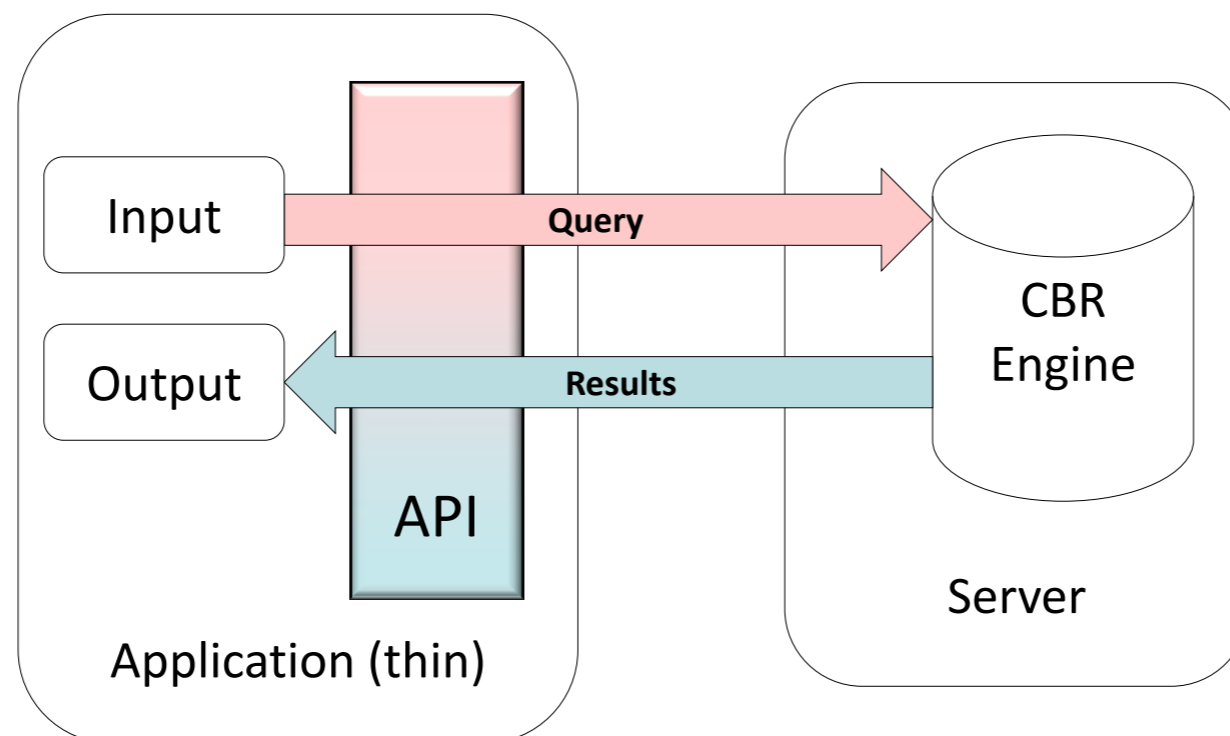
**All designs:** Create an **Instance of CBREngine** and use this to load your model data and case base. After finishing loading all data into your Instance of CBREngine you do retrievals from this CBREngine by posting query instances to it and retrieving result sets, containing lists off best matching cases (Instances).



# myCBR Application design: Thin Client

API use for building and integrating your CBR Engine

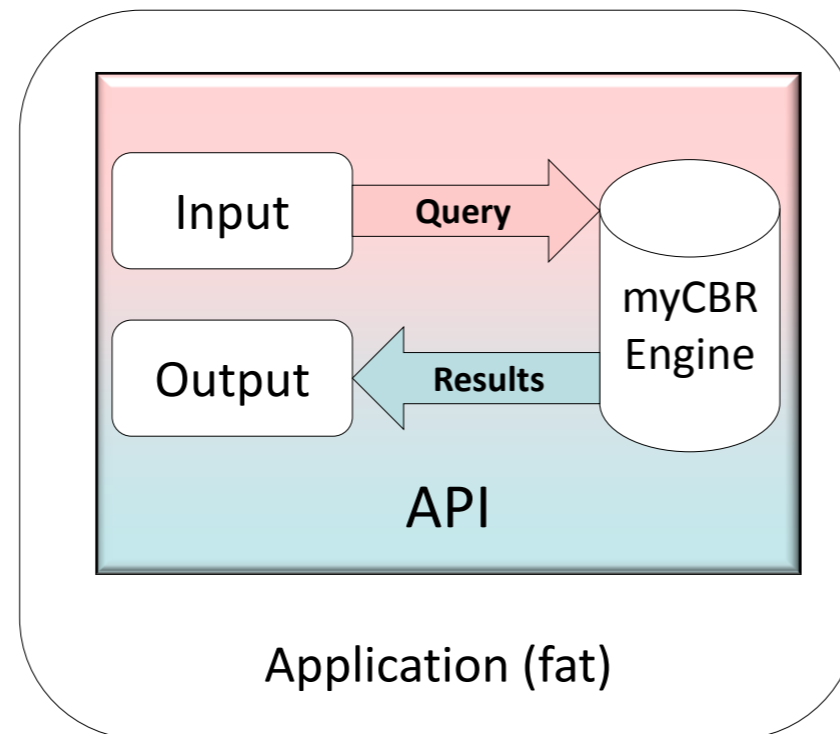
Thin Client Diagram: Web frontend , Server (myCBR) , Query-Retrieval (for example using JSP)



# myCBR Application design: Thin Client

API use for building and integrating your CBR Engine : Using the API

Fat Client Diagram: Application also hosts the CBR Engine



# myCBR SDK integration





# Loading an engine/myCBR project

Don't forget to import the necessary libraries: These are (depending on what you plan to do with your cbr engine):

Several ways to do things with the myCBR API

We chose a most simple one, which might not be most flexible or efficient but is easy to follow as a start for developers new to myCBR. Feel free to use it however you like

Our basic approach:

In your Application class: Create an Instance of CBREngine

Load model and data (case base) into this engine

Construct your Query instance using the engine

Query the engine with the query instance

Use the result (sets) returned by the engine

The following is a walkthrough of the example given in the end of this chapter.

# Overview: CBREngine



Don't forget to define your (myCBR)-project variables in your **copy (sourcecode!)** of the Class "CBREngine"

Constructor and Description
<a href="#"><u>CBREngine()</u></a>

Methods	
Modifier and Type	Method and Description
de.dfki.mycbr.core.Project	<a href="#"><u>createCBRProject()</u></a> This methods creates a myCBR project and loads the cases in this project.
de.dfki.mycbr.core.Project	<a href="#"><u>createemptyCBRProject()</u></a> This methods creates an EMPTY myCBR project.
de.dfki.mycbr.core.Project	<a href="#"><u>createProjectFromPRJ()</u></a> This methods creates a myCBR project and loads the project from a .prj file
static java.lang.String	<a href="#"><u>getCaseBase()</u></a>
static java.lang.String	<a href="#"><u>getConceptName()</u></a>
static java.lang.String	<a href="#"><u>getCsv()</u></a>
static java.lang.String	<a href="#"><u>getProjectName()</u></a>
static void	<a href="#"><u>setCasebase()</u></a> (java.lang.String casebase)
static void	<a href="#"><u>setConceptName()</u></a> (java.lang.String conceptName)
static void	<a href="#"><u>setCsv()</u></a> (java.lang.String csv)
static void	<a href="#"><u>setProjectName()</u></a> (java.lang.String projectName)



# Define your project in the CBREngine java

## Define your (myCBR)-project variables in your copy of the Class “CBREngine”

### Set path to myCBR projects:

```
//private static String data_path, for example = "C:\\myCBRprojects\\project\\";  
private static String data_path = "K:\\Example_Projects\\examples\\";
```

### Project specific variables:

```
// name of the project file  
private static String projectName = "NewExampleProject.prj";  
  
// name of the central concept  
private static String conceptName = "Car";  
  
// name of the csv containing the instances  
private static String csv = "cars_casebase.csv";  
  
// set the separators that are used in the csv file  
private static String columnseparator = ";";  
private static String multiplevalueseparator = ",";  
  
// name of the case base that should be used; the default name in myCBR is CB_csvImport  
private static String casebase = "CarsCB";
```

# Start coding your Application class with imports

Don't forget to import the necessary libraries (Check with Ctrl+Shift+o in Eclipse)  
Usually these libraries are:

```
import java.text.ParseException;
import java.util.List;
import de.dfki.mycbr.core.Project;
import de.dfki.mycbr.core.casebase.Instance;
import de.dfki.mycbr.core.model.Concept;
import de.dfki.mycbr.core.model.FloatDesc;
import de.dfki.mycbr.core.model.IntegerDesc;
import de.dfki.mycbr.core.model.SymbolDesc;
import de.dfki.mycbr.core.retrieval.Retrieval;
import de.dfki.mycbr.core.retrieval.Retrieval.RetrievalMethod;
import de.dfki.mycbr.core.similarity.Similarity;
import de.dfki.mycbr.io.CSVImporter;
import de.dfki.mycbr.core.*;
import de.dfki.mycbr.core.model.*;
import de.dfki.mycbr.util.Pair;
import de.dfki.mycbr.io.CSVImporter;
```



# Loading a project into the Engine and preparing the data

//Within your Application class: Create your instance of the CBR ENGINE (named „engine“ here)

```
CBREngine engine = new CBREngine();
```

//Create a Project (named “rec” here) and load all necessary data into it

```
Project rec = engine.createProjectFromPRJ();
```

// create a case base (named “cb” here)

// and assign the case base to be used for submitting a query

```
DefaultCaseBase cb = (DefaultCaseBase)rec.getCaseBases().get(engine.getCaseBase());
```

// create a Concept (named “myConcept” here) of the main Concept (type) of the Project;

```
Concept myConcept = rec.getConceptByID(engine.getConceptName());
```

# Prepare a query I: Retrieval and query instance



// create a new retrieval (called “ret” here)

Retrieval **ret** = new Retrieval(**myConcept**, **cb**);

// specify the retrieval method

**ret**.setRetrievalMethod(RetrievalMethod.*RETRIEVE\_SORTED*);

// available retrieval methods are: RETRIEVE , RETRIEVE\_SORTED, RETRIEVE\_K ,  
RETRIEVE\_K\_SORTED

// create a query instance (named “query” here) / A query instance essentially is a case that will

// have assigned the values to its attributes that specify the query

Instance **query** = **ret**.getQueryInstance();



## Prepare a query II: Insert values

// Insert values into the query: Symbolic Description

```
SymbolDesc colorDesc = (SymbolDesc) myConcept.getAllAttributeDescs().get("Color");  
query.addAttribute(colorDesc,colorDesc.getAttribute("Green"));
```

// Insert values into the query: Float Description

```
FloatDesc priceDesc = (FloatDesc) myConcept.getAllAttributeDescs().get("Price");  
try {query.addAttribute(priceDesc,priceDesc.getAttribute("4799.0"));}  
catch (ParseException e) {e.printStackTrace();}
```

// Insert values into the query: Float Description

```
FloatDesc mileageDesc = (FloatDesc) myConcept.getAllAttributeDescs().get("Mileage");  
try {query.addAttribute(mileageDesc,mileageDesc.getAttribute("10000"));}  
catch (ParseException e) {e.printStackTrace();}
```



# Perform retrieval and use result

See the example JSP project for a more complex use and display of the retrieval result.

```
// perform retrieval
```

```
ret.start();
```

```
// get the retrieval result as a List (named "result" here)
```

```
List <Pair<Instance, Similarity>> result = ret.getResult();
```

```
if( result.size() > 0 ){
```

```
    String casename = result.get(0).getFirst().getName(); // get the case name
```

```
    Double sim = result.get(0).getSecond().getValue(); // get the similarity value
```

```
    answer = "I found "+casename+" with a similarity of "+sim+" as the best match.";
```

```
}
```

```
else{ System.out.println("Retrieval result is empty"); }
```

# Selection and use of Amalgamation functions



Remember: An amalgamation function is a weighted sum of all local similarities (attribute similarities) of a concept that constitutes the overall global similarity measure of the concept.

Sometimes it can be useful to be able to switch between different amalgamation functions, for example to comply to different user preferences within different user groups. The myCBR API allows you to access and use different amalgamation functions, which you have modelled before using the workbench.

//List all available amalgamation functions for a concept

```
List<AmalgamationFct> liste = myConcept .getAvailableAmalgamFcts();
```

//Set an amalgamation function to be used for the similarity computation of a concept

```
myConcept.setActiveAmalgamFct(Amalgamation function amalgam);
```



# API in general: loading a myCBR project

If you don't want to use the CBREngine class you find the general API command to handle projects etc. In the following slides:

```
// To load new project use:  
// data_path thereby pointing to the path where the project is stored+the projects name
```

```
Project myproject = new Project(data_path+projectName);
```

```
// To create a concept and get the main concept of the project.  
// The name (conceptName) has to be specified at the beginning of your class-code
```

```
Concept myconcept = project.getConceptByID(conceptName);
```

```
// Initialize CSV Import with this code: (data_path: Path to your csv folder + the csv-  
file itself
```

```
CSVImporter mycsvImporter = new CSVImporter(data_path+csv, myconcept);
```

```
// To set the separators that are used in the csv file use these codes.
```

```
csvImporter.setSeparator(columnseparator); // column separator  
csvImporter.setSeparatorMultiple(multiplevalueseparator); // multiple value separator
```



# API in general: importing project data

// To prepare the data for the import of the project data in csv-format you can use these methods:

```
csvImporter.readData();           // read csv data
csvImporter.checkData();          // check formal validity of the data
csvImporter.addMissingValues();   // add missing values with default values
csvImporter.addMissingDescriptions(); // add default descriptions
```

// Finally to do the import of the instances of the Concept defined use:

```
csvImporter.doImport();           // Import the data into the project
```



# API in general: querying your project

Initiate a query :

```
// create a new retrieval
Retrieval ret = new Retrieval(myConcept, cb);

// specify the retrieval method
ret.setRetrievalMethod(RetrievalMethod.RETRIEVE_SORTED);
// available retrieval methods are: RETRIEVE , RETRIEVE_SORTED, RETRIEVE_K , RETRIEVE_K_SORTED

// create a query instance
Instance query = ret.getQueryInstance();
```



# API in general: querying your project

Insert values into the query:

```
// Symbolic Description
```

```
SymbolDesc manufacturerDesc = (SymbolDesc) myConcept.getAllAttributeDescs().get("Manufacturer");  
query.addAttribute(manufacturerDesc, manufacturerDesc.getAttribute("bmw"));
```

```
// Float Description
```

```
FloatDesc priceDesc = (FloatDesc) myConcept.getAllAttributeDescs().get("Price");  
query.addAttribute(priceDesc, priceDesc.getAttribute("47699.0"));
```

```
// Int Description
```

```
IntegerDesc yearDesc = (IntegerDesc) myConcept.getAllAttributeDescs().get("Year");  
query.addAttribute(yearDesc, yearDesc.getAttribute("1996"));
```



# API in general: querying your project

Insert values into the query (continued):

```
// String Description
StringDesc titleDesc = (StringDesc) myConcept.getAllAttributeDescs().get("Title");
query.addAttribute(titleDesc, titleDesc.getAttribute("Cheese-Crusted Chicken with
Cream"));

// Symbolic Description and multiple Values
SymbolDesc methodDesc = (SymbolDesc) myConcept.getAllAttributeDescs().get("Method");
// define a list that will be used to store the values
LinkedList<Attribute> list = new LinkedList<Attribute>();
// add query values to the list
list.add(methodDesc.getAttribute("roast"));
list.add(methodDesc.getAttribute("arrange"));
list.add(methodDesc.getAttribute("warm up"));
list.add(methodDesc.getAttribute("stir"));
list.add(methodDesc.getAttribute("cook"));
list.add(methodDesc.getAttribute("melt"));
// create a multiple attribute and add the attribute's description and the specified
list
MultipleAttribute<SymbolDesc> mult = new MultipleAttribute<SymbolDesc>(methodDesc,
list);
// add the query attribute to the list
query.addAttribute(methodDesc.getName(), mult);
```



# API in general: execute a query / use result

Execute the query (do the retrieval):

```
// perform retrieval
```

```
ret.start();
```

```
// get the retrieval result
```

```
List<Pair<Instance, Similarity>> result = ret.getResult();
```

```
// get the case name
```

```
result.get(0).getFirst().getName();
```

```
// get the similarity value
```

```
Result.get(0).getSecond().getValue();
```



# API in general: accessing the model

```
// get all attributes of the CBR case model
HashMap<String, AttributeDesc> valueMap = myConcept.getAllAttributeDescs();

// get the allowed values for each Attribute
for (Map.Entry<String, AttributeDesc> entry: valueMap.entrySet()) {
    System.out.println(entry.getValue());
    AttributeDesc attdesc = entry.getValue();
    String attClass = attdesc.getClass().toString();

    if (attClass.compareTo("class de.dfki.mycbr.core.model.SymbolDesc")==0) {
        SymbolDesc symbolDesc = (SymbolDesc) entry.getValue();
        Set<String> elements = symbolDesc.getAllowedValues();

        for (String allowedValue : elements) {
            System.out.println("\t\t"+allowedValue);
        }
    }
}
```



# myCBR SDK integration

Directly create a project and enter data into it, create a query and run the retrieval in its most compact form:

```
// requires myCBR 3.1
Project p = new Project();
// Create Concept Car
Concept car = p.createTopConcept("Car");
// add symbol attribute
HashSet<String> manufacturers = new HashSet<String>();
String[] manufacturersArray = { "BMW", "Audi", "VW", "Ford", "Mercedes", "SEAT", "FIAT" };
manufacturers.addAll(Arrays.asList(manufacturersArray));
SymbolDesc manufacturerDesc = new SymbolDesc(car, "manufacturer", manufacturers);
// add table (similarity) function
SymbolFct manuFct = manufacturerDesc.addSymbolFct("manuFct", true);
manuFct.setSimilarity("BMW", "Audi", 0.60d);
manuFct.setSimilarity("Audi", "VW", 0.20d);
manuFct.setSimilarity("VW", "Ford", 0.40d);
// add case base
DefaultCaseBase cb = p.createDefaultCB("myCaseBase");
// add case
Instance i = car.addInstance("car1");
i.addAttribute(manufacturerDesc, manufacturerDesc.getAttribute("BMW"));
cb.addCase(i, "car1");
// set up query and retrieval
Retrieval r = new Retrieval(car);
Instance q = r.getQuery();
q.addAttribute(manufacturerDesc.getName(), manufacturerDesc.getAttribute("Audi"));
r.start();
// r now contains the retrieved best case(s)
```

# myCBR workbench: GUI elements



# Main View Elements

## File Menu:

Open/Save Projects  
Model: Actions available for Models, Concepts, Attributes

## Shortcuts for:

Create, Open and Open recent projects

## The Projects View:

This view provides an overview of your currently opened projects in a tree structure. It further provides all necessary actions to add/remove elements to/from your model.

## The Similarity Measure view:

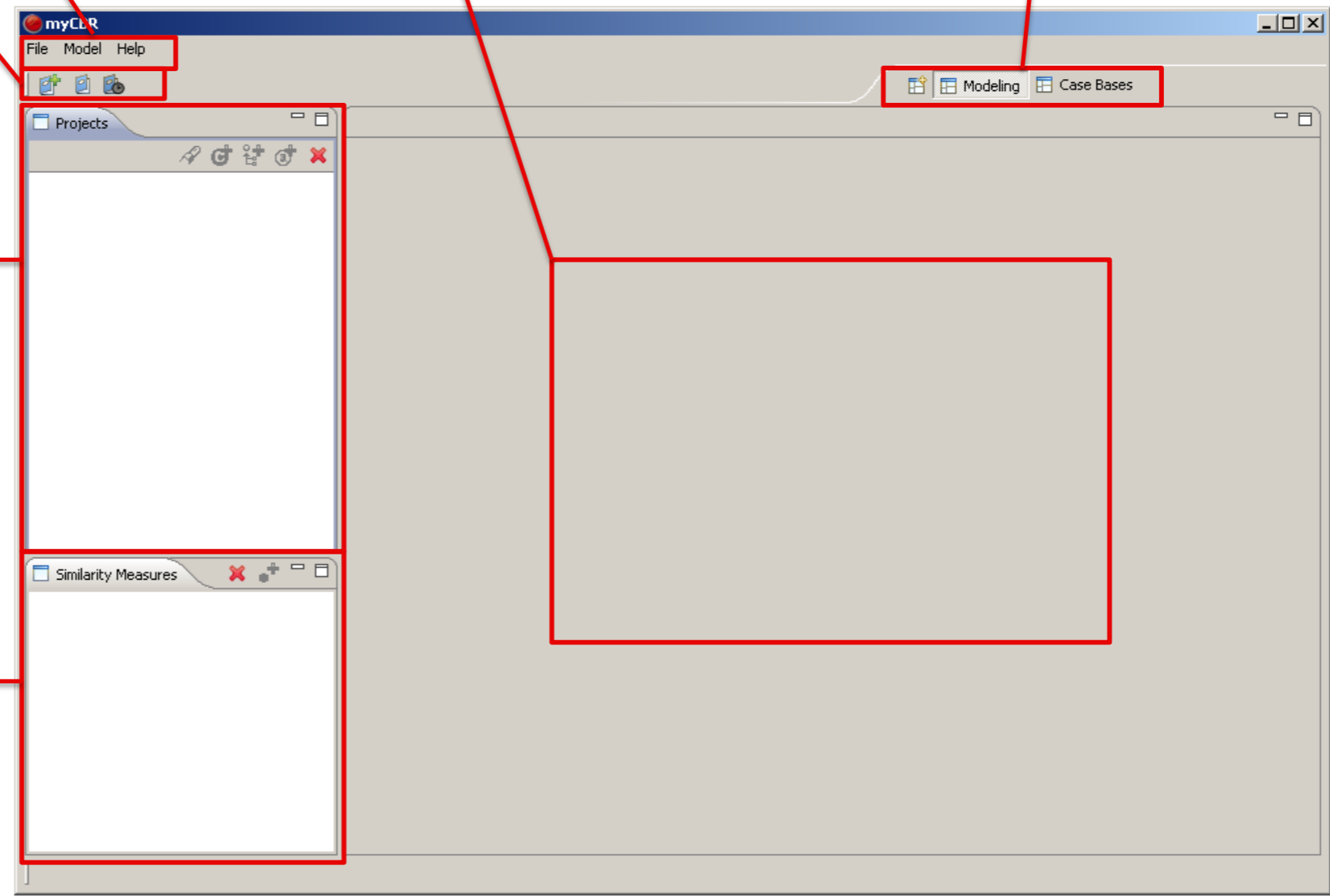
This view shows you the assigned Similarity Measures for a selected Project, Concept or Attribute. It further provides the actions to add or remove Similarity Measures to the selected Project, Concept or Attribute.

## Main View:

This is where you get detailed information and ways for interactions with your model for a selected component of your model.

## Perspective Tabs:

Selecting the Tabs switches between the different available Perspectives in myCBR. The two main perspectives now are: Modelling, where you design your model and Case Bases where you create and optimise your Cases and Case bases



# Projects View Elements

Start a retrieval: Starts the retrieval task for the selected project

Add Concept: Add a new concept (thing) to the selected project

Add a Concept as Attribute: Add a Concept as an Attribute to the selected Concept: think of Part-of relations

Add an Attribute: Add an Attribute to a selected Concept

Delete: Delete the selected Component (Either Concept or Attribute) Every dependent Component will be deleted as well

The Project Tree:

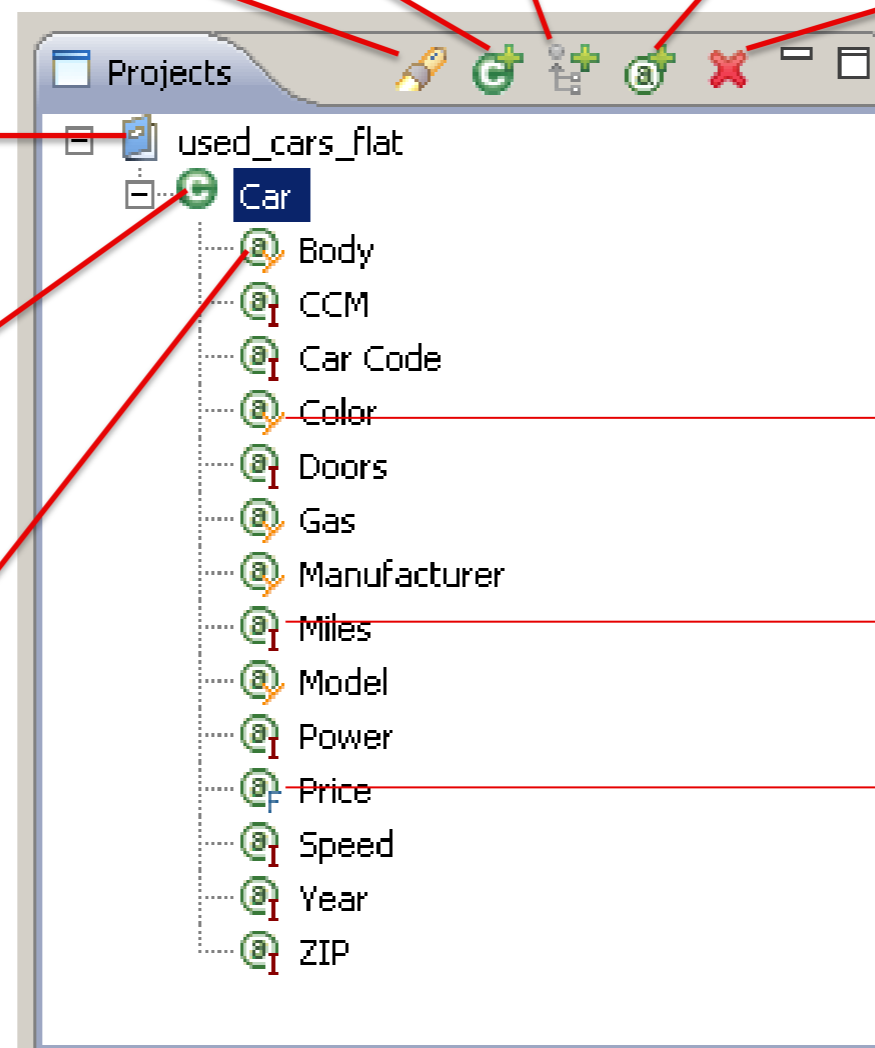
This Tree lists every Concept within the project, a DoubleClick: shows detailed information about the project

The Concept Tree:

This Tree lists all Attributes of the selected Concept, a DoubleClick: shows detailed information about the concept

An Attribute:

A DoubleClick: shows detailed information about the Attribute.



Symbol Attribute Icon

Integer Attribute Icon

Float Attribute Icon

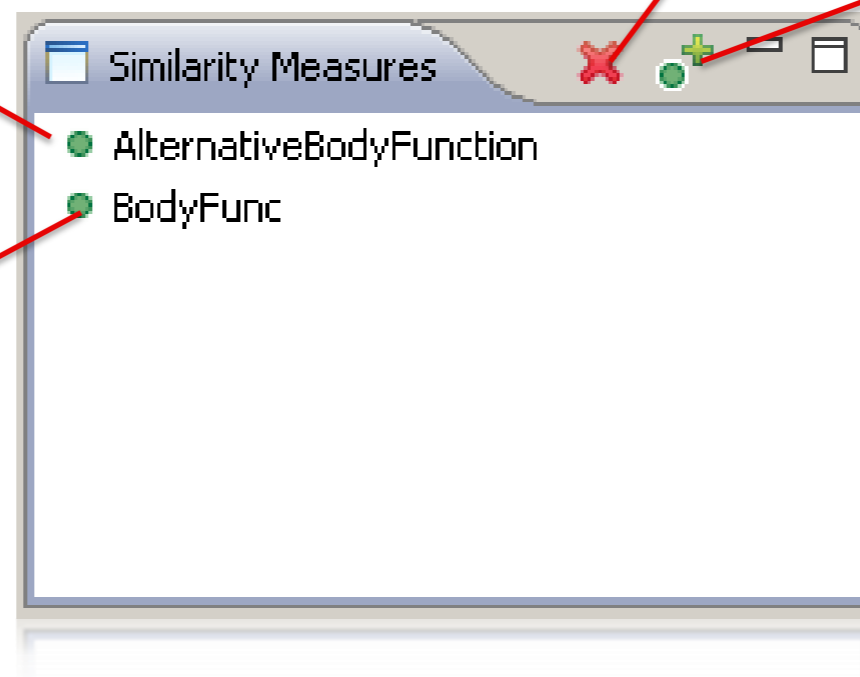
# Similarity Measures View

Example of an added alternative Similarity Measure

The generated default Similarity Measure (In this Example for the Attribute 'Body' of the Car Concept)

Delete a selected Similarity Measure

Add a Similarity Measure for the selected component. To be able to add a Measure the default generated Measure must be selected.



Tabs for the quick navigation  
 to the different views/editors  
 of a selected component

The screenshot displays the 'Body' attribute configuration window. The 'Name' field is set to 'Body' and the 'Type' is 'Symbol'. The 'Multiple' checkbox is unchecked. The 'Allowed Values' table lists: convertible, coupe, fastback, roadster, sedan, and station\_wagon. The 'Add', 'Rename', and 'Remove' buttons are visible on the right. The 'Projects' pane on the left shows the 'used\_cars\_flat' project with a tree view where 'Body' is selected under 'Car'. The 'Similarity Measures' pane at the bottom left lists 'AlternativeBodyFunction' and 'BodyFunc'. The bottom of the window shows a tabbed interface with 'Attribute', 'Symbol Explanation', and 'Concept Explanation' tabs.

# Edit similarity measures: Symbol, table function

The table editor is used to describe the similarity mode table. This similarity mode can be chosen for the slot type symbol.

If there are only a few values for your slot which can't be ordered absolutely or hierarchically, you should use the table editor.

Symmetry: Choosing symmetric makes the similarity matrix symmetric

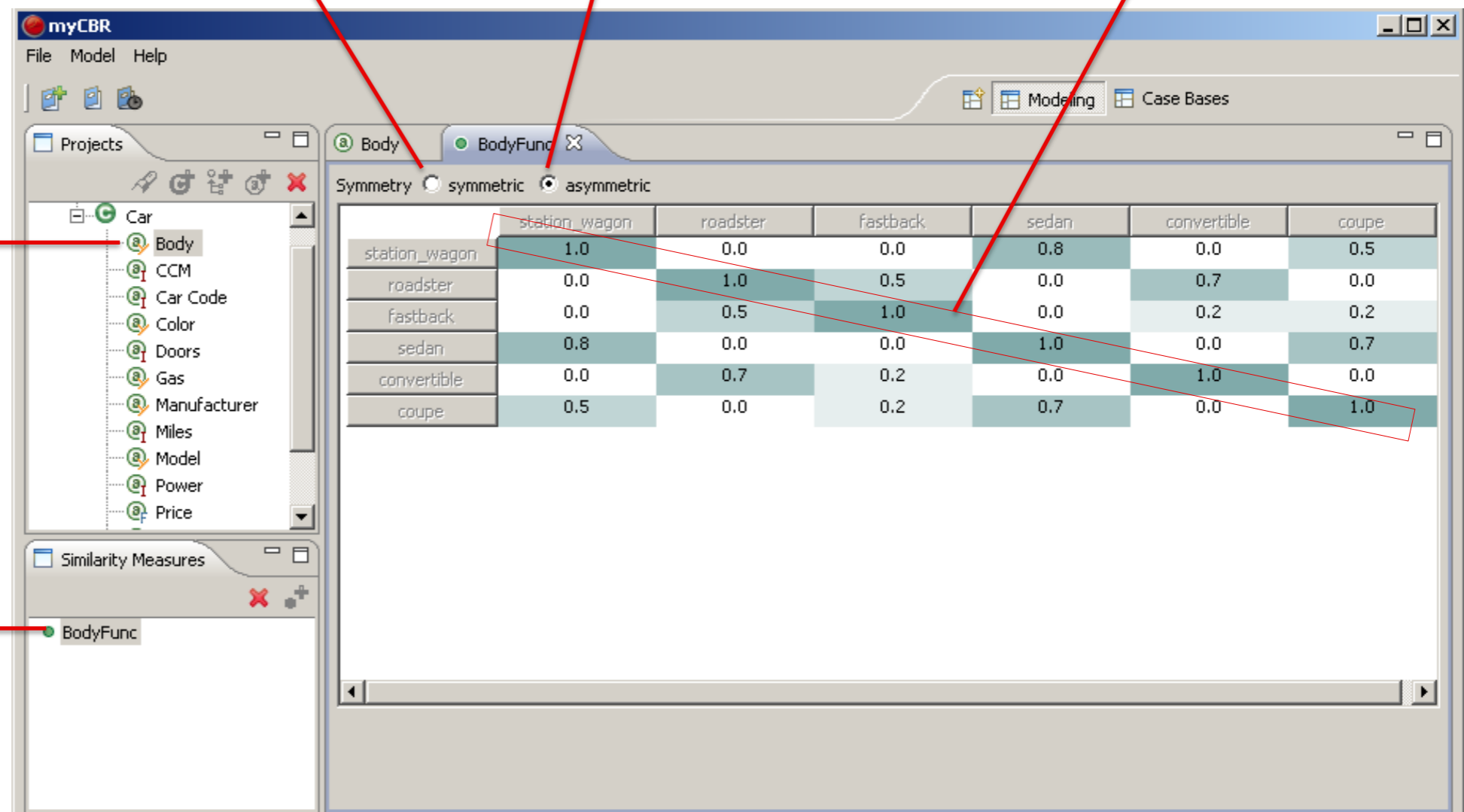
Symmetry: Choosing asymmetric makes the similarity matrix asymmetric

The diagonal which splits the either symmetric or asymmetric halves of the similarity matrix.

The colours of the fields are an optical aid to visualise the values.

The Attribute 'Body' of the Concept 'Car' is selected

The default (auto generated) similarity measure for the attribute 'body' is selected



The screenshot shows the myCBR 3.0 interface. On the left, the 'Projects' pane shows a tree structure with 'Car' selected, and 'Body' is highlighted under it. Below it, the 'Similarity Measures' pane shows 'BodyFunc' selected. The main window displays a table editor for the 'Body' attribute, showing a 6x6 similarity matrix. The matrix is symmetric, with values ranging from 0.0 to 1.0. The diagonal elements are all 1.0. The off-diagonal elements represent the similarity between different car body types.

	station_wagon	roadster	fastback	sedan	convertible	coupe
station_wagon	1.0	0.0	0.0	0.8	0.0	0.5
roadster	0.0	1.0	0.5	0.0	0.7	0.0
fastback	0.0	0.5	1.0	0.0	0.2	0.2
sedan	0.8	0.0	0.0	1.0	0.0	0.7
convertible	0.0	0.7	0.2	0.0	1.0	0.0
coupe	0.5	0.0	0.2	0.7	0.0	1.0

# Edit similarity measures: Symbol, Taxonomy

Symmetry: Selects if the similarity is symmetric or asymmetric ( $f()$  or  $(f(),g())$ )

Include the values of inner (abstract) nodes in the similarity calculation. An inner node is for example: blue, as an abstract (class) of light\_blue and dark\_blue

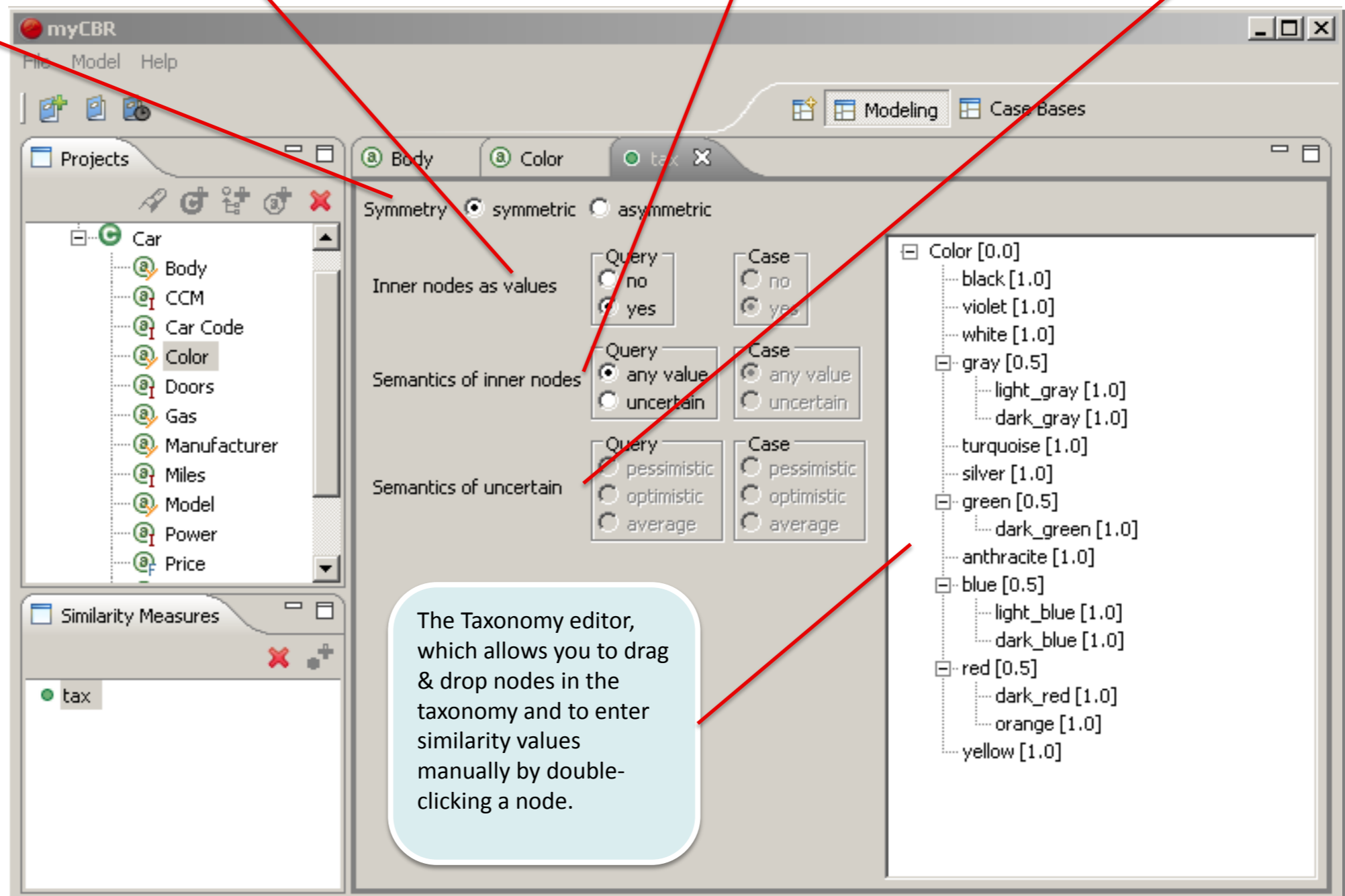
Select how to handle values for inner (abstract) nodes. Any Value: Any node below this node is to be considered. "I search for a blue car (but don't mind if it is light or dark blue)".

Uncertain: Opens up access to the "Semantics of uncertain" values below to choose one semantic (See next slide)

The taxonomy editor is used to describe the similarity mode taxonomy. This similarity mode can be chosen for the slot type symbol.

You could use taxonomy as similarity mode in case the slot's values can be arranged in a hierarchical structure, such that:

- nodes on same levels are disjoint sets
- nodes on the last level are real-world objects
- inner nodes consist of the real-world objects that follow in the hierarchical order



The Taxonomy editor, which allows you to drag & drop nodes in the taxonomy and to enter similarity values manually by double-clicking a node.

# Edit similarity measures: Symbol, Taxonomy

The taxonomy is to be used in an asymmetric way. This means that the taxonomy is used in two different ways, depending on the fact if the query value is greater than the value in the case or vice versa. Therefore the GUI provides you with the option to specify different uses of the taxonomy for the Query (Q) as well as the Case (C)

You can specify distinct ways to describe (calculate) the similarity for the query and for the case comparison.

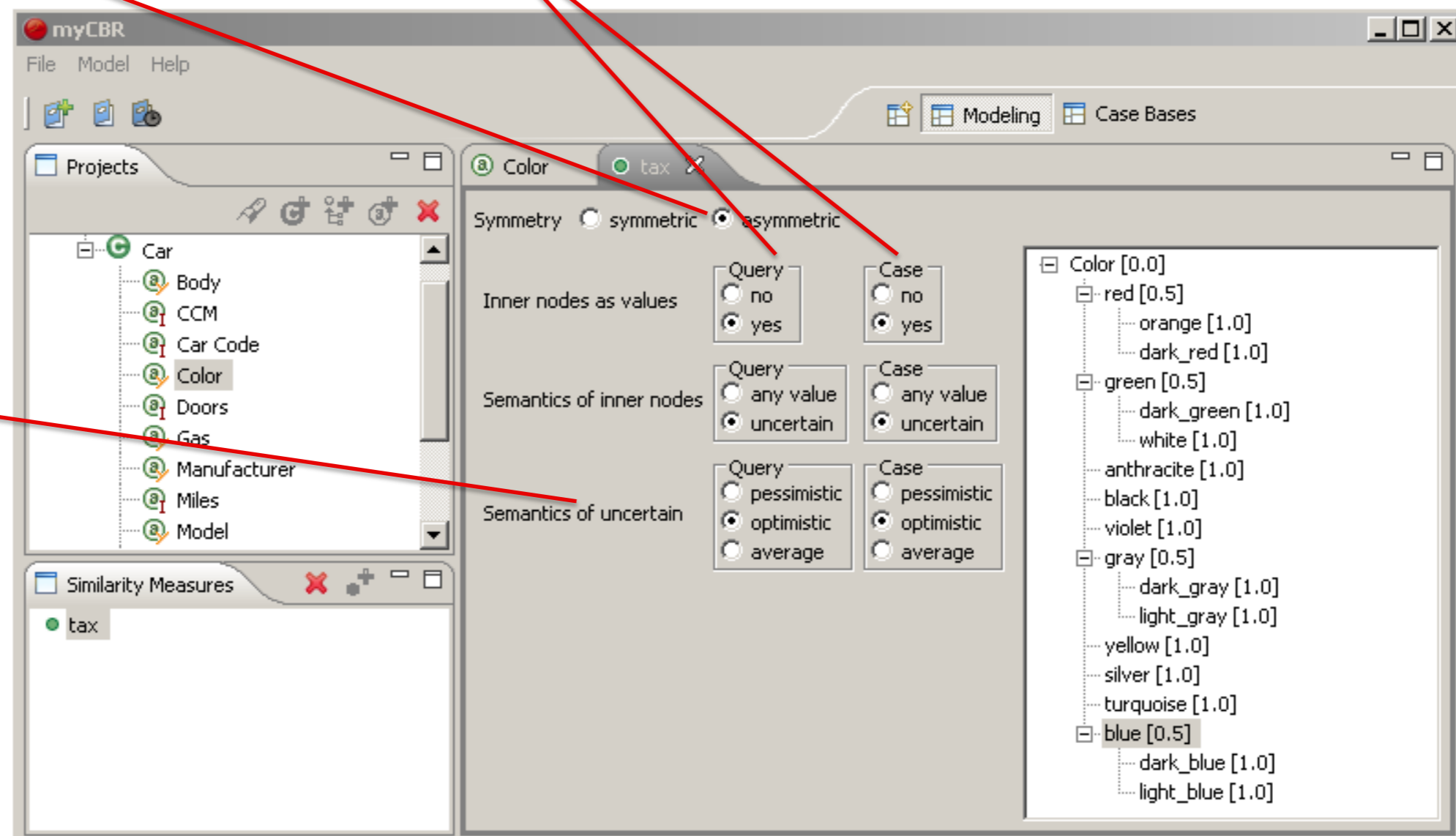
The taxonomy editor automatically creates the table used for the table editor. So you can use the taxonomy editor to initially fill the table and then switch the similarity mode to table for manually edit the table. This is very useful in case you want to use the table editor but your project structure is very complex.

If you opt for handling values of inner nodes with uncertainty you have to specify the semantic you want to use to handle the uncertainty:

Pessimistic: Use the lower bound of the similarity (Q,C)

Optimistic: Use the upper bound of the similarity (Q,C)

Average: Use the average between the lower and upper bound



# Edit similarity measures: Integer, Simple function

Symmetry allows you to choose to specify a symmetric or asymmetric function, choosing asymmetric activates the input for both sides of the function ( $C < Q$  and  $C > Q$ )

Difference calculates a numerical value for the difference between Q)uery and (C)ase values  
Quotient calculates a quotient out of the (Q)uery and (C)ase values

Left side: The function specifies the similarity for (C)ase values lower than the (Q)uery value

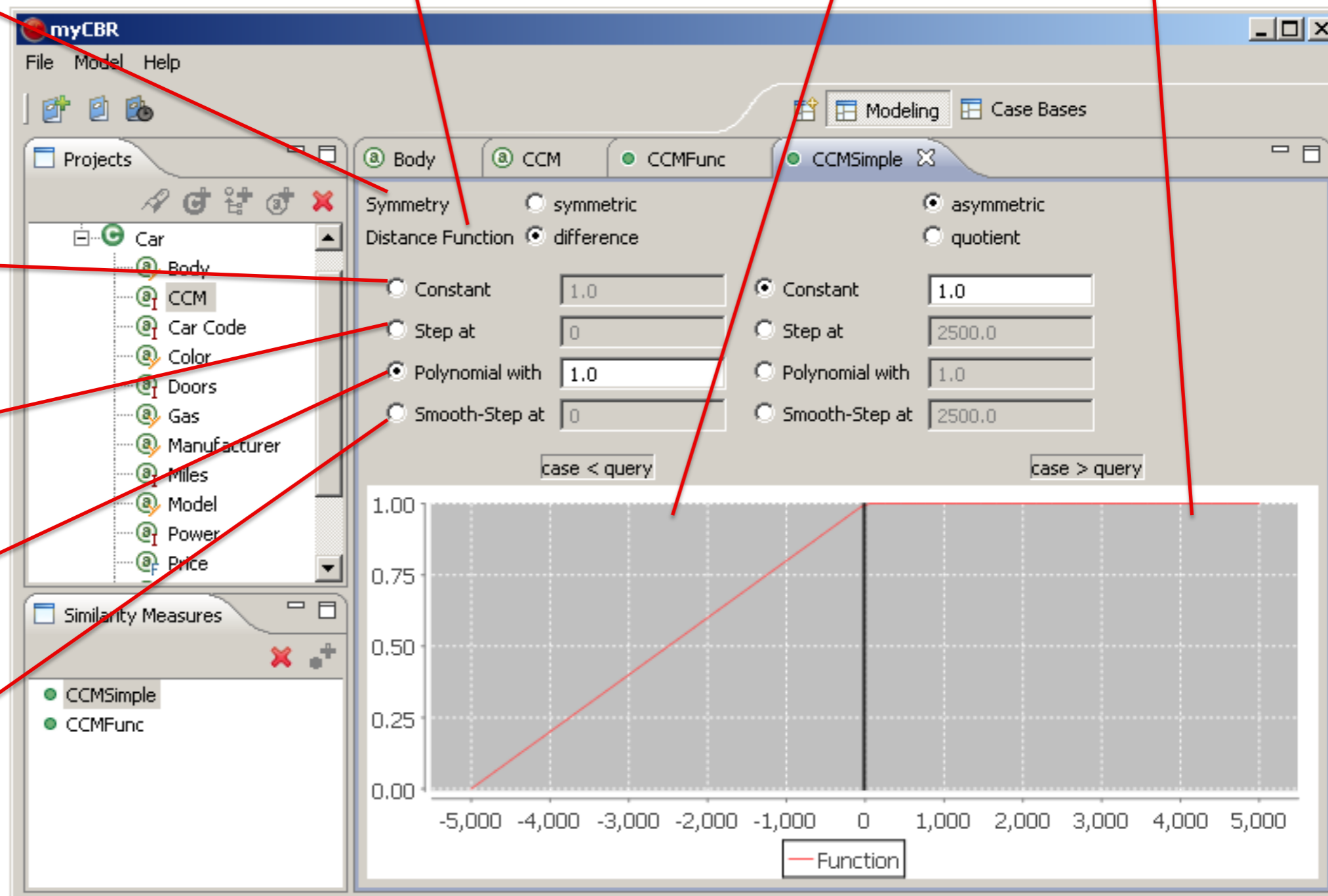
Right side: The function specifies the similarity for (C)ase values higher than the (Q)uery value

Constant: Enter a value that the function will return as a constant

Modell a step in the function and the value at which it should occur

Modell the polynomial change of similarity with a basic value

Modell a smooth step in the function at a given value



# Edit similarity measures: Integer, Advanced function

The advanced similarity mode can be chosen for the attribute types integer and float.

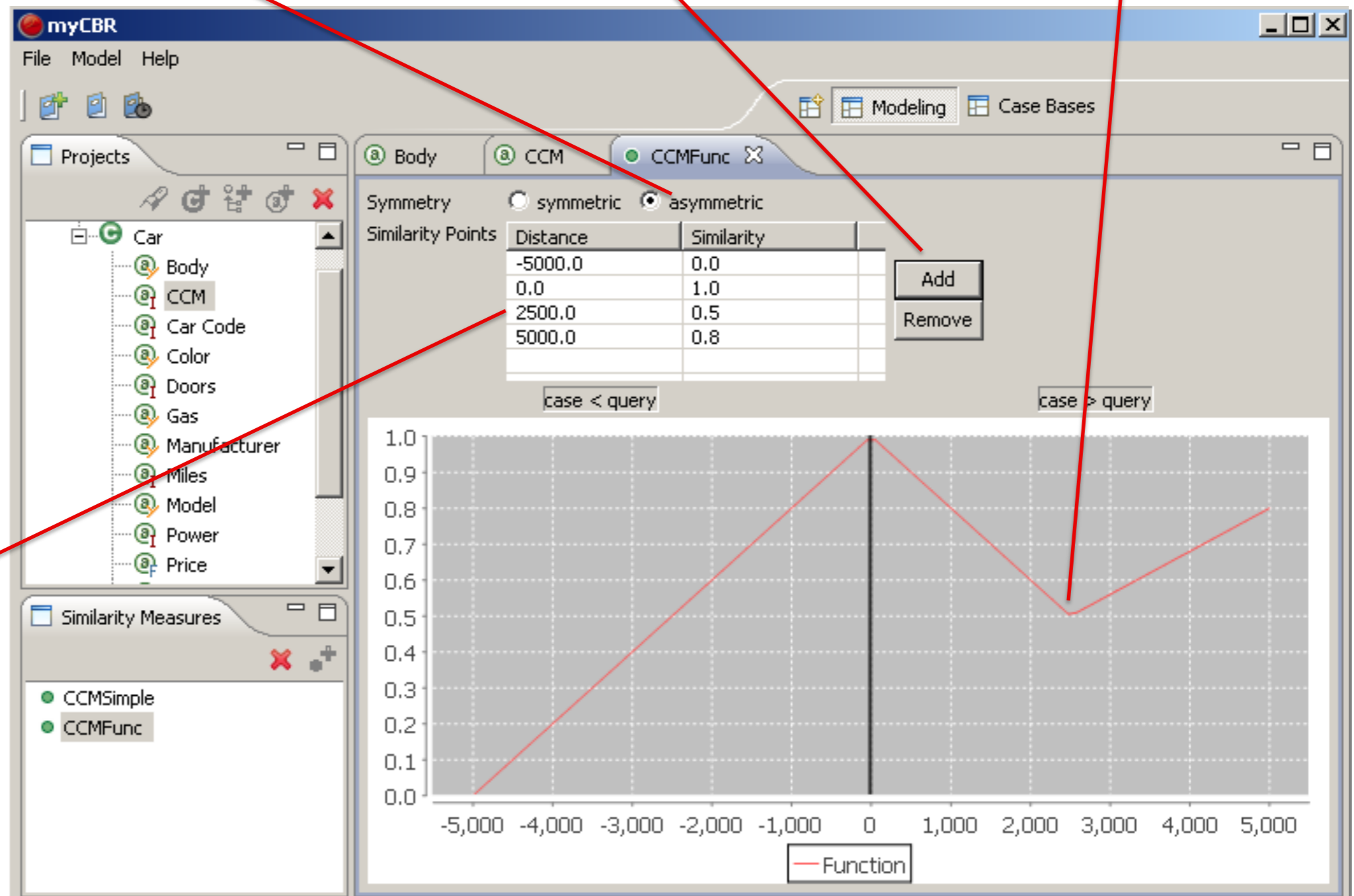
You should use advanced similarity mode in case your similarity measure function cannot be represented by the standard similarity mode.

This is the List of your added similarity points to the function. You can remove points by selecting the entry in the table and then click 'Remove'.

As you are modelling the function freely you should always select asymmetric for advanced integer and float functions

Add a new similarity point to the function. To add it click the button and then enter the function value (Distance) and the desired similarity for this value.

You can add similarity points and the result will be an interpolated function for your similarity measure.



# Edit similarity measures: Global Similarity Measure

Options for the Global similarity measure are:

Weighted sum: use weights on the attributes

Euclidian

Minimum: of the local similarities ( $\max(\text{weight} * \text{local\_sim})$ )

Minimum: of the local similarities ( $\min(\text{weight} * \text{local\_sim})$ )

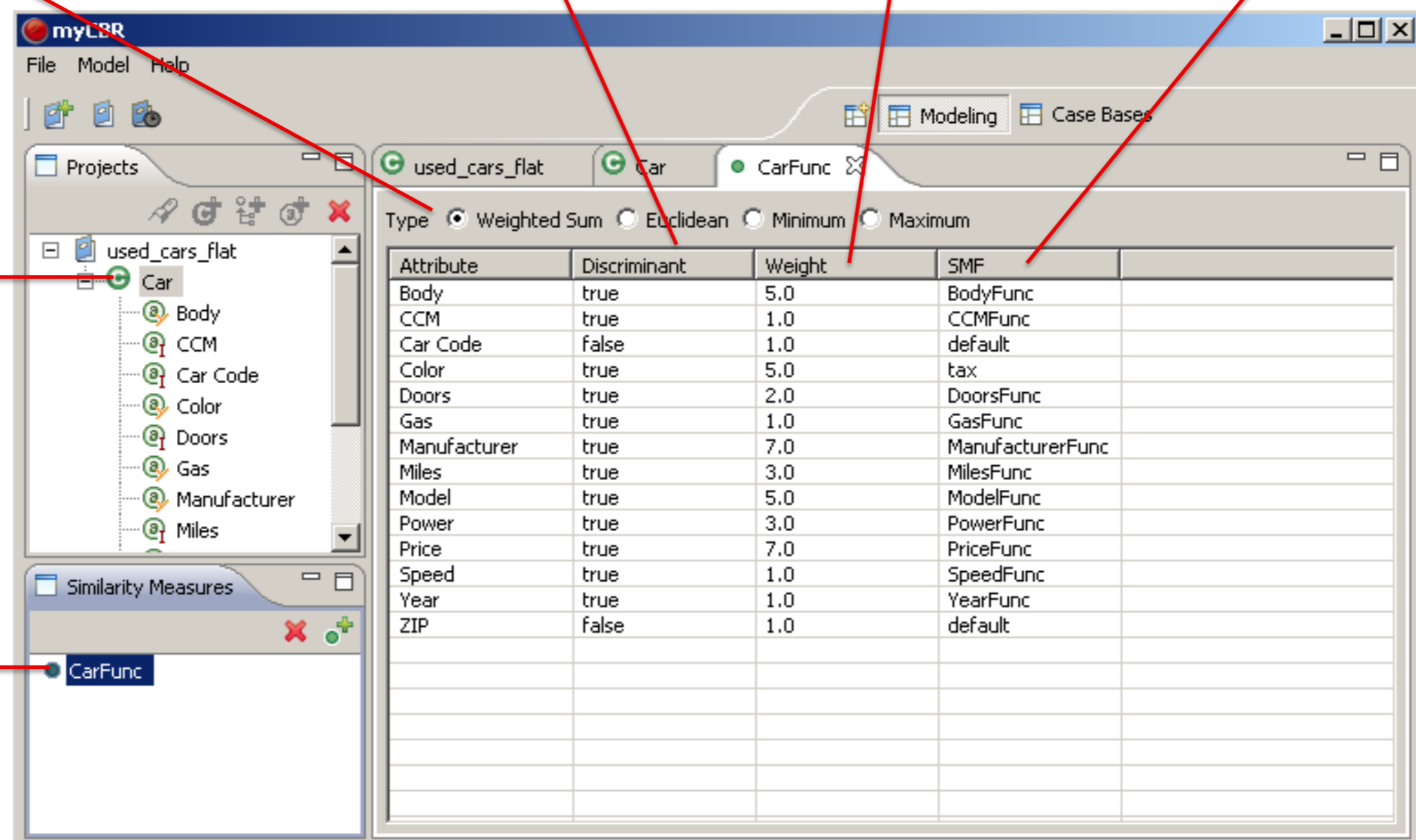
Set the Attribute as to be included in the global similarity calculation or not

Set the weight of the attribute (It's a good idea to use 100 as an overall value for the case and distribute it as weights onto the attributes)

The SMF (Similarity Function) in use for this attribute, Click the field to select a SMF from the available SMF's for the Attribute

Select the concept to model the global similarity measure for. In this example the global similarity of the concept 'Car' is modelled

The global similarity function (Amalgamation function) selected for this concept. You can add alternative global similarity functions, analogue to the definition of alter-native SMF's for Attributes.



myCBR

File Model Help

Projects

used\_cars\_flat

Car

Body

CCM

Car Code

Color

Doors

Gas

Manufacturer

Miles

Similarity Measures

CarFunc

used\_cars\_flat Car CarFunc

Type ☒ Weighted Sum ☐ Euclidean ☐ Minimum ☐ Maximum

Attribute	Discriminant	Weight	SMF
Body	true	5.0	BodyFunc
CCM	true	1.0	CCMFunc
Car Code	false	1.0	default
Color	true	5.0	tax
Doors	true	2.0	DoorsFunc
Gas	true	1.0	GasFunc
Manufacturer	true	7.0	ManufacturerFunc
Miles	true	3.0	MilesFunc
Model	true	5.0	ModelFunc
Power	true	3.0	PowerFunc
Price	true	7.0	PriceFunc
Speed	true	1.0	SpeedFunc
Year	true	1.0	YearFunc
ZIP	false	1.0	default





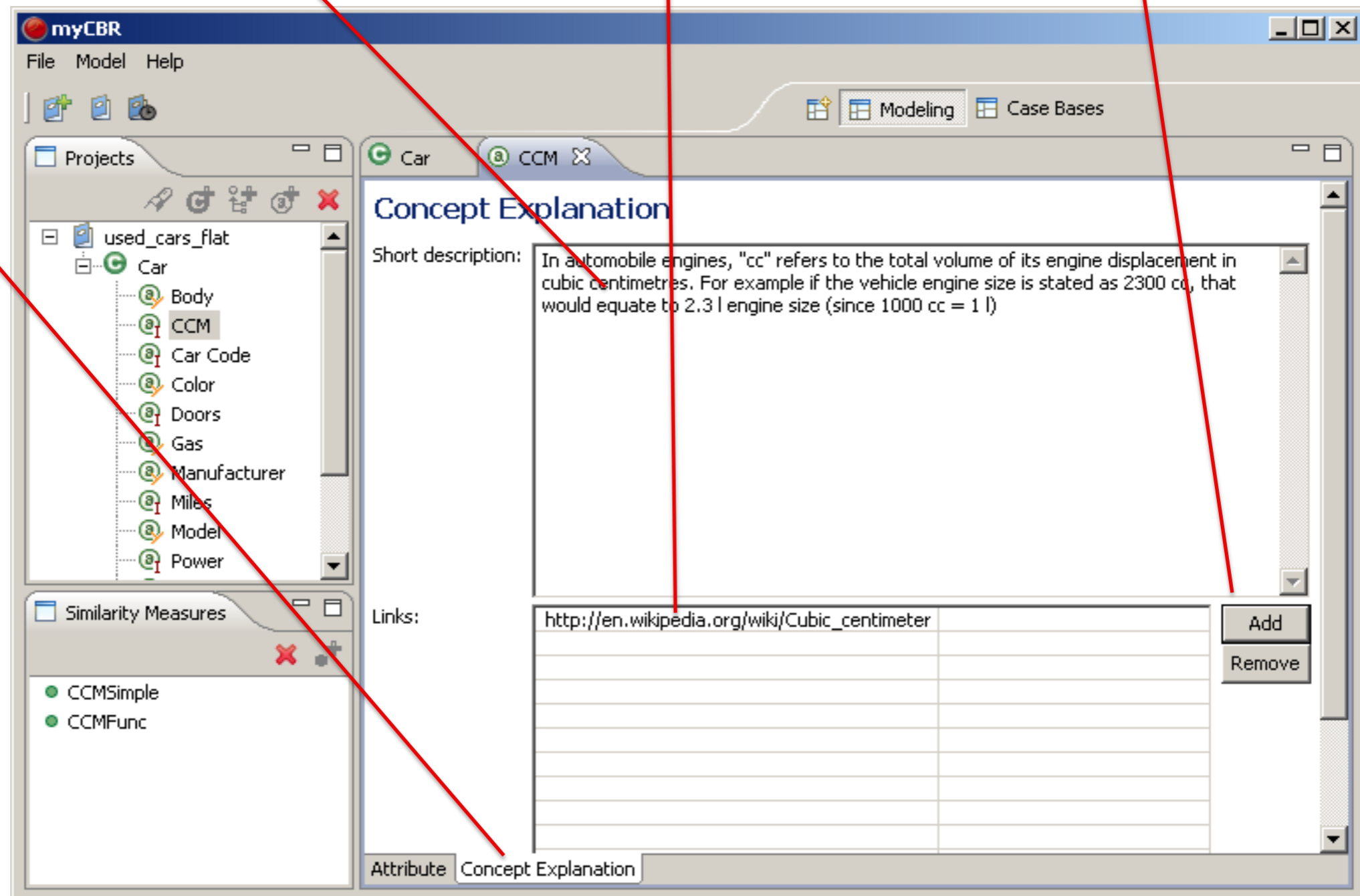
# Edit similarity measures: Attribute Explanation

Add a short descriptive text that explains the nature and/or purpose of the attribute.

Add a URL to a resource that helps explaining the attribute

Add: Opens a dialog in which you can specify further URL's that point to resources that explain the attribute.  
Remove: Removes a selected URL entry from the list.

The Concept Explanation Tab enables you to provide explanations about the selected attributes nature and/or purpose as well as to provide URL's pointing to explanatory artefacts.



The screenshot shows the myCBR software interface. The main window is titled 'myCBR' and has a menu bar with 'File', 'Model', and 'Help'. Below the menu bar are tabs for 'Modeling' and 'Case Bases'. On the left, there is a 'Projects' pane showing a tree structure with 'used\_cars\_flat' as the root, containing 'Car', 'Body', 'CCM', 'Car Code', 'Color', 'Doors', 'Gas', 'Manufacturer', 'Miles', 'Model', and 'Power'. The 'CCM' attribute is selected. Below the 'Projects' pane is a 'Similarity Measures' pane showing 'CCMSimple' and 'CCMFunc'. The main area is titled 'Concept Explanation' and contains a 'Short description:' field with the text: 'In automobile engines, "cc" refers to the total volume of its engine displacement in cubic centimetres. For example if the vehicle engine size is stated as 2300 cc, that would equate to 2.3 l engine size (since 1000 cc = 1 l)'. Below this is a 'Links:' section with a table containing one row with the URL 'http://en.wikipedia.org/wiki/Cubic\_centimeter'. To the right of the table are 'Add' and 'Remove' buttons. At the bottom, there are tabs for 'Attribute' and 'Concept Explanation', with 'Concept Explanation' being the active tab.

myCBR

File Model Help

Modeling Case Bases

Projects

- used\_cars\_flat
  - Car
    - Body
    - CCM
    - Car Code
    - Color
    - Doors
    - Gas
    - Manufacturer
    - Miles
    - Model
    - Power

Similarity Measures

- CCMSimple
- CCMFunc

Concept Explanation

Short description: In automobile engines, "cc" refers to the total volume of its engine displacement in cubic centimetres. For example if the vehicle engine size is stated as 2300 cc, that would equate to 2.3 l engine size (since 1000 cc = 1 l)

Links:

http://en.wikipedia.org/wiki/Cubic_centimeter	

Add Remove

Attribute Concept Explanation

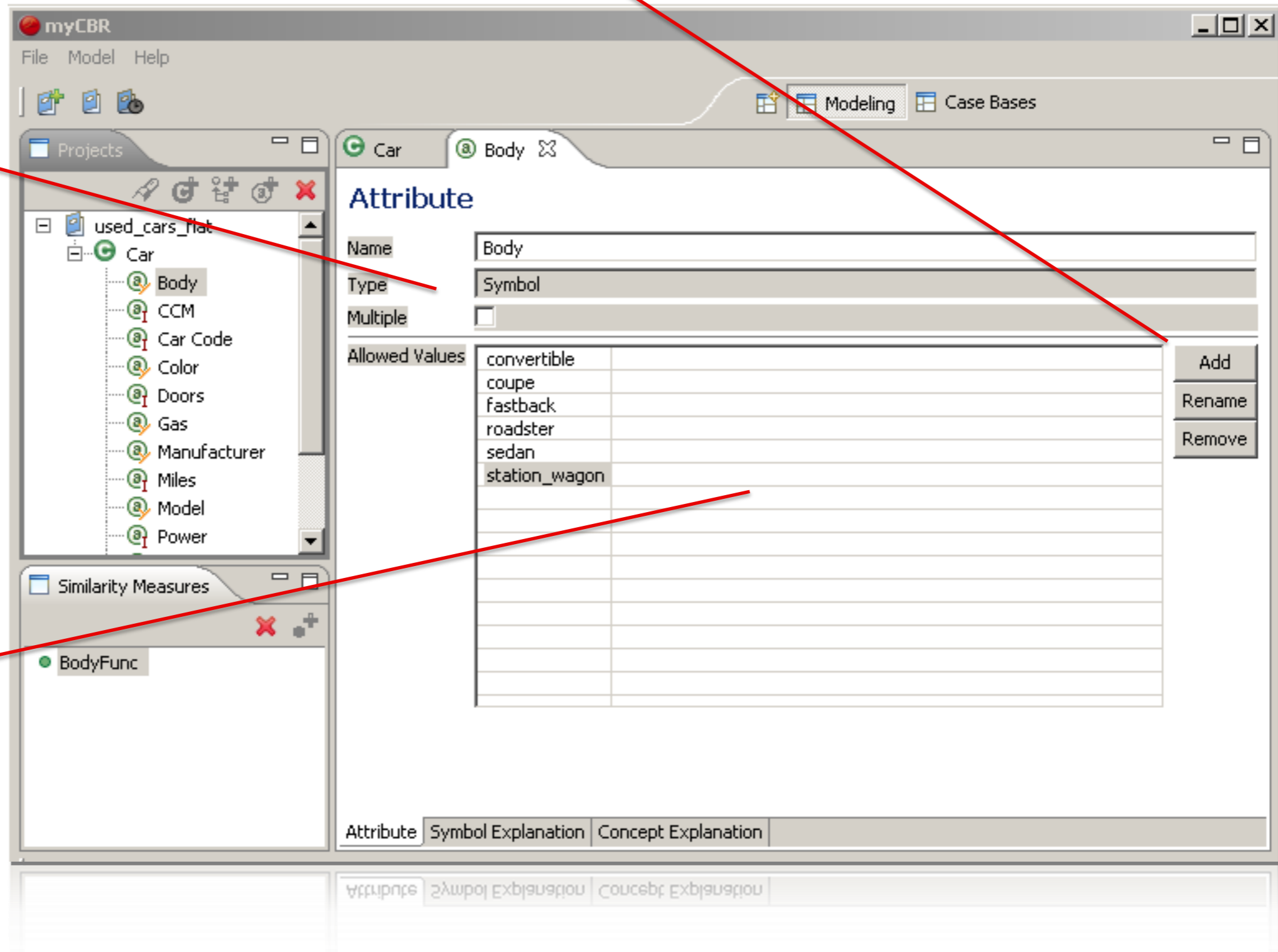
# Edit similarity measures: Define Attribute: Symbol

Add: Lets you add symbols to the list of defined symbols for the attribute  
 Note: As long as you don't cancel the dialog you will be asked to enter further symbols  
 Rename: Lets you rename a selected symbol from the list  
 Remove: Removes the selected symbol from the list

Name: Enter the name of the attribute

Type: Symbol(ic)  
 This is automatically chosen, depending on the type you specified for the attribute.

The list of values (Symbols) the attribute has



The screenshot shows the myCBR software interface. The main window is titled 'myCBR' and has a menu bar with 'File', 'Model', and 'Help'. Below the menu bar are tabs for 'Modeling' and 'Case Bases'. The 'Projects' pane on the left shows a tree structure under 'used\_cars\_flat' with a 'Car' attribute. The 'Attribute' dialog box is open, showing the 'Name' field with 'Body', the 'Type' field with 'Symbol', and the 'Multiple' checkbox unchecked. The 'Allowed Values' list contains: convertible, coupe, fastback, roadster, sedan, and station\_wagon. To the right of the list are buttons for 'Add', 'Rename', and 'Remove'. The 'Similarity Measures' pane at the bottom left shows 'BodyFunc'.

Attribute	Symbol Explanation	Concept Explanation
Body	convertible	
	coupe	
	fastback	
	roadster	
	sedan	
	station_wagon	

# Edit similarity measure: Define Attribute: Integer

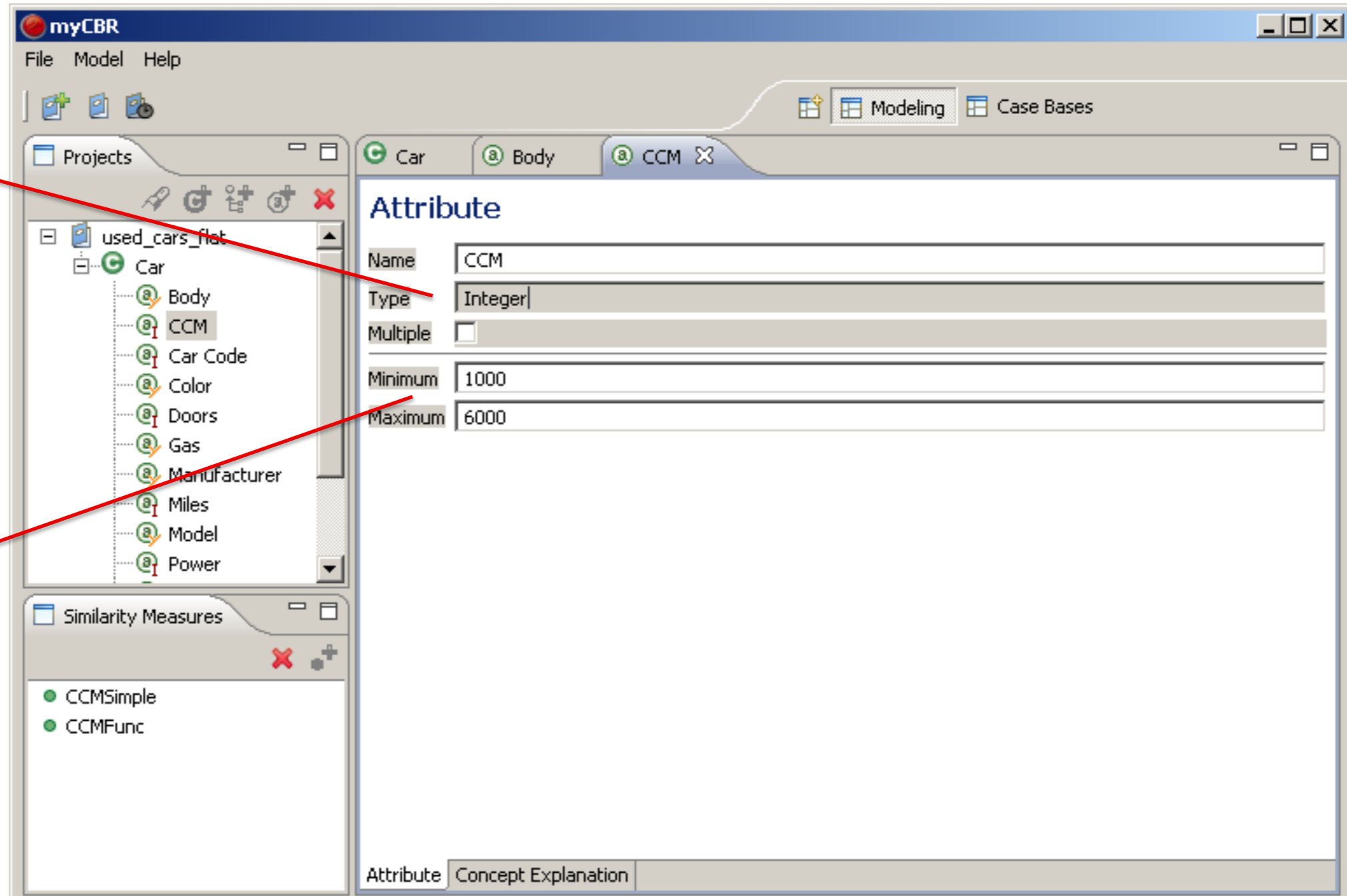
*Editing float similarity measures works analogue*

Name: Enter the name of the attribute

Type: Integer

Multiple: ?

Enter the allowed Minimum and Maximum values for this attribute.



The screenshot shows the myCBR software interface. The 'Attribute' window is open, displaying the following fields:

Attribute	
Name	CCM
Type	Integer
Multiple	<input type="checkbox"/>
Minimum	1000
Maximum	6000

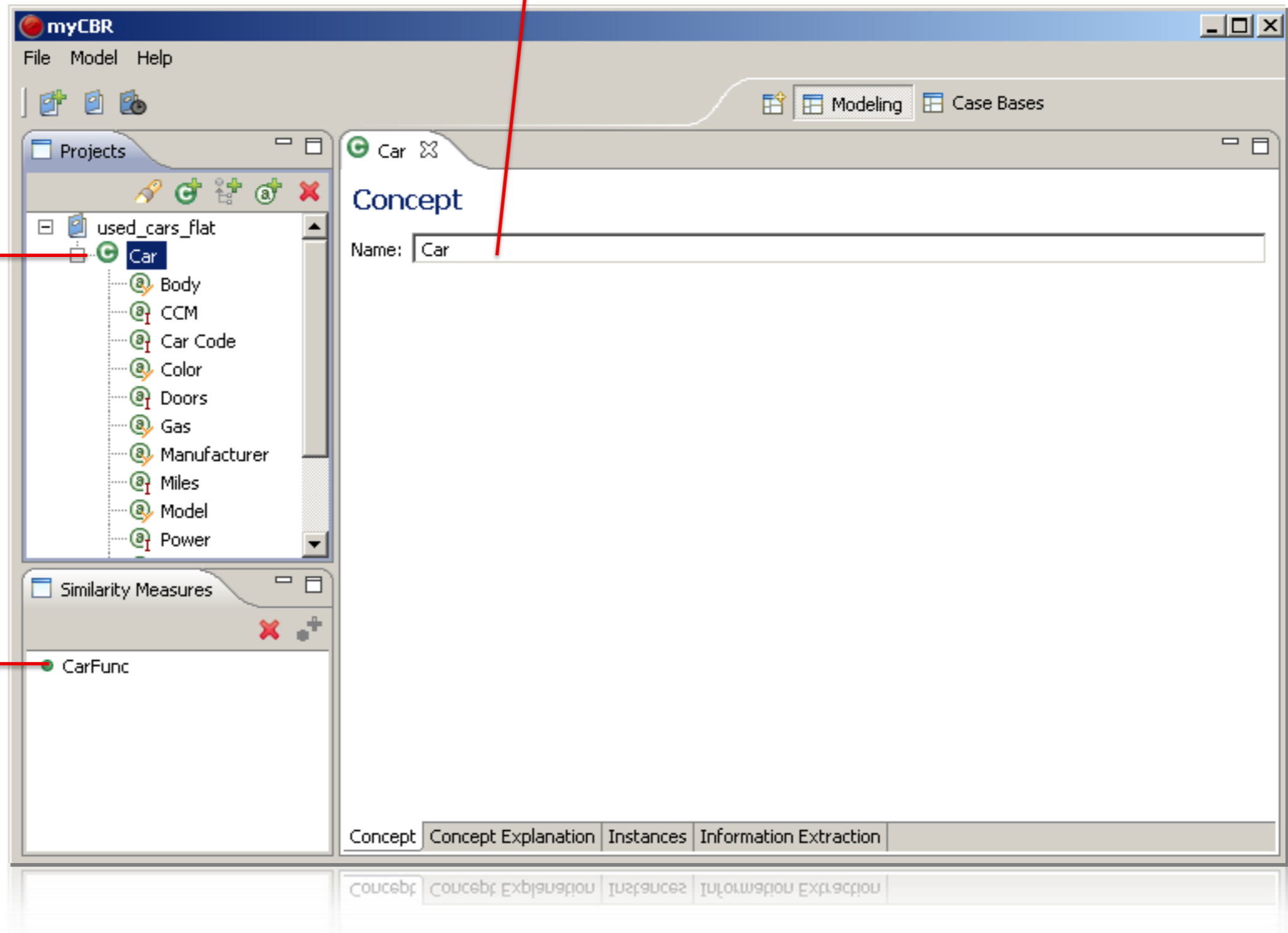
The 'Projects' pane on the left shows a tree structure under 'used\_cars\_flat' with the following attributes: Car, Body, CCM, Car Code, Color, Doors, Gas, Manufacturer, Miles, Model, and Power. The 'Similarity Measures' pane at the bottom left shows two measures: CCMSimple and CCMFunc.

# Concept Information

Double clicking a concept brings up its information's, these are its Name and the SMF used on it

The name of the concept. To rename it simply enter a new name here

The available and used SMF for the concept



The screenshot displays the myCBR software interface. The main window is titled 'myCBR' and has a menu bar with 'File', 'Model', and 'Help'. Below the menu bar are icons for adding, deleting, and saving. The interface is divided into several panes:

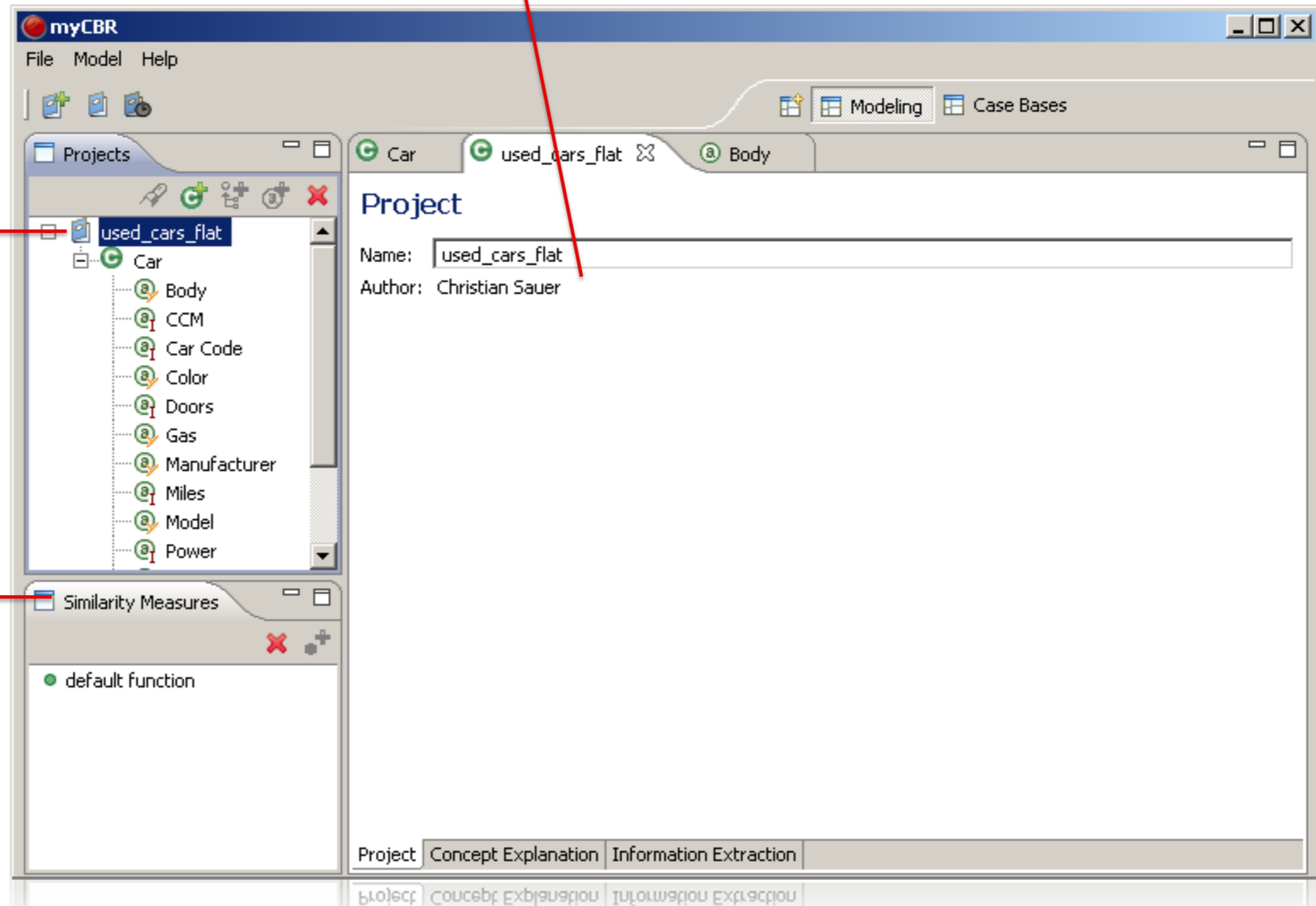
- Projects:** A tree view showing a project named 'used\_cars\_flat' containing a concept named 'Car'. The 'Car' concept has several attributes listed below it: Body, CCM, Car Code, Color, Doors, Gas, Manufacturer, Miles, Model, and Power. Each attribute has a small icon next to it.
- Similarity Measures:** A pane showing a list of similarity measures. One measure, 'CarFunc', is highlighted with a green dot, indicating it is the selected or active measure.
- Concept Window:** A large pane on the right titled 'Concept' with a tab labeled 'Car'. It contains a 'Name:' label followed by a text input field containing the word 'Car'. A red line points from the text box above to this input field.
- Bottom Tabs:** A row of tabs at the bottom of the window, including 'Concept', 'Concept Explanation', 'Instances', and 'Information Extraction'. The 'Concept' tab is currently selected.

# Project Information

Double clicking a project brings up its information's, these are its Name and the SMF used on it

The SMF shown here specifies how special values (like unknown, unspecified) are handled within the project. Please see the next slide on this.

The name of the concept. To rename it simply enter a new name here  
You can also provide Information about the author of the project



# Edit similarity measures: Project, Handling unspecified attributes



Others: If the value/range for an Attribute is not provided by the cbr engine the user can chose "other"

Unknown: If the value/range for an Attribute is not known by the user, he can chose "unknown"

Undefined: If a user doesn't want to specify the value/range for an Attribute he can chose "undefined"

The selected project

The SMF of the project defines the handling of the special values for attribute. These values are:  
\_others\_  
\_unknown\_  
\_undefined\_

The SIM describes how similar these special values are to each other. You can define the way your CBR engine handles unknown data by changing this SIM.

The screenshot shows the myCBR application window. On the left, the 'Projects' pane shows a tree structure with 'used\_cars\_flat' selected, containing attributes like Body, CCM, Car Code, Color, Doors, Gas, Manufacturer, Miles, Model, and Power. Below this, the 'Similarity Measures' pane shows 'default function' selected. The main area displays the 'Car' model with 'default function' selected. A table shows the similarity values for the special values \_others\_, \_unknown\_, and \_undefined\_.

	_others_	_unknown_	_undefined_
_others_	1.0	0.0	0.0
_unknown_	0.0	1.0	0.0
_undefined_	0.0	0.0	1.0

# Retrieval



Select a case base from the available case bases within your project via this dropdown menu

Change: Allows you to select a new symbol from the list of symbols that are defined for the Attribute

Special value: Allows you to select a special value (like "unknown") for the attribute. This is useful to formulate sparse queries.

List view of the cases present in the selected case base

The concept car is selected as the concept to start a retrieval for.

This input form lets you specify concrete values for all attributes. By specifying attribute values you formulate your query to the system.

Start retrieval: Start the retrieval process on the selected case base with the specified query.

This list shows the result set of a retrieval. The overall similarity is shown and also used to sort the retrieved cases by their similarity to the case specified in the retrieval query

**Retrieval**

Case base: CaseBase0

**Query**

Body	sedan	<a href="#">Change</a>	Special Value: <a href="#">none</a>
CCM	1000		Special Value: <a href="#">none</a>
Color	black	<a href="#">Change</a>	Special Value: <a href="#">none</a>
Doors	4		Special Value: <a href="#">none</a>
Gas	gasoline	<a href="#">Change</a>	Special Value: <a href="#">none</a>
Manufacturer	mercedes-benz	<a href="#">Change</a>	Special Value: <a href="#">none</a>
Miles	_unknown_		Special Value: <a href="#">unknown</a>
Model	_unknown_	<a href="#">Change</a>	Special Value: <a href="#">unknown</a>
Power	_unknown_		Special Value: <a href="#">unknown</a>
Price	15000		Special Value: <a href="#">none</a>
Speed	150		Special Value: <a href="#">none</a>
Year	1990		Special Value: <a href="#">none</a>

[Start retrieval](#)

**Similarity Measures**

	424_mercedes-...	765_mercedes-...	549_mercedes-...	734_mercedes-...
Similarity	0.38	0.38	0.36	0.36
Body	sedan	sedan	sedan	sedan
CCM	2300	3200	2200	2200
Car Code	424	765	549	734
Color	black	black	black	black
Doors	4	4	4	4
Gas	gasoline	gasoline	diesel	diesel
Manufacturer	mercedes-benz	mercedes-benz	mercedes-benz	mercedes-benz
Miles	18726	89768	18684	42550
Model	c_230_kompres...	e_320	e_220_diesel	c_220_diesel
Power	192	224	95	95

**List view of the cases present in the selected case base**

424_mercedes-benz - 0.38
765_mercedes-benz - 0.38
549_mercedes-benz - 0.36
734_mercedes-benz - 0.36
529_mercedes-benz - 0.36
876_mercedes-benz - 0.36
143_mercedes-benz - 0.36
704_mercedes-benz - 0.35
770_mercedes-benz - 0.35
684_mercedes-benz - 0.35
63_mercedes-benz - 0.35
853_mercedes-benz - 0.35
142_mercedes-benz - 0.34
310_mercedes-benz - 0.33
747_mercedes-benz - 0.32
674_audi - 0.3
181_audi - 0.3
556_audi - 0.3
564_audi - 0.3
455_audi - 0.3
742_audi - 0.27
936_audi - 0.27
206_bmw - 0.27
297_mercedes-benz - 0.26
779_mercedes-benz - 0.26

# Initial Case Bases perspective view (Project selected)

The Case Bases perspective allows you to add/remove and modify the data within your cases and subsequently case base(s) after modelling your domain knowledge in the Modelling perspective

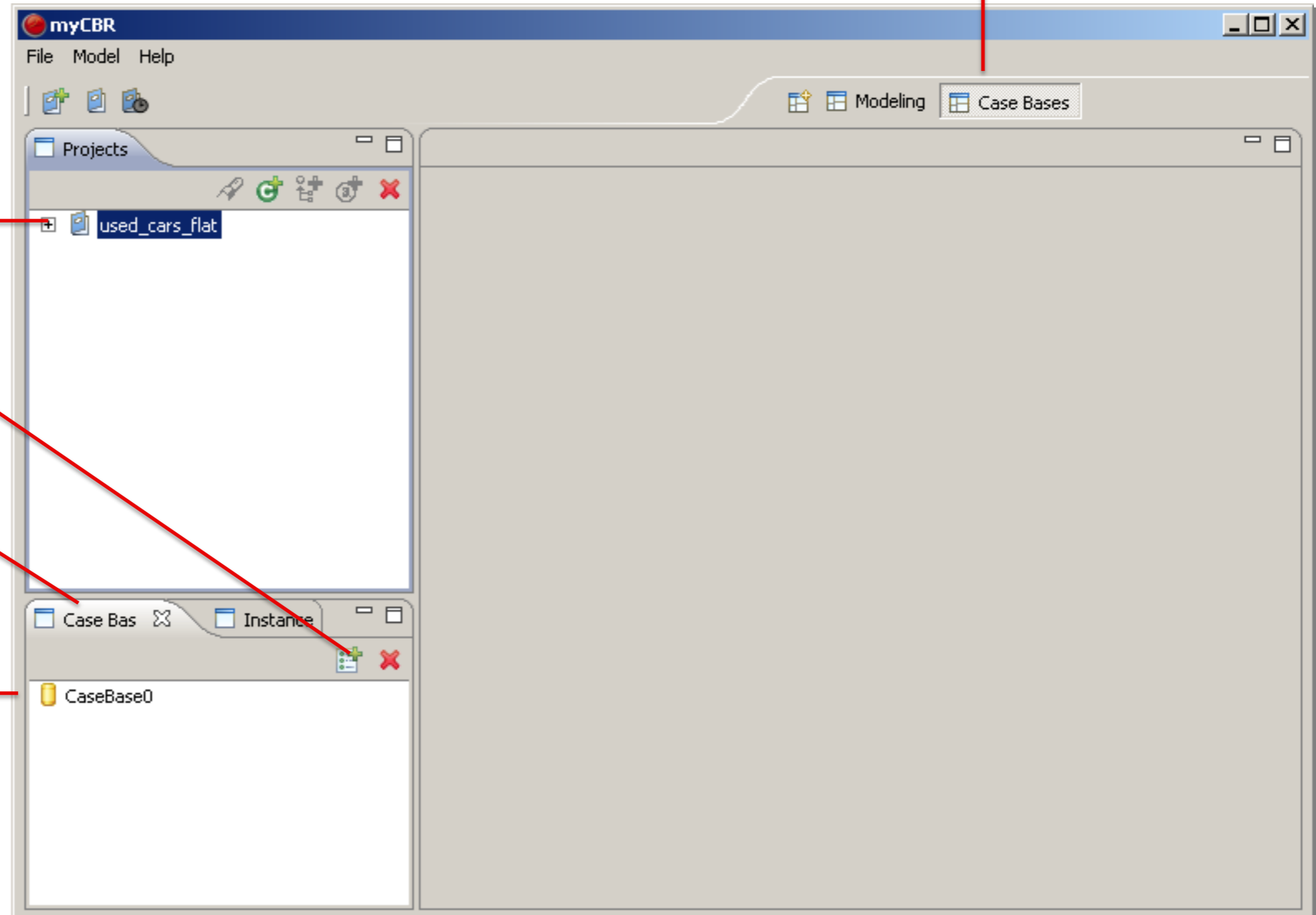
You have to select a project to work on its case base(s)

Add and delete case bases

The Case Base Tab is selected: This Tab lets you add and delete case bases.

The case base(s) used within the project are listed here, if a project is selected.

The Case base perspective Tab is selected.



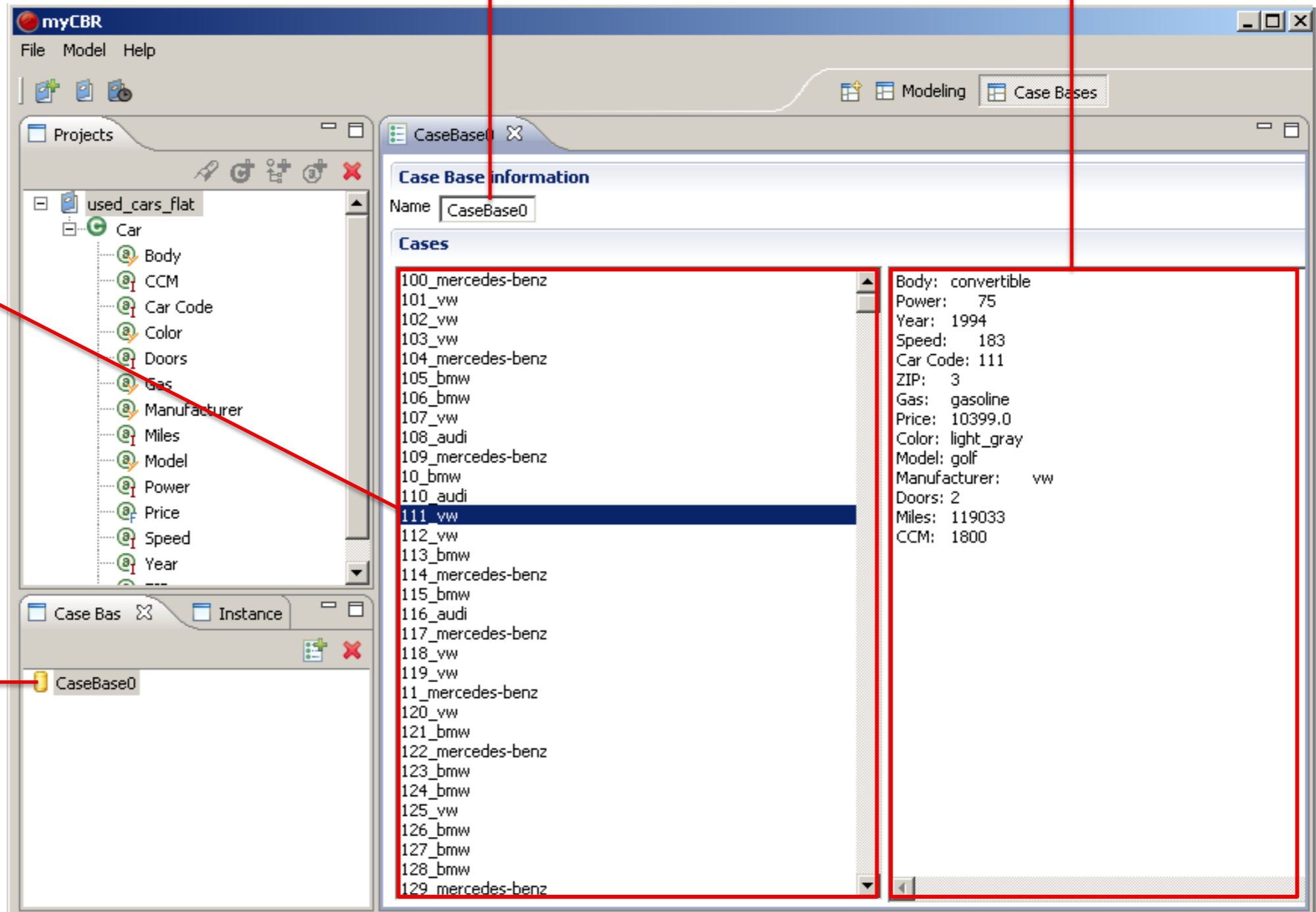
# Case base browsing, Case viewing

Display of the selected case base's name

The case view displays the individual data (values of the Attributes) of a selected case

Clicking a case in the case list brings up the individual data of this case in the case view

Double clicking a case base brings up its Cases in the case list.



The screenshot shows the myCBR software interface with the following components:

- Projects Panel:** Displays a tree structure under 'used\_cars\_flat' with attributes: Car, Body, CCM, Car Code, Color, Doors, Gas, Manufacturer, Miles, Model, Power, Price, Speed, and Year.
- Case Base Information Panel:** Shows the selected case base name 'CaseBase0'.
- Cases Panel:** A list of cases, with '111\_vw' selected. The list includes cases like '100\_mercedes-benz', '101\_vw', '102\_vw', '103\_vw', '104\_mercedes-benz', '105\_bmw', '106\_bmw', '107\_vw', '108\_audi', '109\_mercedes-benz', '110\_audi', '111\_vw', '112\_vw', '113\_bmw', '114\_mercedes-benz', '115\_bmw', '116\_audi', '117\_mercedes-benz', '118\_vw', '119\_vw', '120\_vw', '121\_bmw', '122\_mercedes-benz', '123\_bmw', '124\_bmw', '125\_vw', '126\_bmw', '127\_bmw', '128\_bmw', and '129\_mercedes-benz'.
- Case View Panel:** Displays the individual data for the selected case '111\_vw':
  - Body: convertible
  - Power: 75
  - Year: 1994
  - Speed: 183
  - Car Code: 111
  - ZIP: 3
  - Gas: gasoline
  - Price: 10399.0
  - Color: light\_gray
  - Model: golf
  - Manufacturer: vw
  - Doors: 2
  - Miles: 119033
  - CCM: 1800

# Instance View, Concept selected and an Instance double-clicked



Overview of the selected Instance of a concept (selected case)

Within this detailed view you can enter/change the data (attribute values) of a case

The Concept car is selected, which allows to add/remove and edit Instances (cases) of this concept.

Delete the selected Instance

Add a new Instance

The Instance tab is selected: This Tab lets you add and delete case(s) / Instances of the selected concept.

The selected case from the list of cases (Instances) in the currently selected case base, available for the selected concept (car) is shown in the Instance view.

**myCBR**

File Model Help

Projects

- used\_cars\_flat
  - Car
    - Body
    - CCM
    - Car Code
    - Color
    - Doors
    - Gas
    - Manufacturer
    - Miles
    - Model
    - Power
    - Price
    - Speed
    - Year

Case Bas

- 100\_mercedes-benz
- 101\_vw
- 102\_vw
- 103\_vw
- 104\_mercedes-benz
- 105\_bmw
- 106\_bmw
- 107\_vw
- 108\_audi

Instance

102\_vw

**Instance information**

Name 102\_vw

**Attributes**

Body	station_wagon	<a href="#">Change</a>	Special Value: <a href="#">none</a>
CCM	1800		Special Value: <a href="#">none</a>
Car Code	102		Special Value: <a href="#">none</a>
Color	yellow	<a href="#">Change</a>	Special Value: <a href="#">none</a>
Doors	5		Special Value: <a href="#">none</a>
Gas	gasoline	<a href="#">Change</a>	Special Value: <a href="#">none</a>
Manufacturer	vw	<a href="#">Change</a>	Special Value: <a href="#">none</a>
Miles	59802		Special Value: <a href="#">none</a>
Model	golf	<a href="#">Change</a>	Special Value: <a href="#">none</a>
Power	190		Special Value: <a href="#">none</a>
Price	28599.0		Special Value: <a href="#">none</a>
Speed	183		Special Value: <a href="#">none</a>
Year	1996		Special Value: <a href="#">none</a>
ZIP	1		Special Value: <a href="#">none</a>

Concept Concept Explanation

# Adding a new Instance: Entering the attribute values

For every attribute not of the data type symbol you can click into the field to directly enter a value for the attribute, for example an integer number.

A new tab is opened when a new instance is created, displaying the ID of the new instance.

For all attributes of the data type "symbol" by clicking "Change" a symbol can be assigned as the value of the attribute.

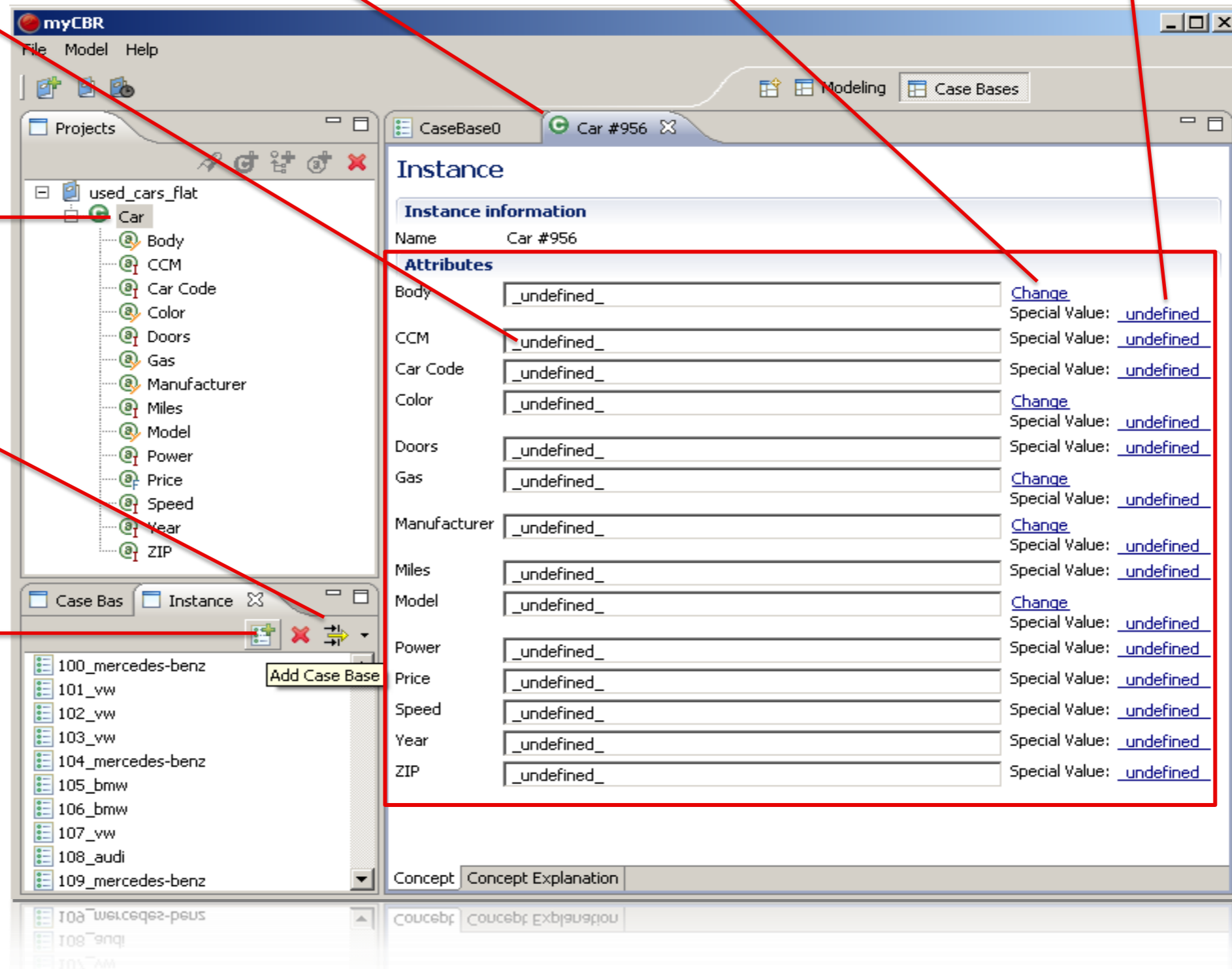
"undefined" is the default value for all attribute values. By clicking here you can change the value to a different special value: "unknown".

The Concept car is selected, which allows to add/remove and edit Instances (cases) of this concept.

Filter integration is not yet enabled

Click here to create a new Instance

Make sure to save your Project after adding new Instances to your case base(s)



**Instance**

**Instance information**

Name Car #956

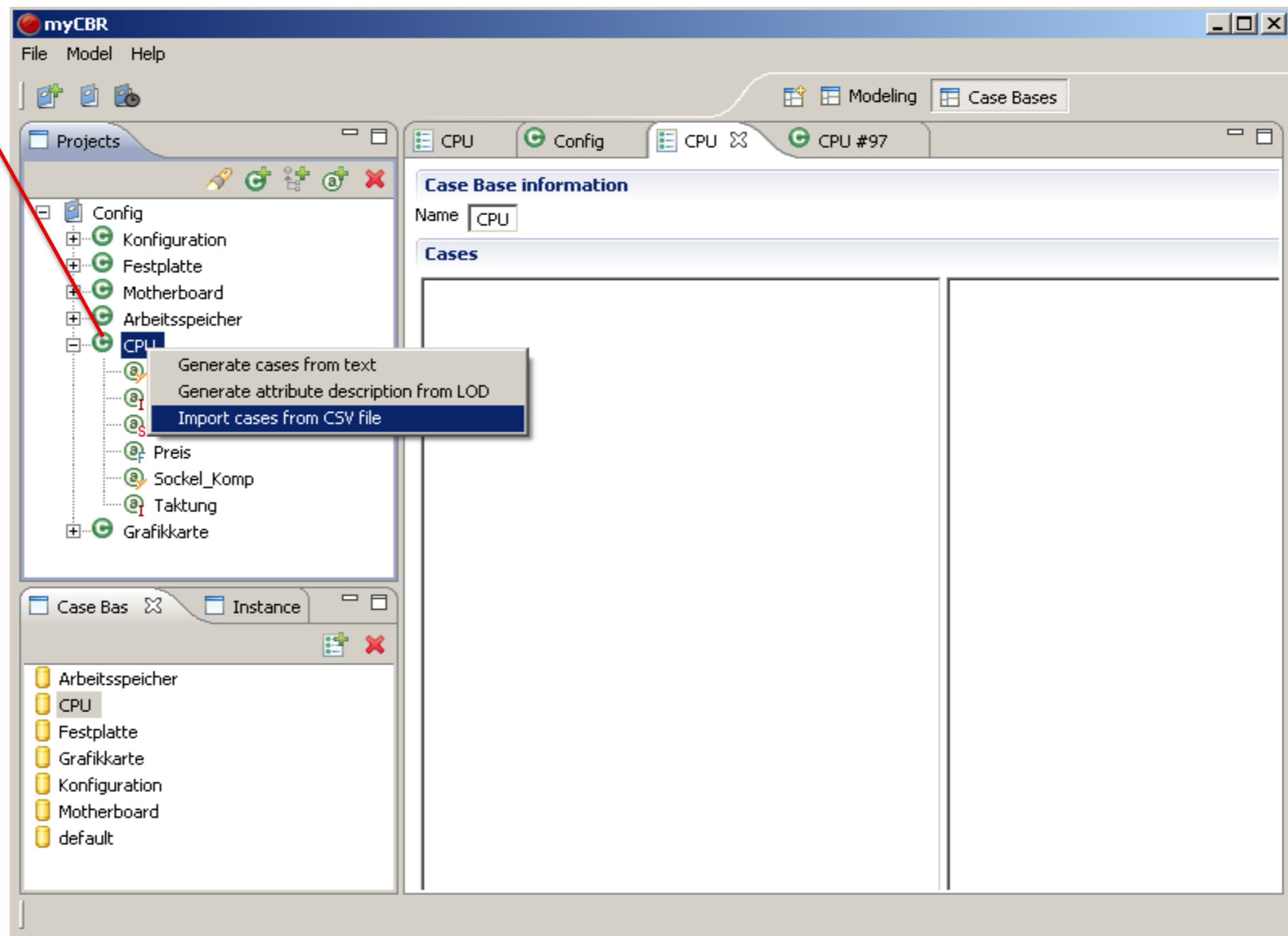
**Attributes**

Attribute	Value	Change	Special Value
Body	_undefined_	<a href="#">Change</a>	Special Value: <a href="#">undefined</a>
CCM	_undefined_	<a href="#">Change</a>	Special Value: <a href="#">undefined</a>
Car Code	_undefined_	<a href="#">Change</a>	Special Value: <a href="#">undefined</a>
Color	_undefined_	<a href="#">Change</a>	Special Value: <a href="#">undefined</a>
Doors	_undefined_	<a href="#">Change</a>	Special Value: <a href="#">undefined</a>
Gas	_undefined_	<a href="#">Change</a>	Special Value: <a href="#">undefined</a>
Manufacturer	_undefined_	<a href="#">Change</a>	Special Value: <a href="#">undefined</a>
Miles	_undefined_	<a href="#">Change</a>	Special Value: <a href="#">undefined</a>
Model	_undefined_	<a href="#">Change</a>	Special Value: <a href="#">undefined</a>
Power	_undefined_	<a href="#">Change</a>	Special Value: <a href="#">undefined</a>
Price	_undefined_	<a href="#">Change</a>	Special Value: <a href="#">undefined</a>
Speed	_undefined_	<a href="#">Change</a>	Special Value: <a href="#">undefined</a>
Year	_undefined_	<a href="#">Change</a>	Special Value: <a href="#">undefined</a>
ZIP	_undefined_	<a href="#">Change</a>	Special Value: <a href="#">undefined</a>

Concept Concept Explanation

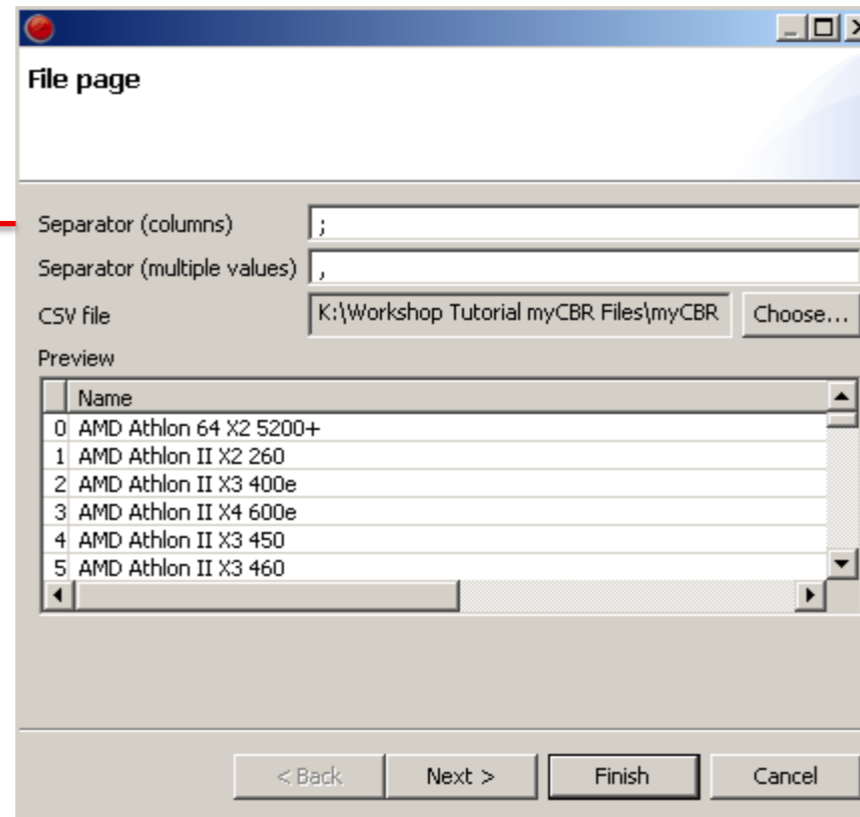
# Adding new Instances: Importing from CSV files

To import cases for a concept (Instances of the concept) right click the concept and select "Import cases from CSV file" in the context menu



# Adding new Instances: Importing from CSV files

In the upcoming dialogue you can specify the csv file to load and the separators that are used within the specified csv file. The dialogue will show you a preview of the data found in the csv file you specified using the separators you specified. To import this data click on "Finish".



The dialog box is titled "File page". It contains the following fields and controls:

- Separator (columns): A text box containing ";"
- Separator (multiple values): A text box containing ","
- CSV File: A text box containing "K:\Workshop Tutorial myCBR Files\myCBR" and a "Choose..." button.
- Preview: A table with a header "Name" and five rows of data.
- Buttons: "< Back", "Next >", "Finish", and "Cancel".

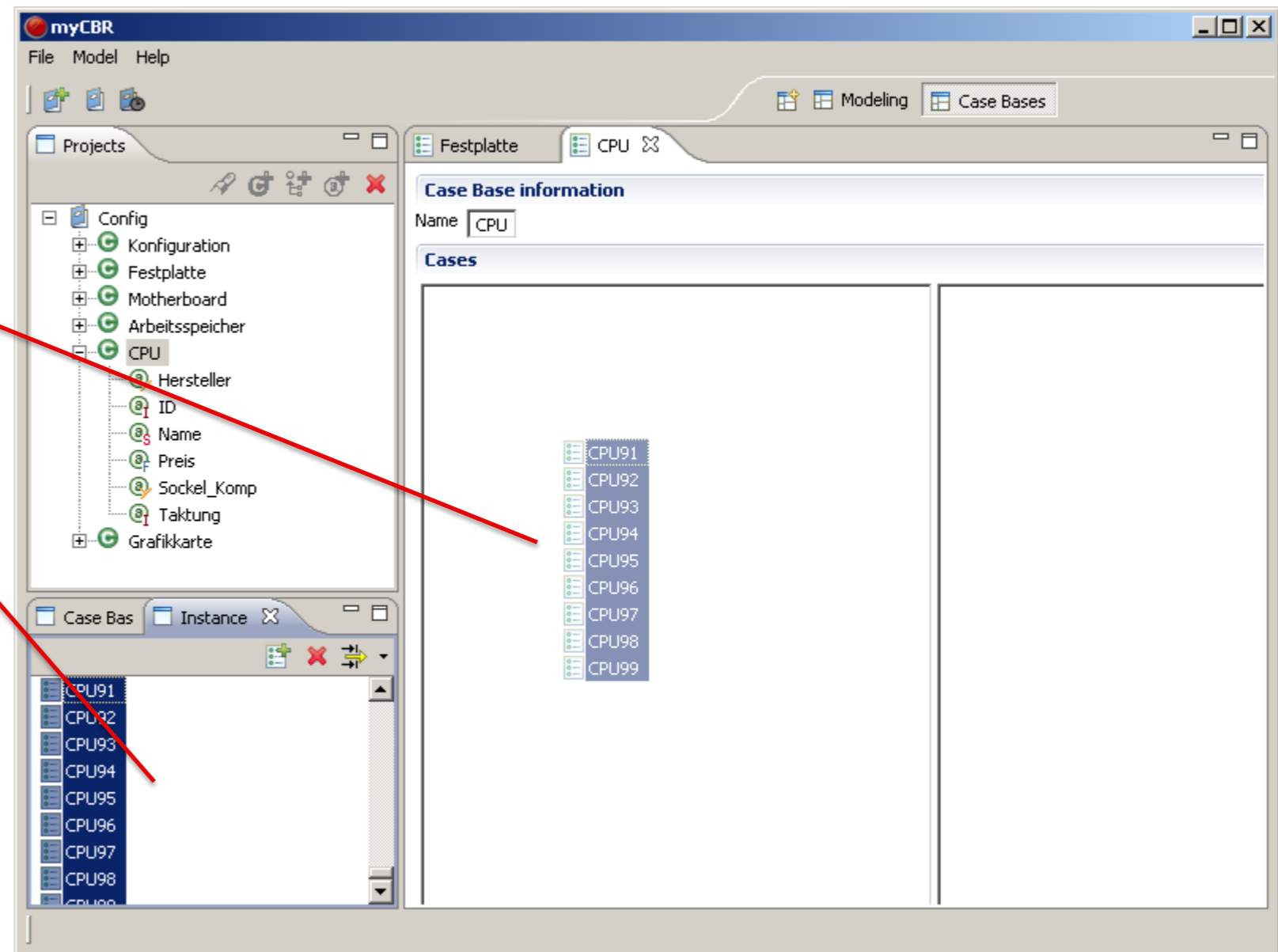
	Name
0	AMD Athlon 64 X2 5200+
1	AMD Athlon II X2 260
2	AMD Athlon II X3 400e
3	AMD Athlon II X4 600e
4	AMD Athlon II X3 450
5	AMD Athlon II X3 460

# Adding the new Instances to a case base

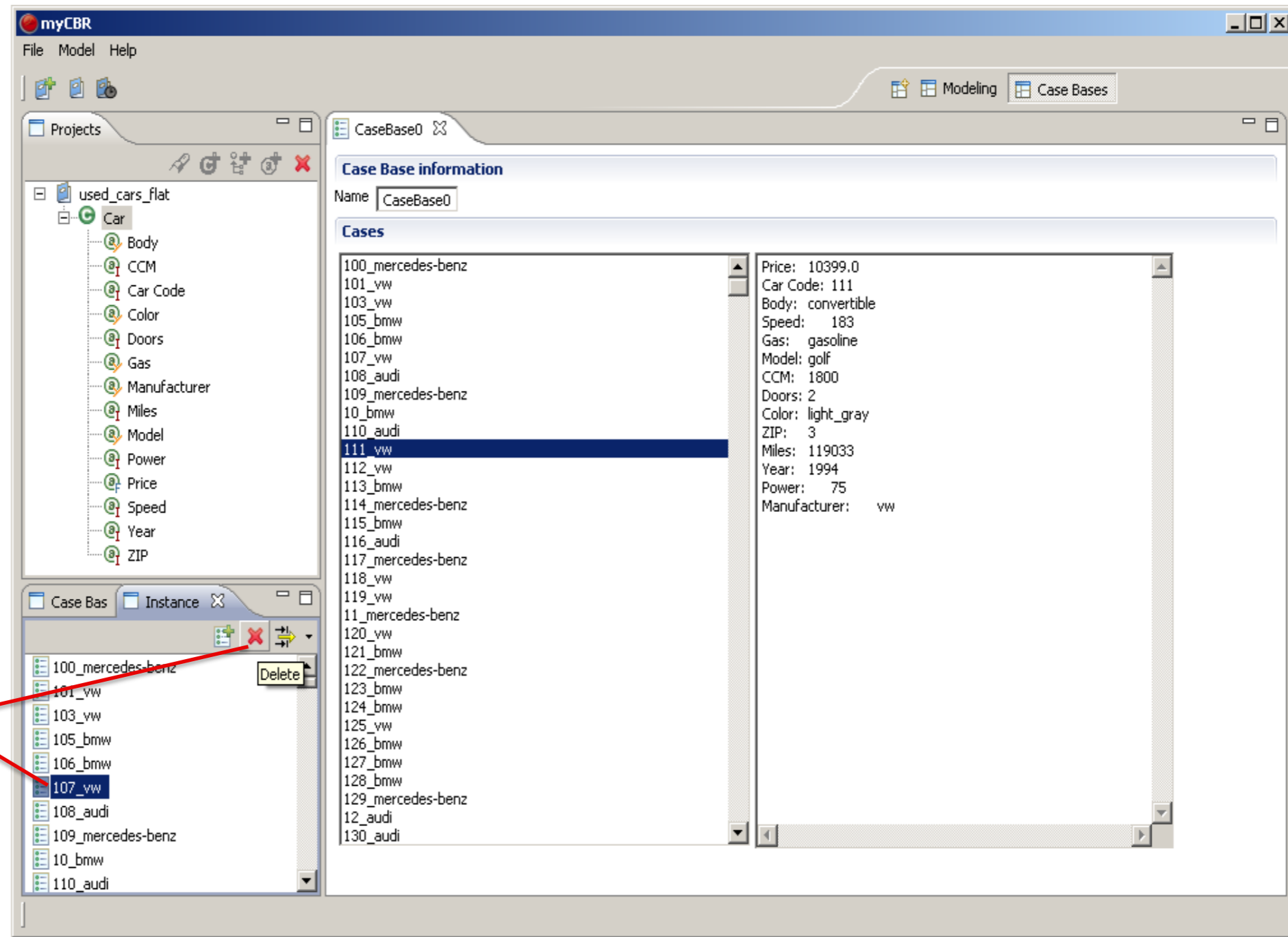
The imported instances can now be added to a (selected) case base by dragging and dropping them into the case base's "Cases" field.

You always drag and drop your new instances into the case base, regardless if imported or generated or entered.

Make sure to save your Project after adding new Instances to your case base(s)



# Deleting an Instances from the case base



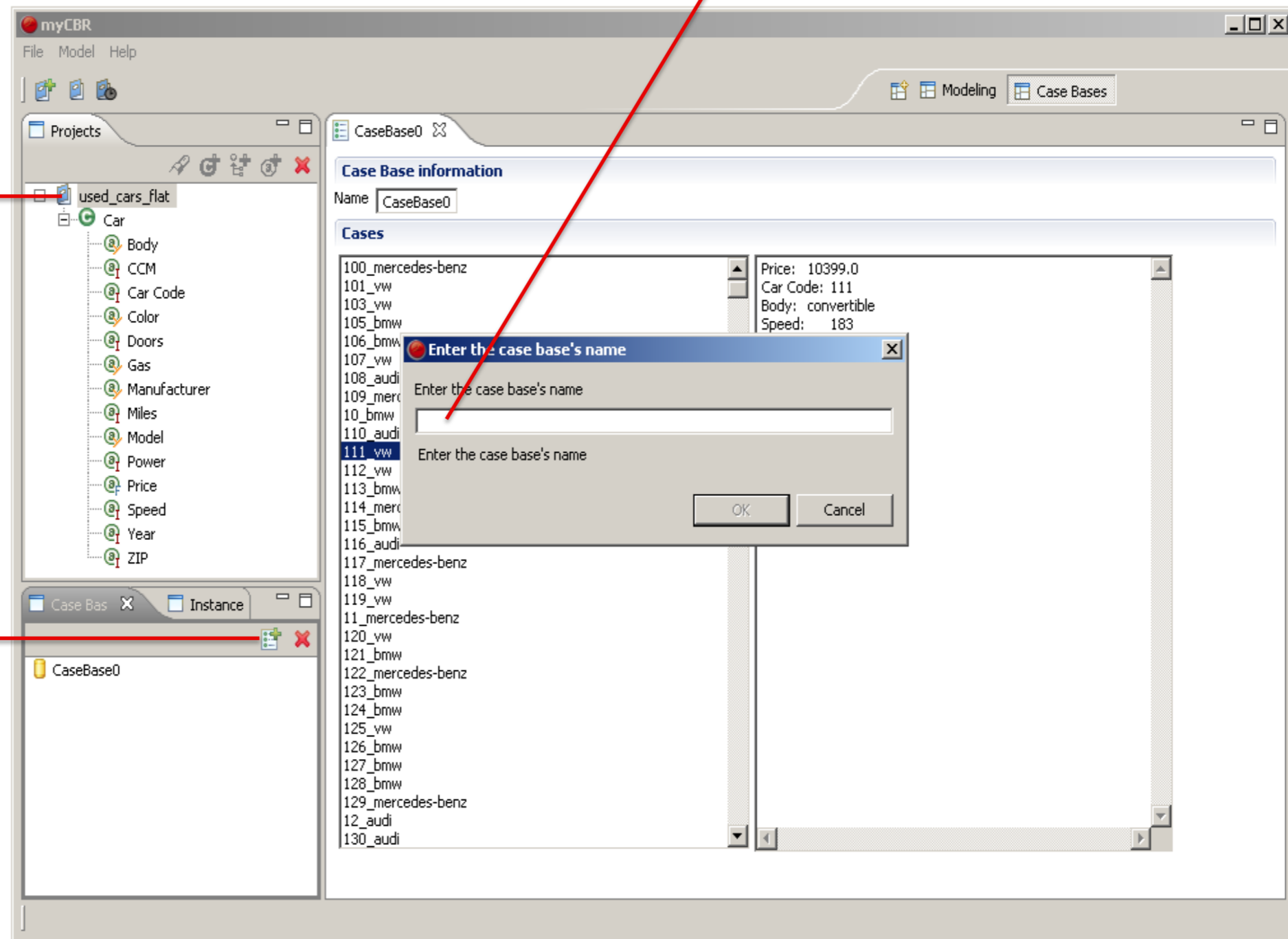
Select the Instance(s) to be deleted and then click on the Delete button (To view the deletion update the case base view (re open it) manually for now.

# Adding a new case base to a project

Select the project you want to add a case base to

Click here to create a new case base for the currently selected project

Name the new case base and press "Ok"

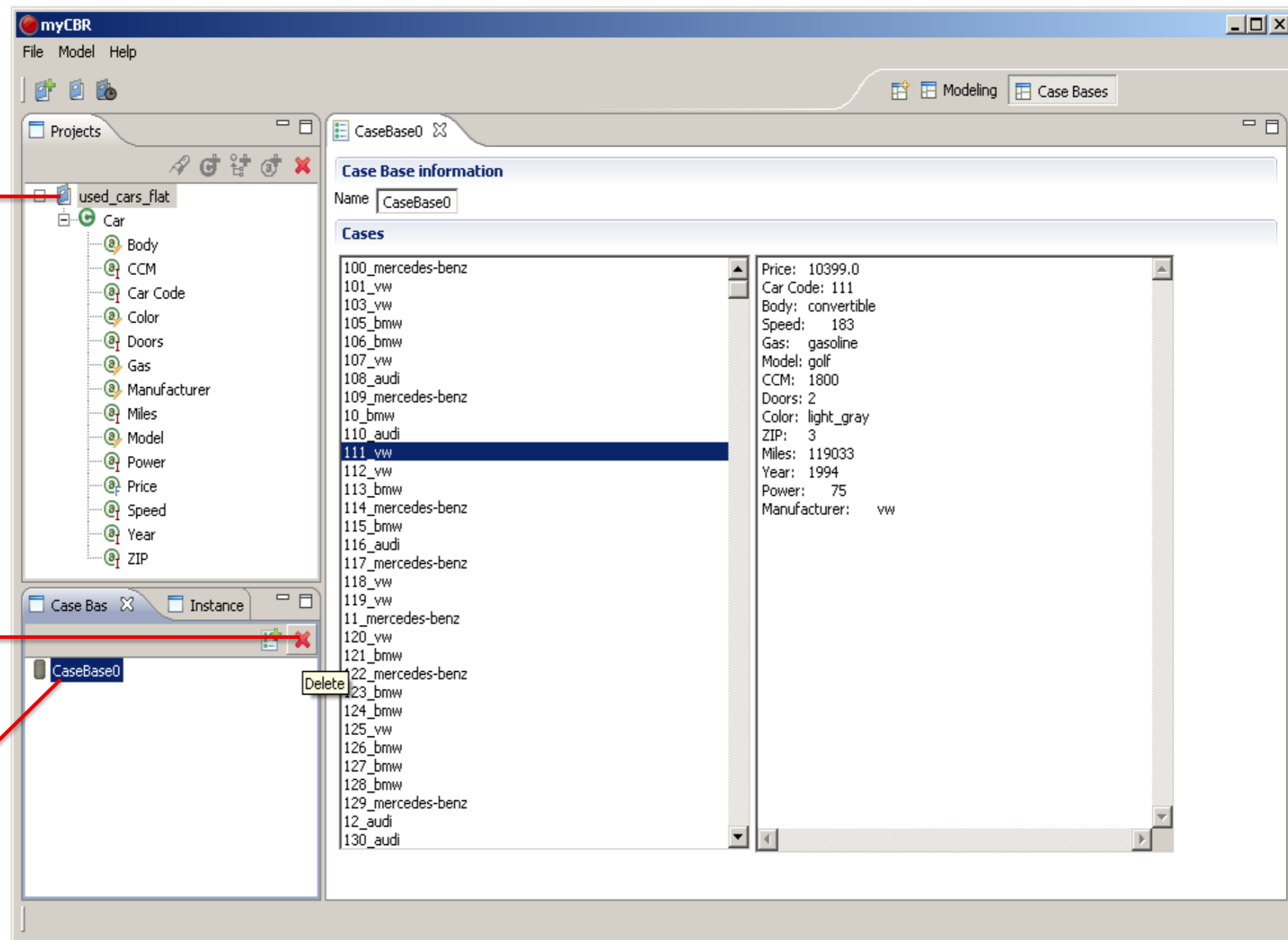


# Deleting a case base from a project

Select the project you want to delete a case base from

Click here to delete the selected case base

Select the case base you want to delete

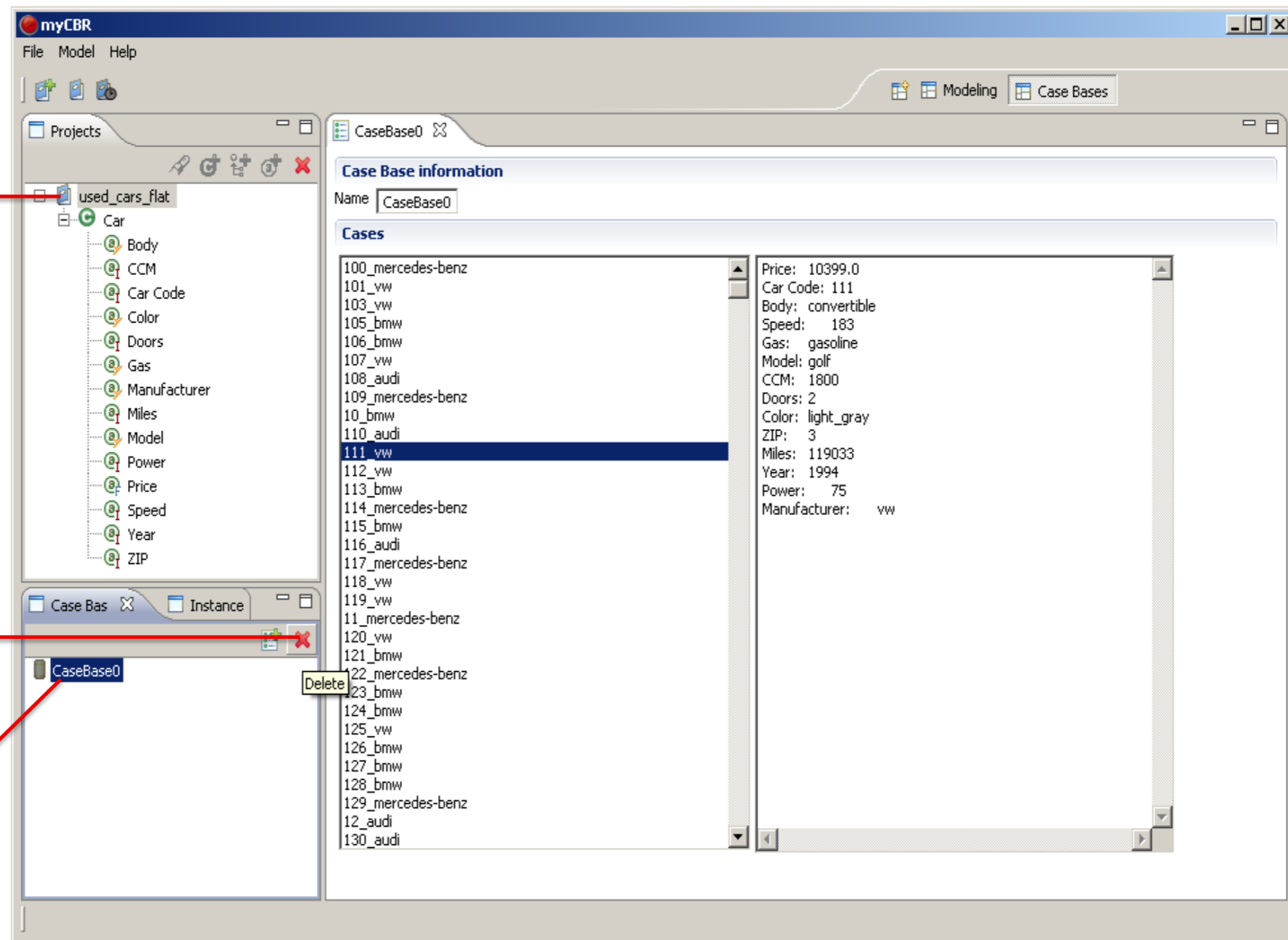


# Deleting a case base from a project

Select the project you want to delete a case base from

Click here to delete the selected case base

Select the case base you want to delete

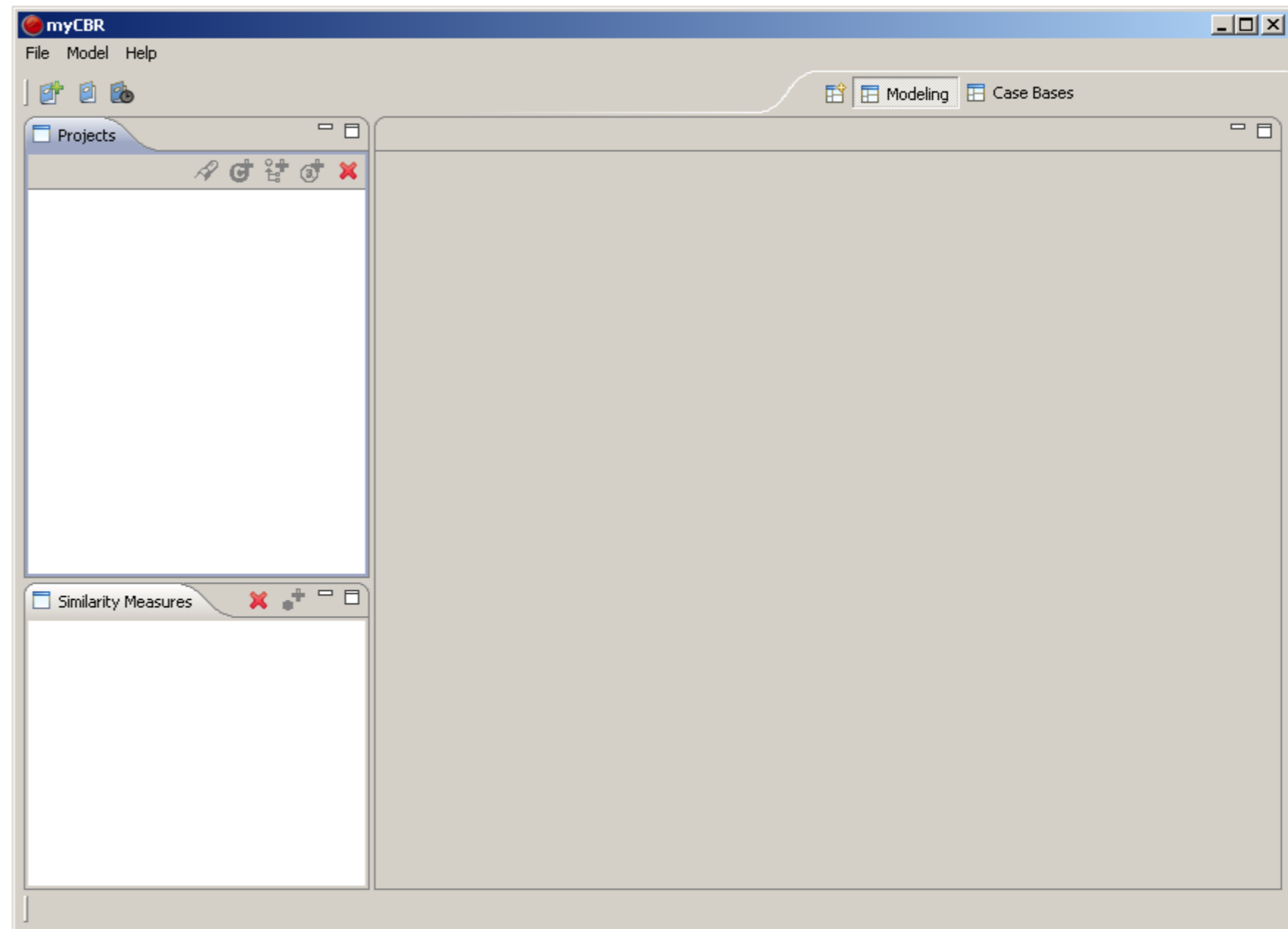


# myCBR Getting started: Modeling your domain



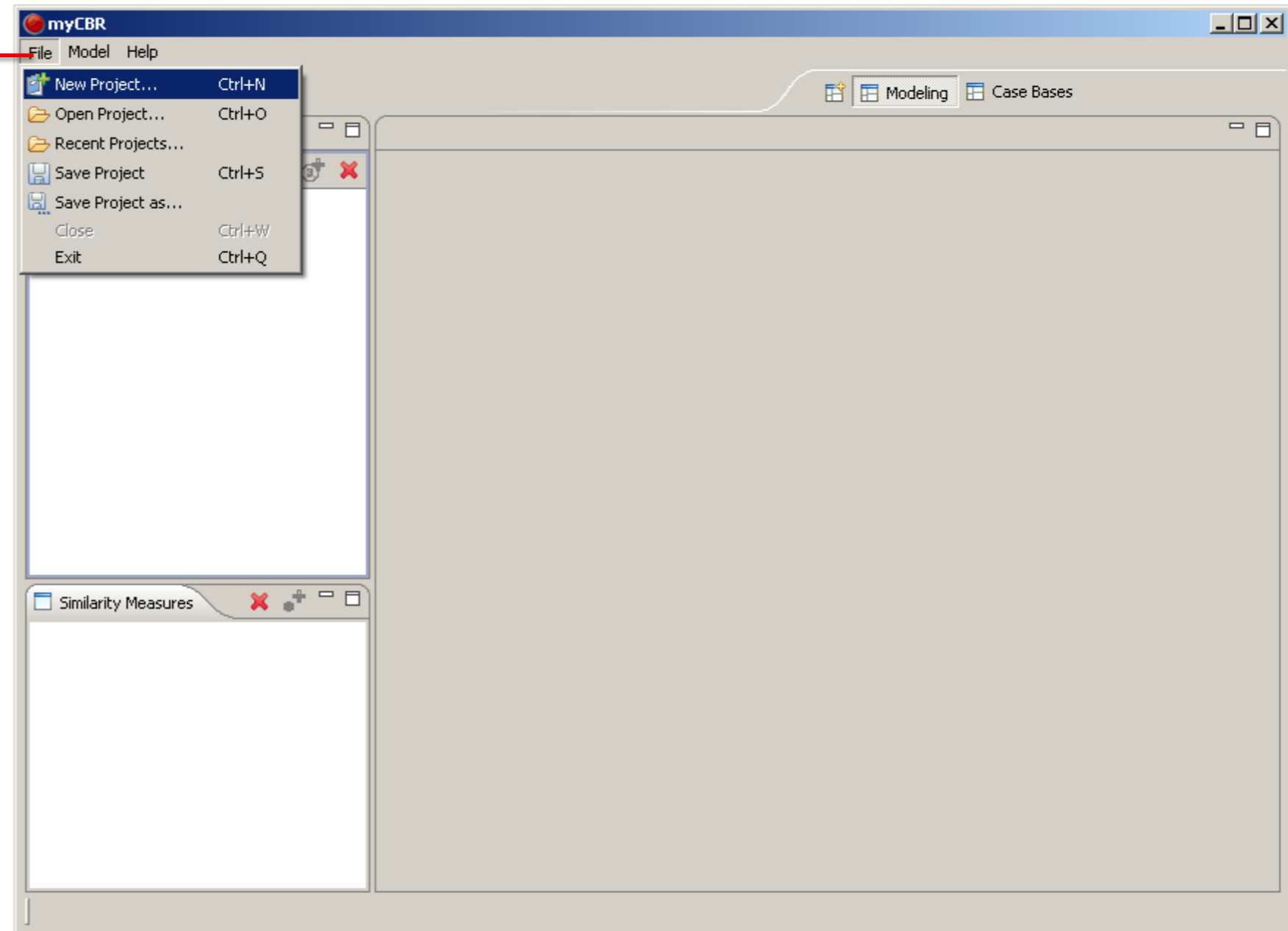
# Start up myCBR

Starting up myCBR will bring up the UI as seen above. From this point on you can either start a new project or load an existing project.



# Create a new project

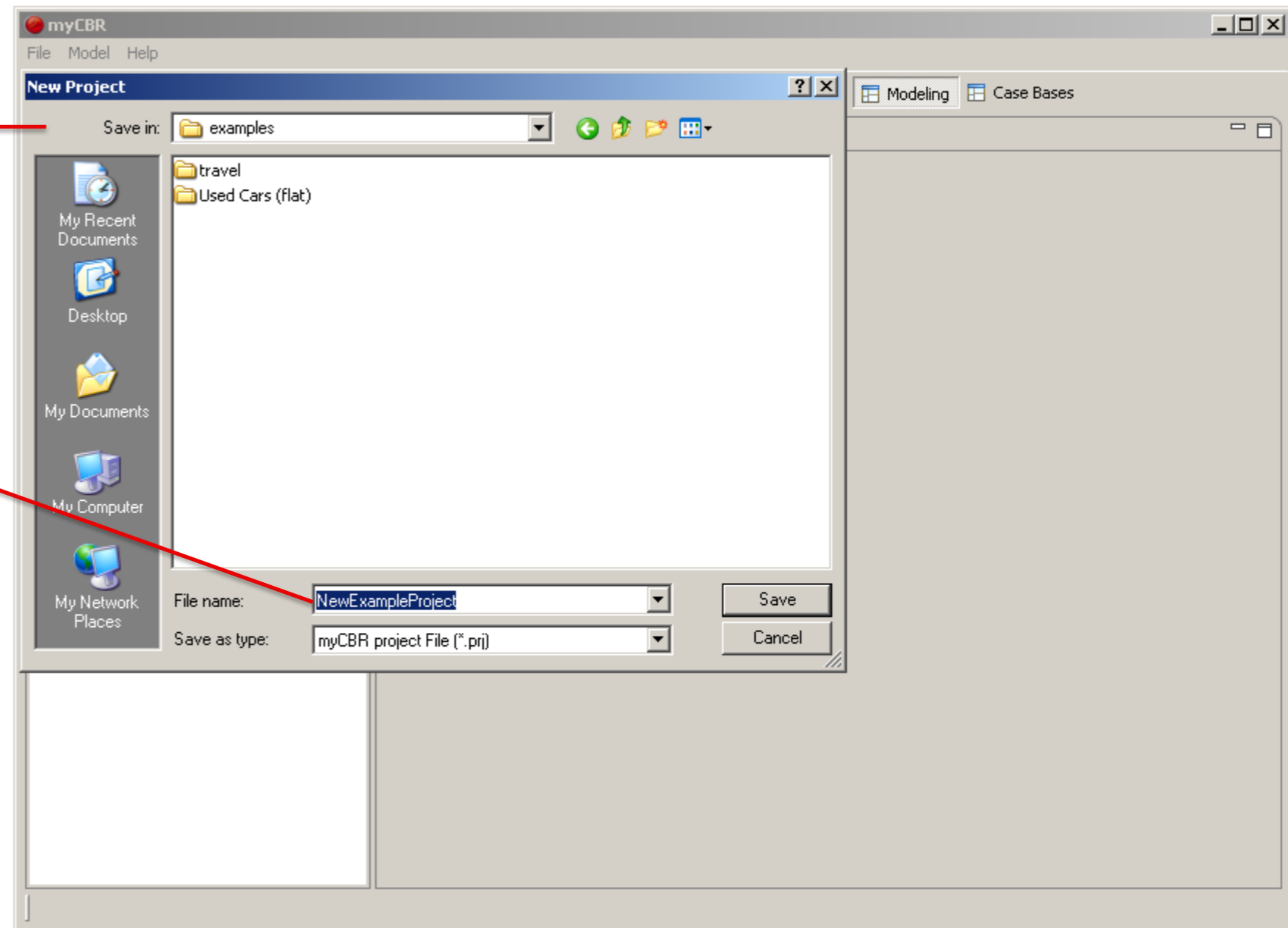
Selecting "New Project" from the "File" context menu for now will start our new example project.



# Create a new project

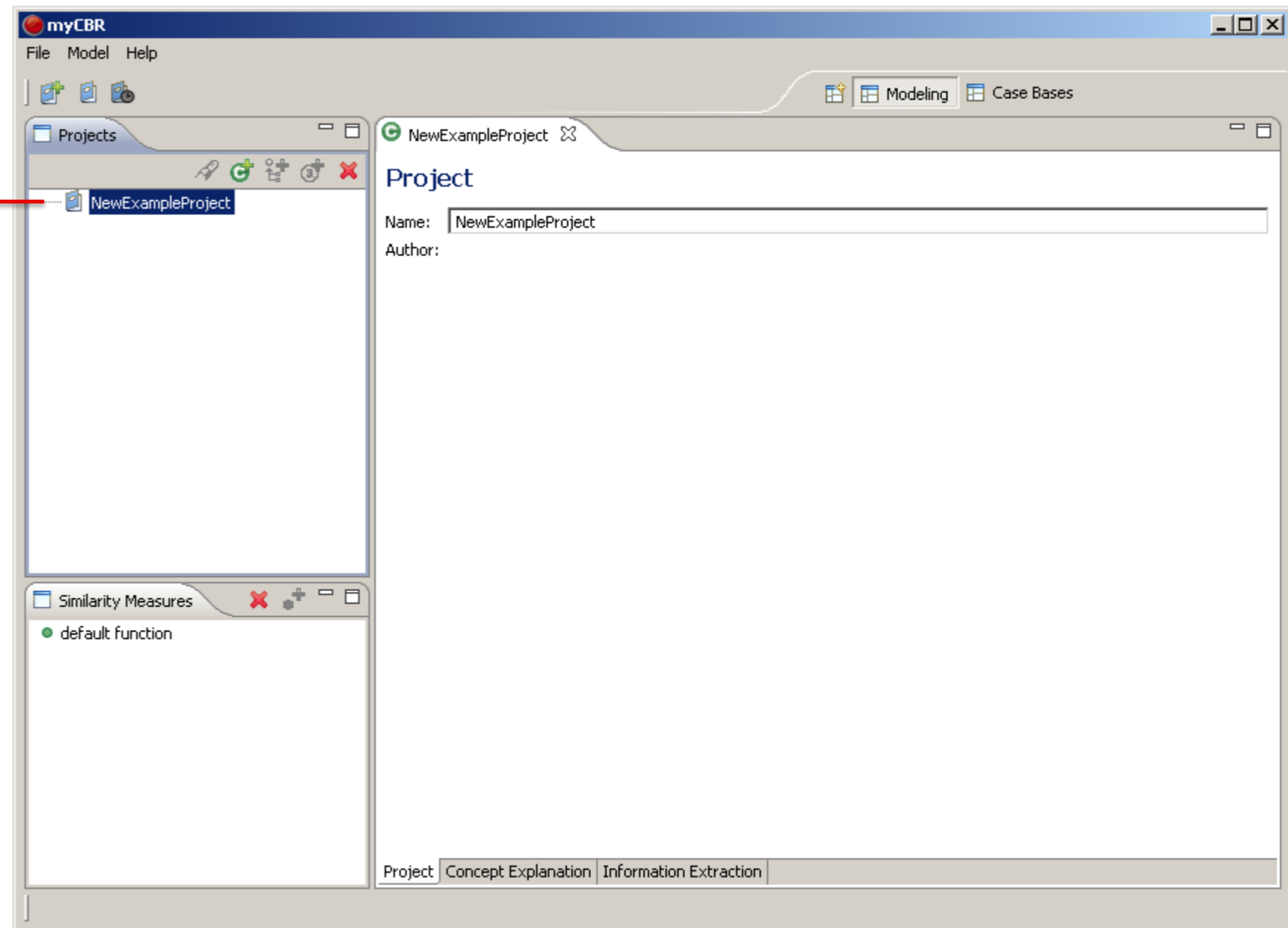
In the “New Project” dialog, specify the name of the new project and where to store it, then press “Save”

Our new project for this example will be named “NewExampleProject”



# Create a new project

After creating and double-clicking the new project myCBR should present you with a view like this.



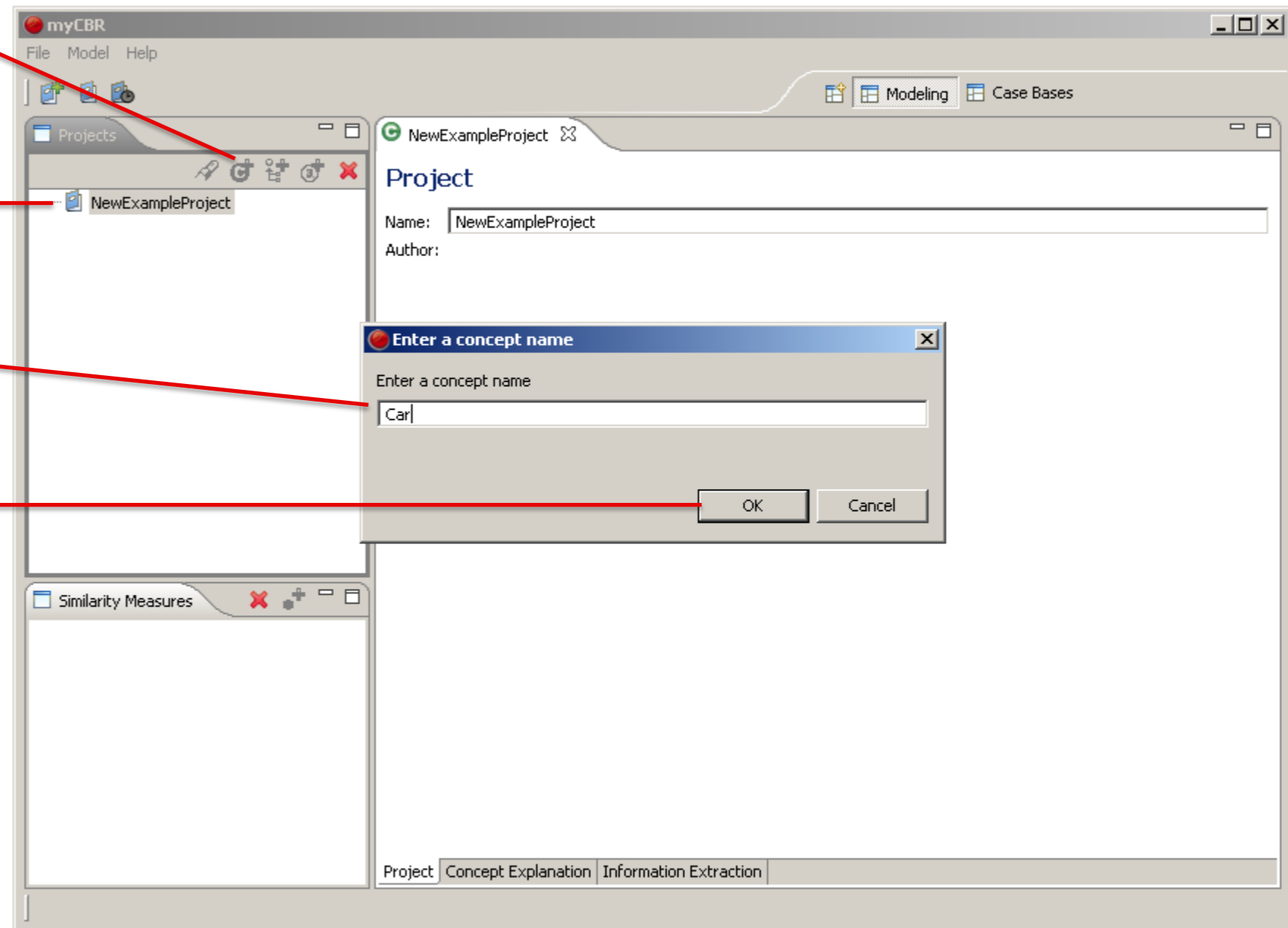
# Add a concept to the project

2) Click on "Add a concept"

1) Select the project to add a concept to

3) In the dialog, enter the concepts name

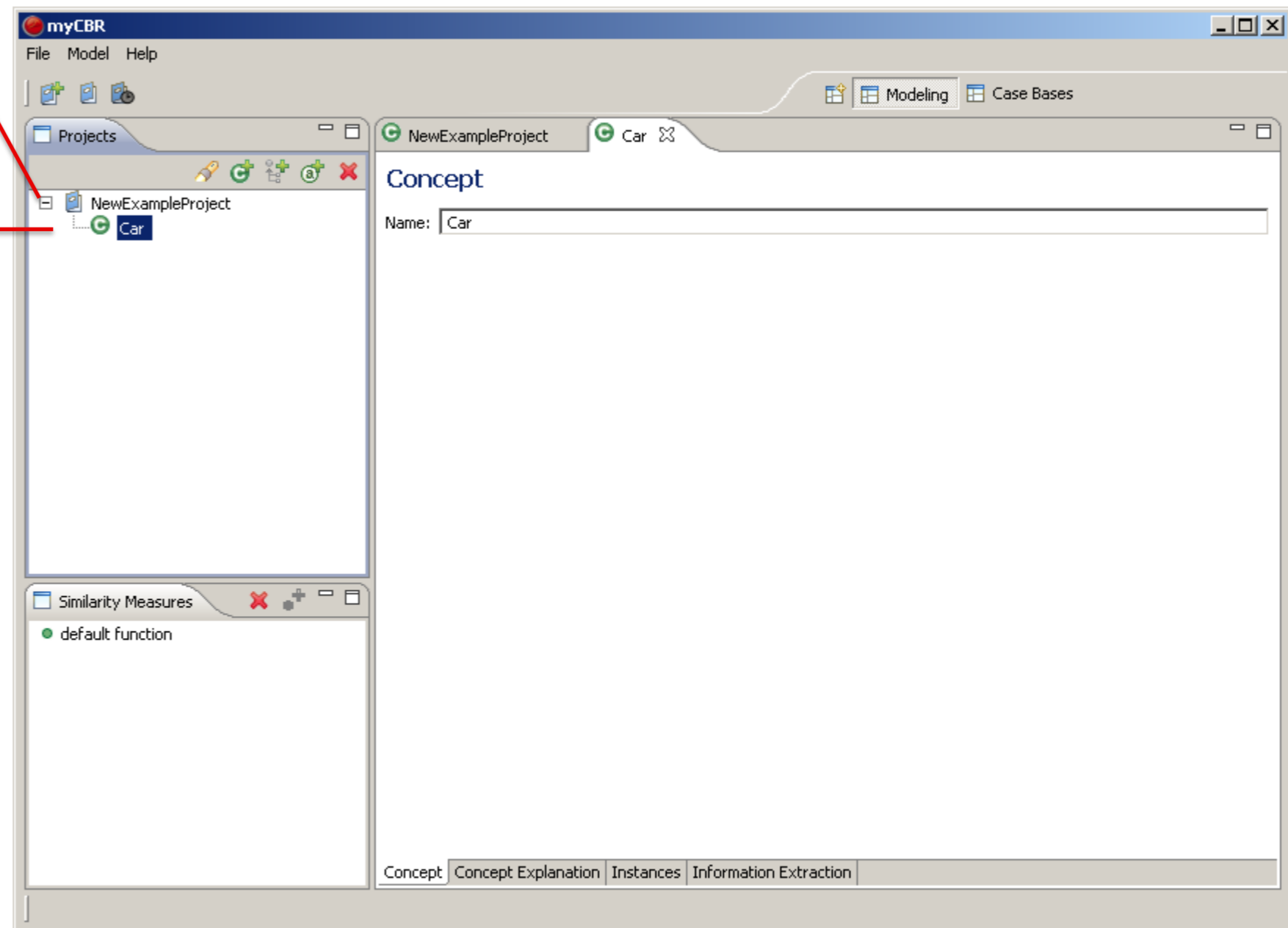
4) Confirm the creation of the concept by clicking OK



# Add a concept to the project

1) Double-clicking your project expands the concept tree which now should be populated by the concept "Car".

2) Double-clicking the concept "Car" should open the basic concept view as shown in this screenshot.



# Add an attribute to the concept "Car"

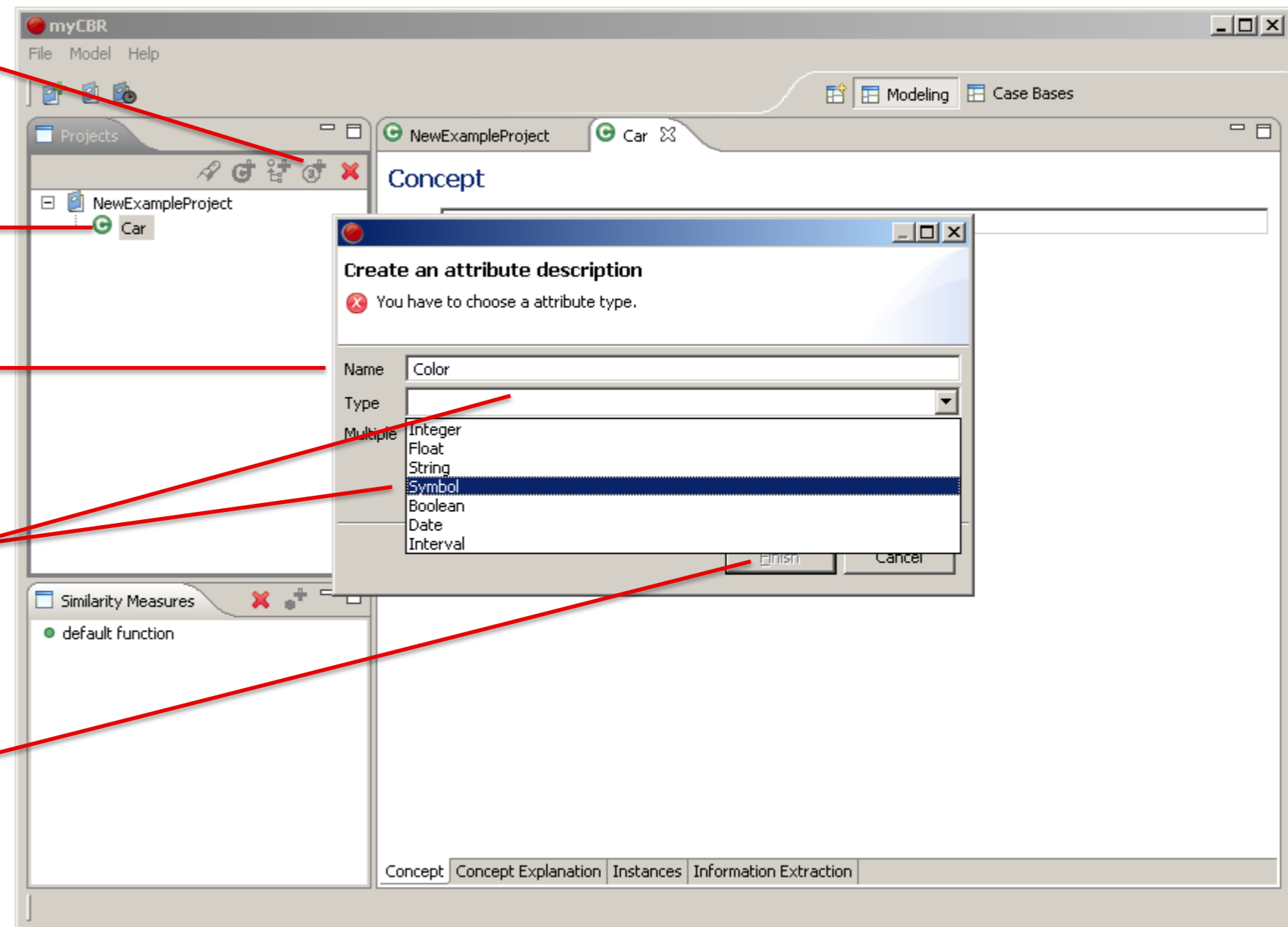
2) Click on "Add new attribute"

1) Select the concept you want to add an attribute to. In our example we selected the attribute "Car".

3) In the dialog, enter the attributes name

4) Also in the dialog select the attributes data type by clicking on the drop down menu "Type" and selecting the data type you want for the attribute, in our example this is the data type "Symbol".

5) Confirm the creation of the attribute by clicking on the "Finish" button.



# Entering values for an attribute of the symbol data type

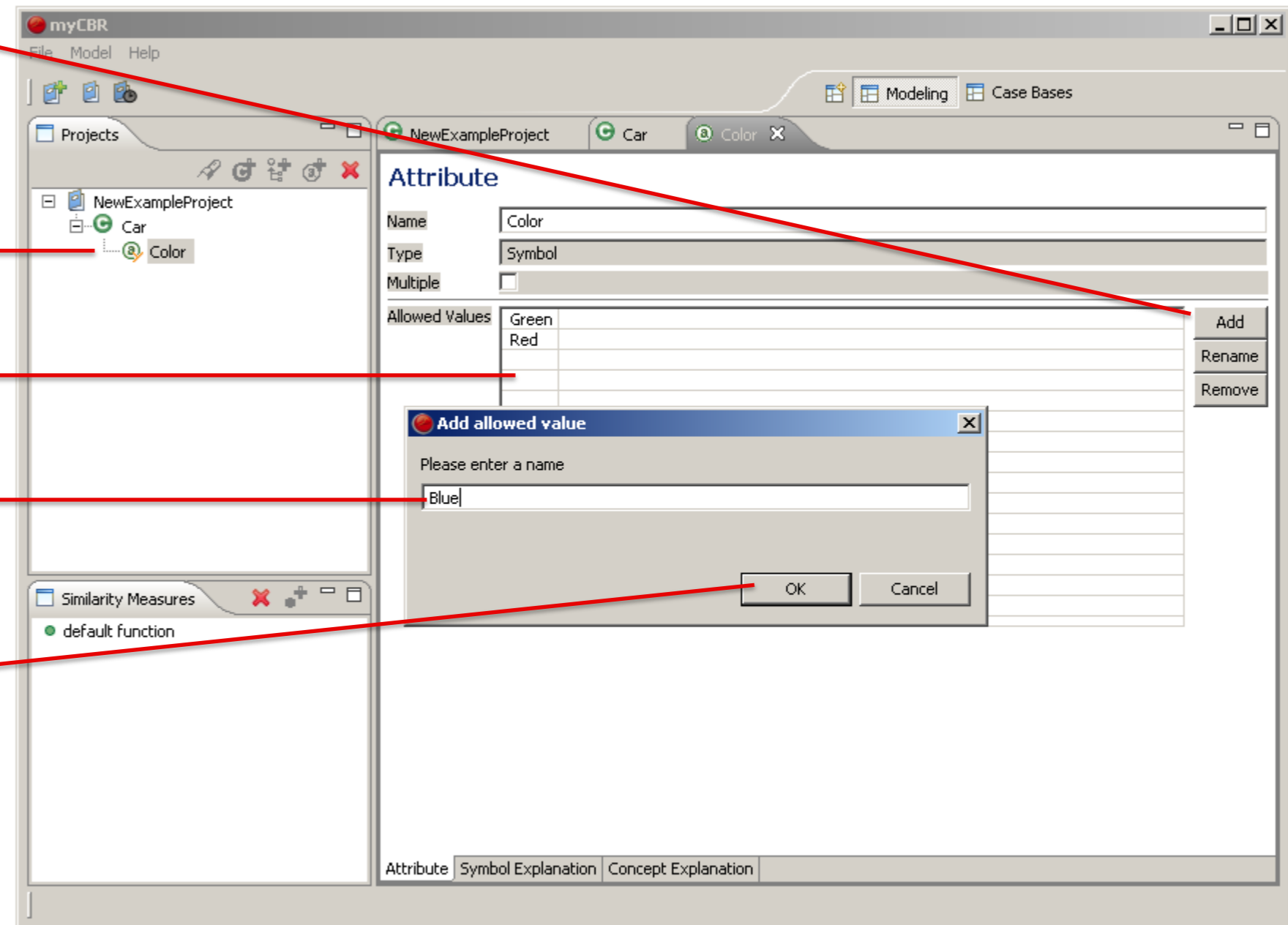
2) Click on "Add" (a value)

1) Select the attribute you want to enter values for.

Values already entered are listed here.

3) Enter the value, in our example: Enter a string describing a colour.

4) Confirm the value by clicking "OK". The dialog will reappear to allow you to enter further values. To stop entering values just click on "Cancel".

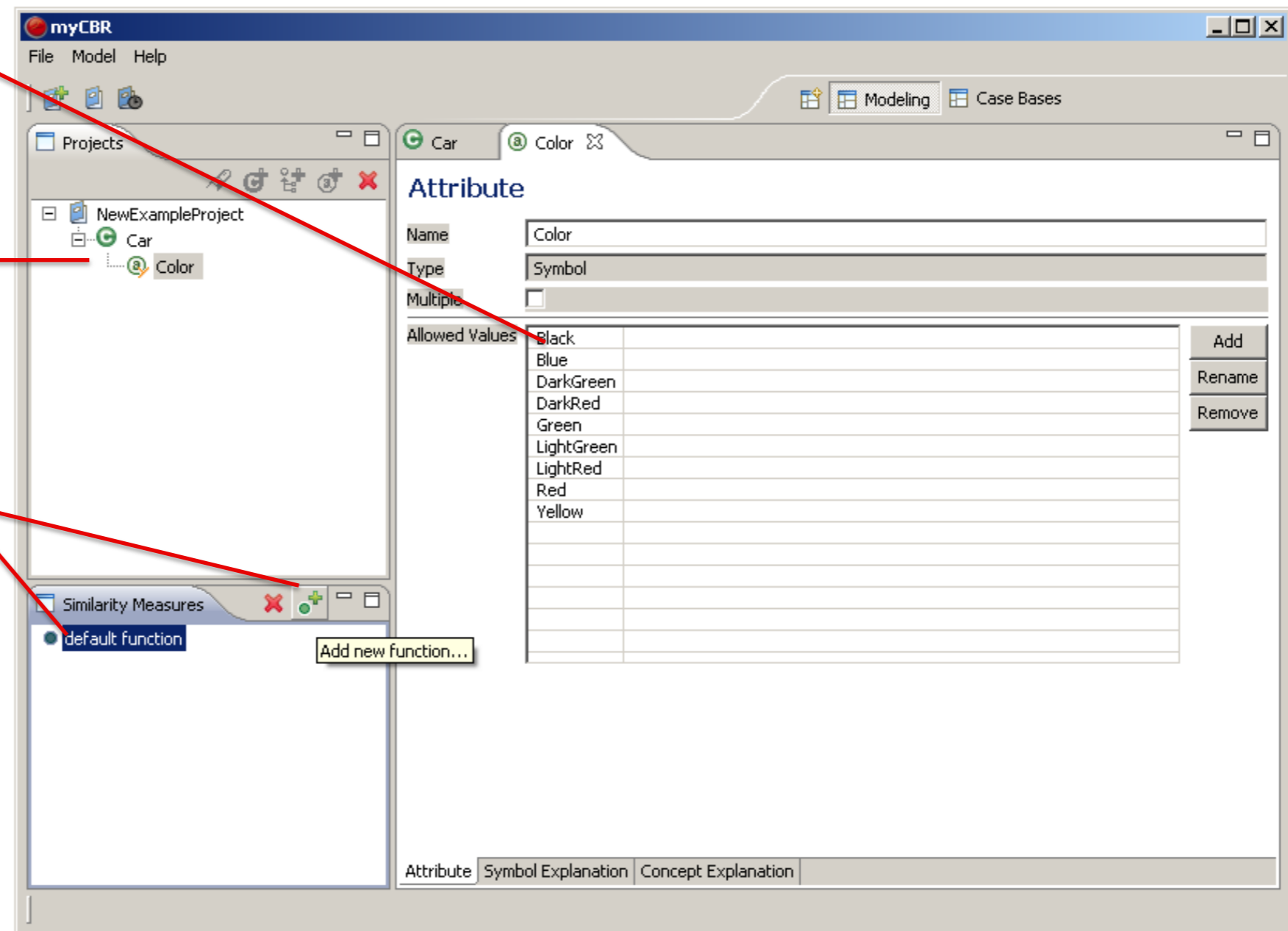


# Modeling a similarity measure for a symbol attribute

1) Finish entering all the values you want the attribute to have

2) Select the attribute (if not already working on the attribute and thus having it selected anyway).

3) Either double-click the default function to edit it or rather click on "Add new function" to add a new similarity measure (function).

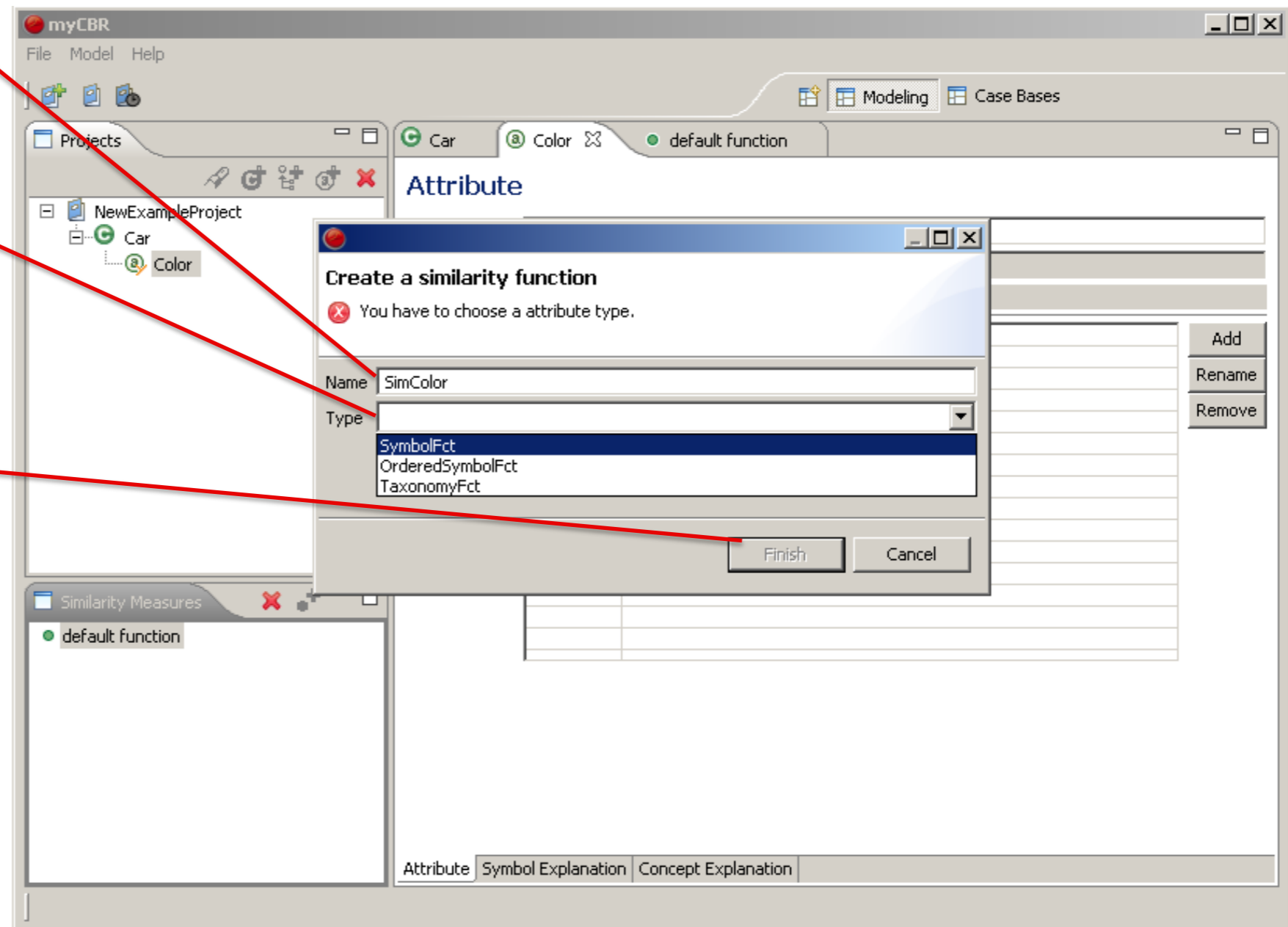


# Modeling a similarity measure for a symbol attribute

1) Enter a name for the new function

2) Select the kind of the new function

3) Confirm your entries by clicking on "Finish"

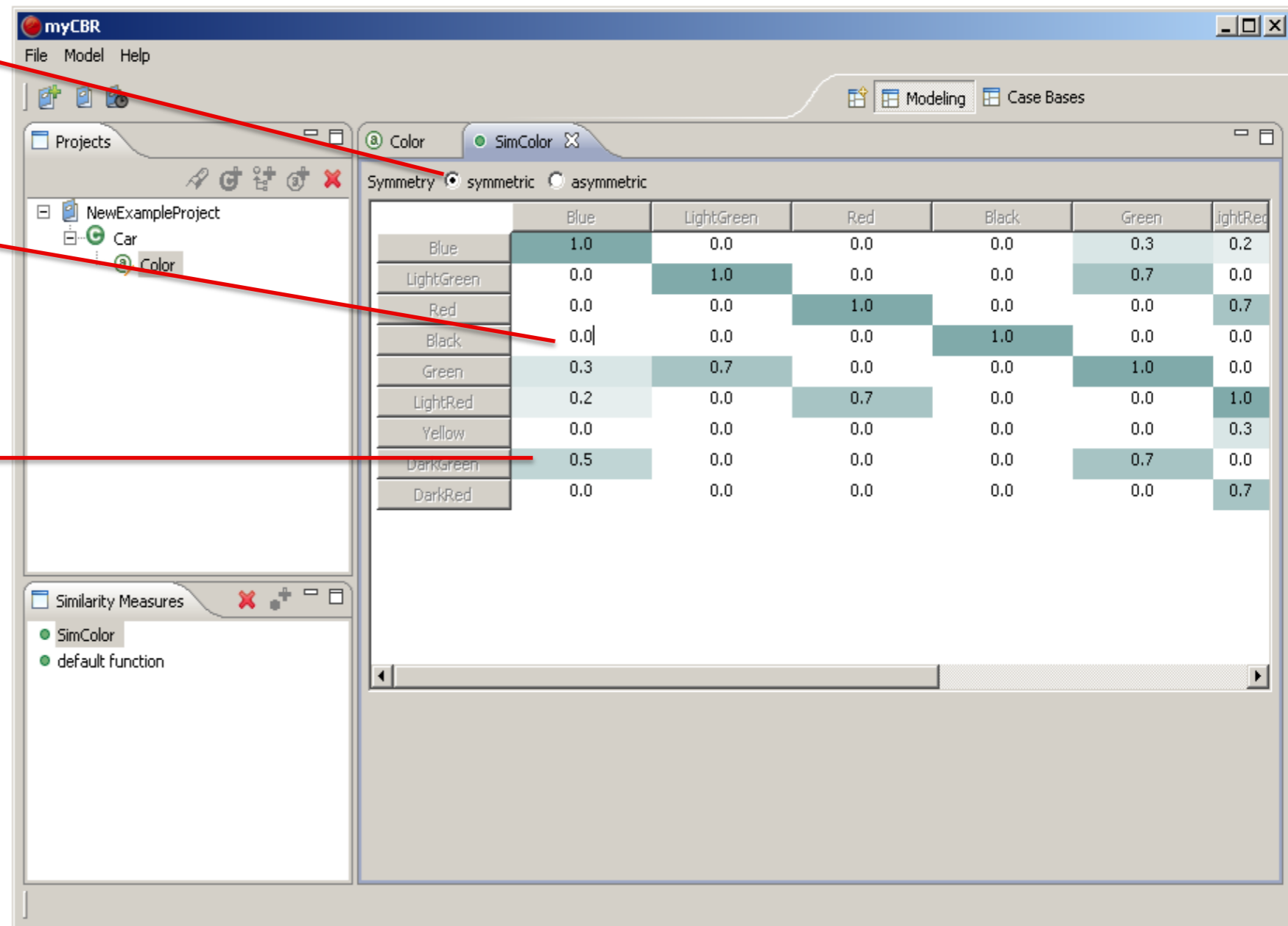


# Modeling a similarity measure for a symbol attribute

1) You can choose between a symmetric or asymmetric matrix

2) You can enter values between 0.0 and 1.0 by directly clicking into a field.

3) The values are also color-coding their cells to help you with visual feedback while modelling your similarity measure.



The screenshot shows the myCBR software interface. The 'SimColor' window is active, displaying a similarity matrix for the 'Color' attribute. The matrix is symmetric, as indicated by the 'symmetric' radio button being selected. The matrix values are color-coded: 1.0 is dark blue, 0.7 is medium blue, 0.5 is light blue, 0.3 is very light blue, and 0.0 is white. The 'Projects' pane on the left shows a project named 'NewExampleProject' with a 'Car' attribute and a 'Color' attribute. The 'Similarity Measures' pane at the bottom shows 'SimColor' and 'default function'.

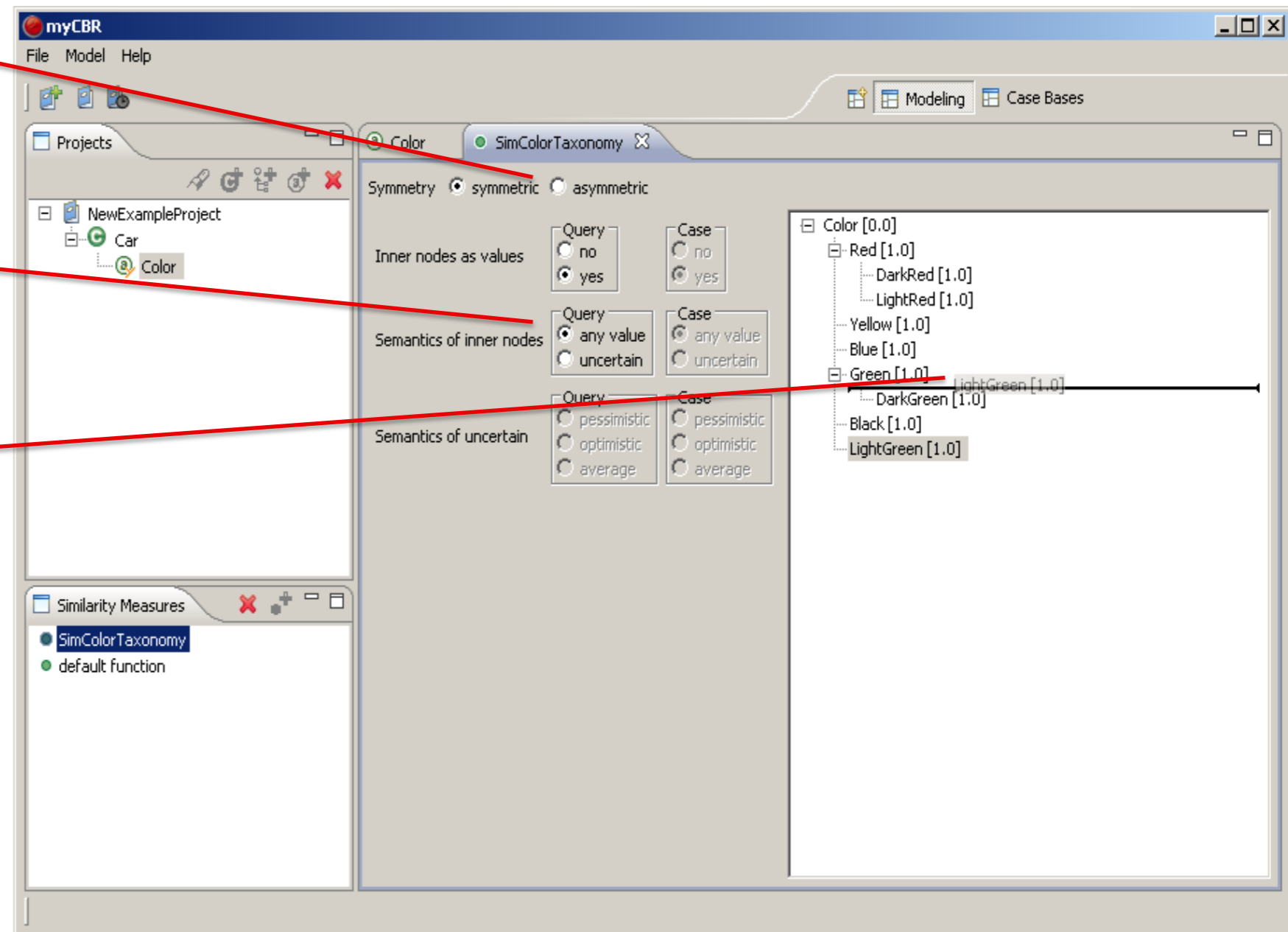
	Blue	LightGreen	Red	Black	Green	lightRed
Blue	1.0	0.0	0.0	0.0	0.3	0.2
LightGreen	0.0	1.0	0.0	0.0	0.7	0.0
Red	0.0	0.0	1.0	0.0	0.0	0.7
Black	0.0	0.0	0.0	1.0	0.0	0.0
Green	0.3	0.7	0.0	0.0	1.0	0.0
LightRed	0.2	0.0	0.7	0.0	0.0	1.0
Yellow	0.0	0.0	0.0	0.0	0.0	0.3
DarkGreen	0.5	0.0	0.0	0.0	0.7	0.0
DarkRed	0.0	0.0	0.0	0.0	0.0	0.7

# Modeling a similarity measure as a symbol taxonomy

You can choose between a symmetric or asymmetric calculation

You can specify distinct ways to describe (calculate) the similarity for the query and for the case comparison.

You can arrange the symbols in the taxonomy by drag & drop.



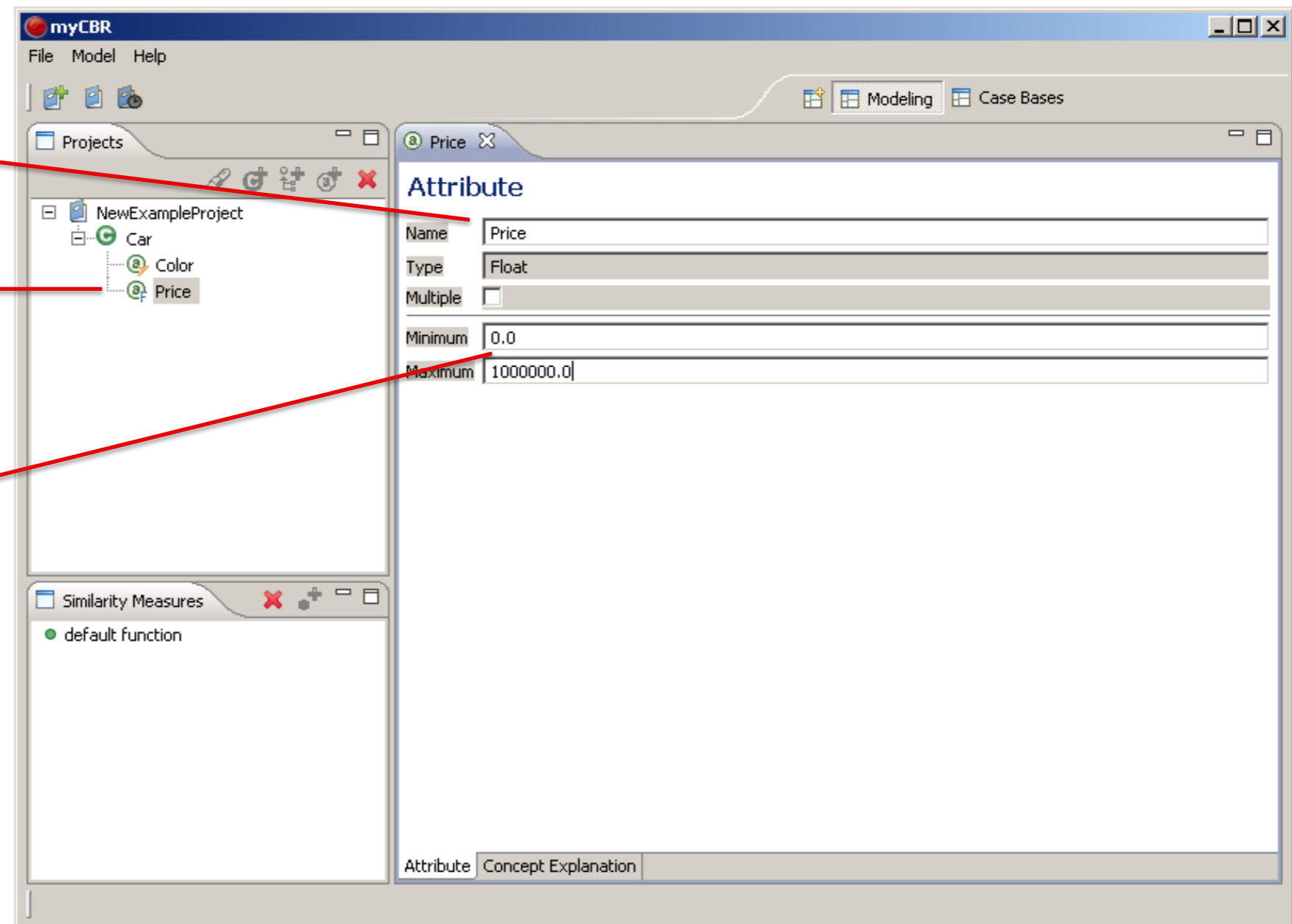


# Adding a similarity measure for a float attribute

After adding a new attribute of the data type float you can again specify its name.

Access the attributes characteristics by double-clicking it.

For float, as well as integer attributes you are required to enter a minimum and maximum value wherein the minimum requires to be lower than the maximum value.



# Modeling a similarity measure for a float attribute

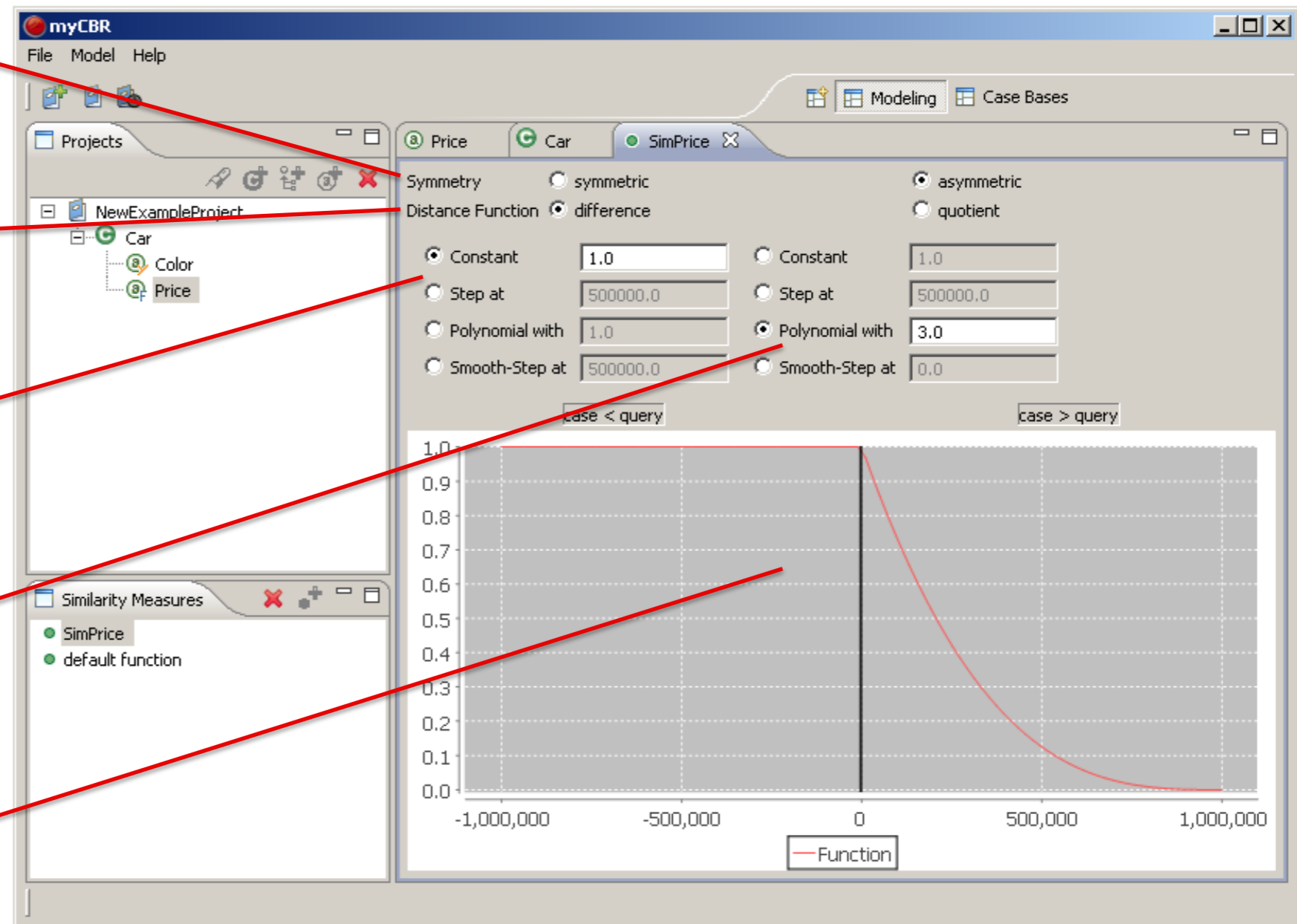
You can select symmetric or asymmetric behaviour of the function. In our example we have chosen an asymmetric function

You can decide if you want to have the plain distance or a quotient calculated as a similarity value.

In asymmetric mode this set (the left one) of options shapes the half of the function that calculates the similarity for query values greater then the values in a given case.

In asymmetric mode this set (the right one) of options shapes the half of the function that calculates the similarity for query values smaller then the values in a given case.

As we want to model the idea that a higher price is bad we made the function rapidly decline the calculated similarity value after passing the query value (aka Having a higher price in the case then in the query).

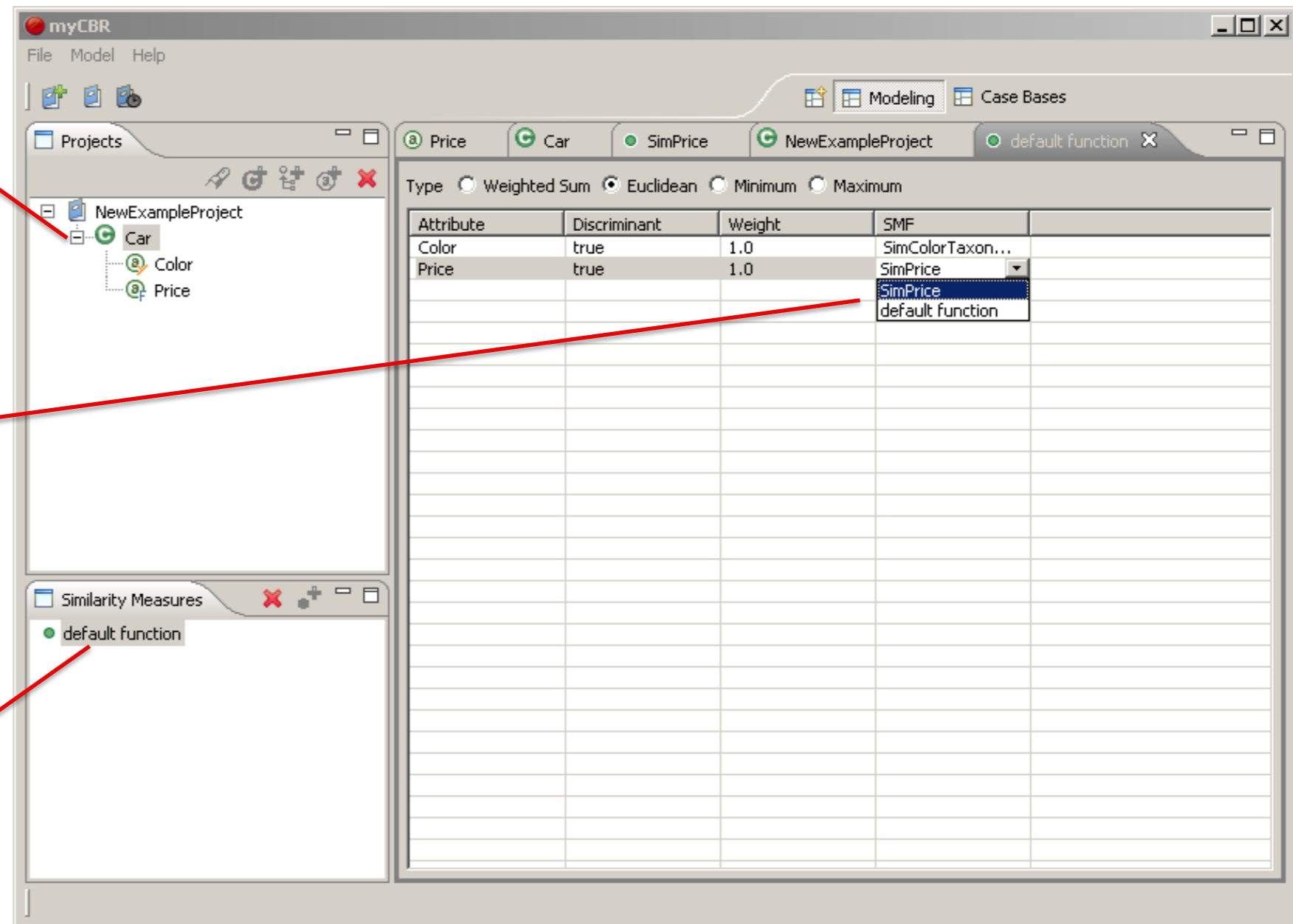


# Assigning the similarity measure to the float attribute

1) Double-click the concept that contains the attribute you want to assign a similarity measure for.

3) Select the desired similarity measure from the drop down menu "SMF". In our example we are choosing "SimPrice" as the similarity measure to be used to compute the local similarity of two float numbers representing a cars price within a query and our cases.

2) Double-click the global similarity measure of the concept (car in our case) default function (or the currently selected similarity measure for the concept).



The screenshot shows the myCBR software interface with the following components:

- Projects Panel:** Displays a tree structure under "NewExampleProject" containing a "Car" concept, which has two attributes: "Color" and "Price".
- Similarity Measures Panel:** Located at the bottom left, it shows a list of similarity measures, with "default function" selected.
- Modeling Panel:** The main workspace on the right, showing a table with columns: Attribute, Discriminant, Weight, and SMF. The "Price" attribute is selected, and the "SMF" dropdown menu is open, showing "SimPrice" as the chosen measure.

The table in the Modeling Panel is as follows:

Attribute	Discriminant	Weight	SMF
Color	true	1.0	SimColorTaxon...
Price	true	1.0	SimPrice

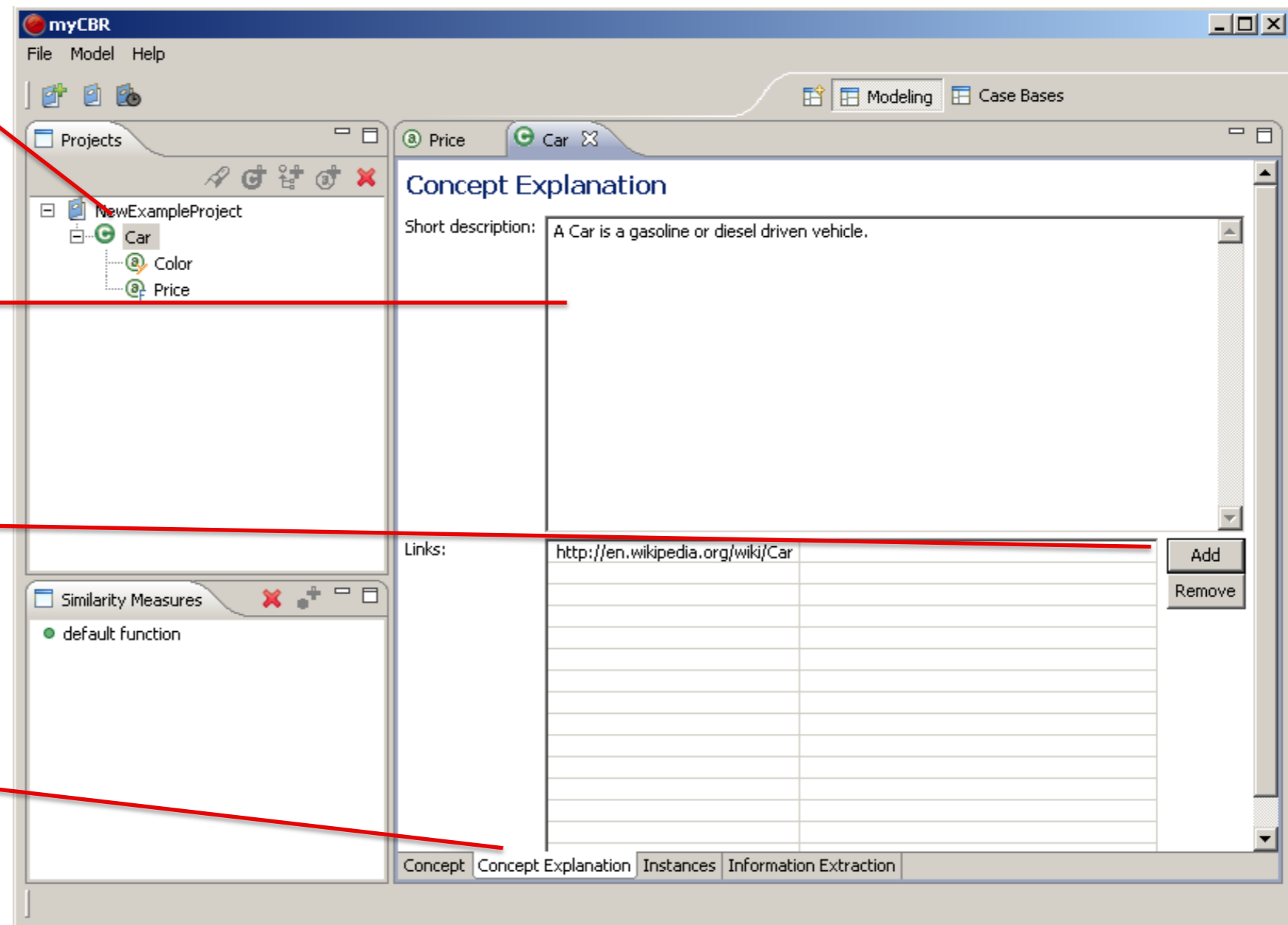
# Adding explanations to a concept

Select the concept you want to add canned explanations and/or an explanatory artefact for.

You can enter a free text that will be available via the API as the canned explanation for this concept.

Or you can click "Add" to add a reference (URL) to an explanatory artefact (for example a webpage with further information about the concept) for the concept.

Remember that you have to click on the "concept Explanation" Tab to get to the Concept Explanation edit view. To normally keep working with the concept make sure you switch back to the "Concept" tab.



# Modeling the global Similarity Measure

You can select the functions nature here. In the example we have chosen a weighted sum.

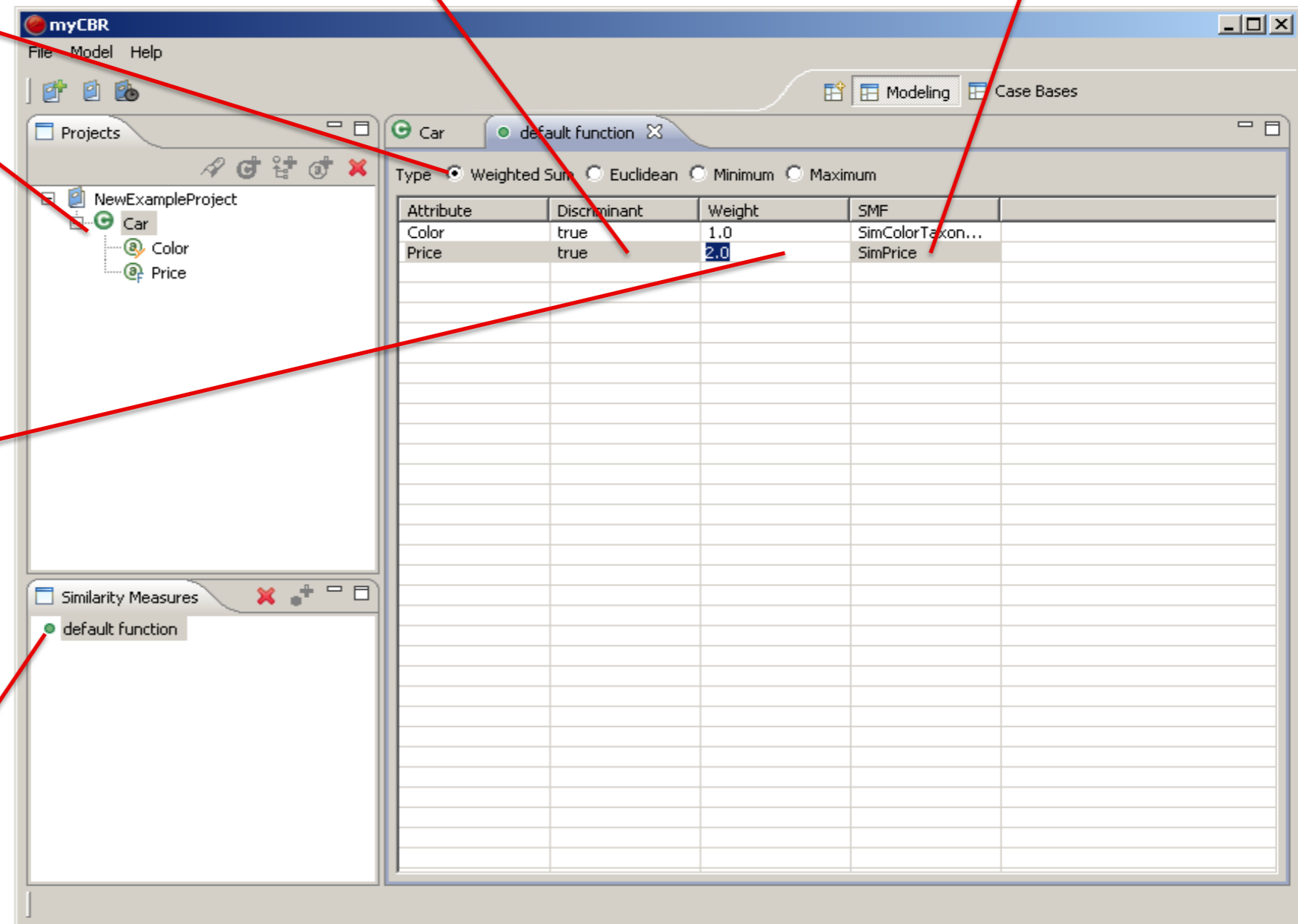
By setting "Discriminant" to "false" you exclude the local similarity value from being used in the calculation of the global similarity value.

Clicking in the "SMF" field allows you to chose one of the available similarity functions associated with the attribute.

Select the concept you want to define a global similarity measure for

Double clicking in the "Weight" field allows you to enter a value for the weight of the local similarity measure associated with this field. The weighted sum of all local similarity measures will then form the global similarity between, in our example, one car and another. In our example "Price is now twice as important for the global similarity between two cars than "Color" is.

Select the concepts default function to edit it.

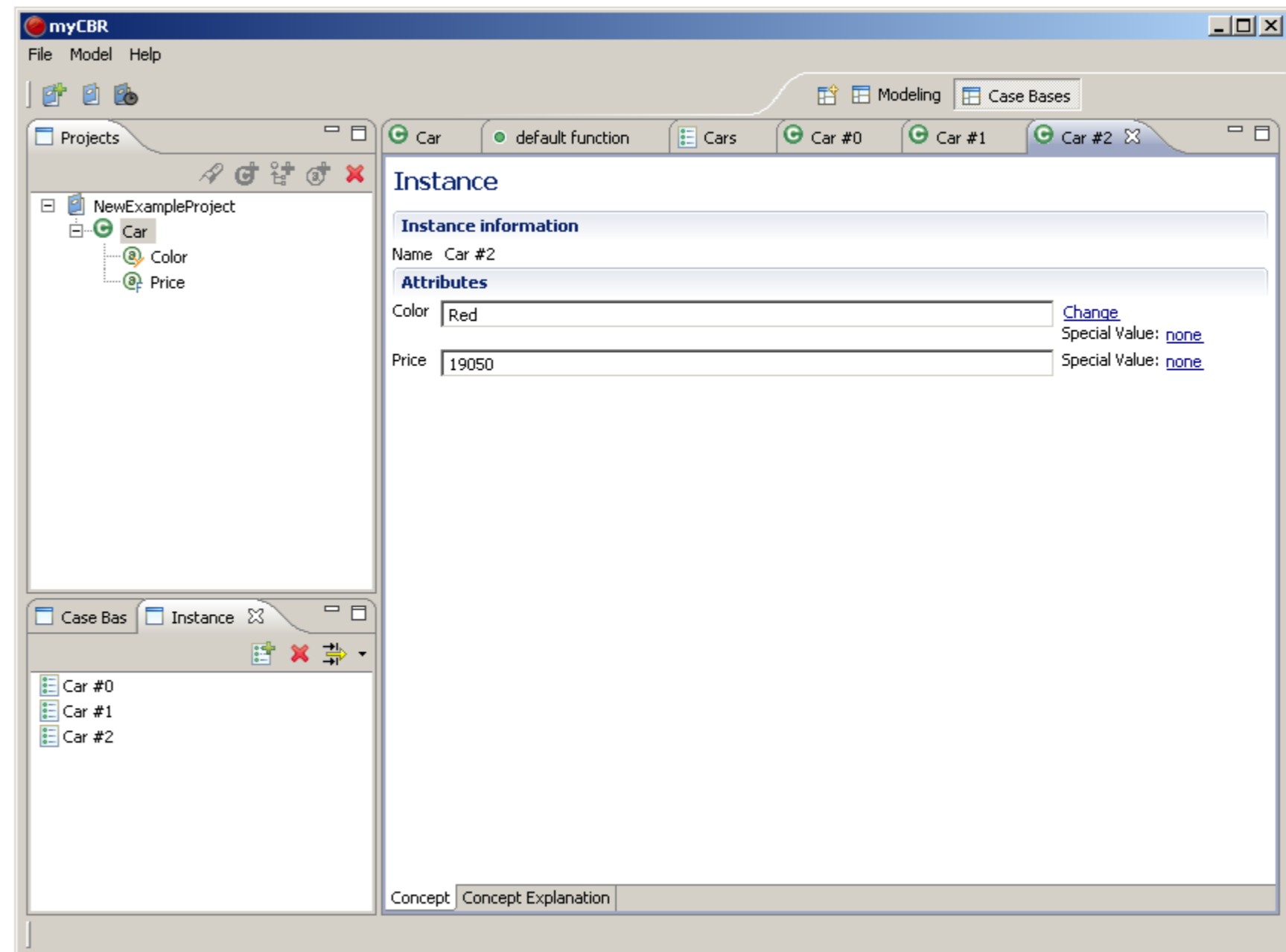


The screenshot shows the myCBR software interface. The 'Modeling' tab is active. The 'Car' concept is selected. The 'default function' is chosen. The 'Type' is set to 'Weighted Sum'. The table shows attributes 'Color' and 'Price' with 'Discriminant' set to 'true', 'Weight' of 1.0 and 2.0 respectively, and 'SMF' set to 'SimColorTaxon...' and 'SimPrice'. The 'Similarity Measures' panel shows 'default function' selected.

Attribute	Discriminant	Weight	SMF
Color	true	1.0	SimColorTaxon...
Price	true	2.0	SimPrice

# Doing a Retrieval test [1]

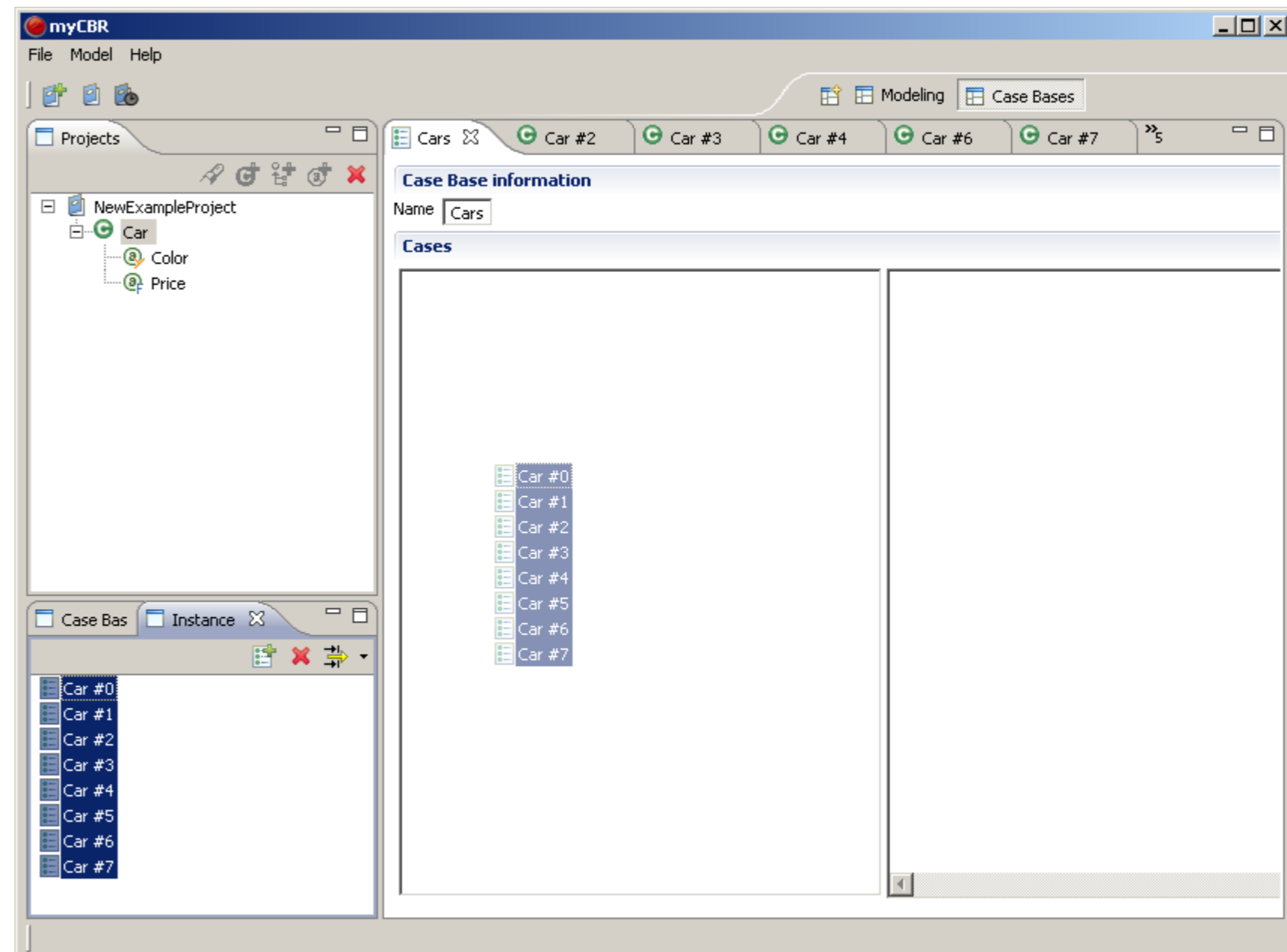
After adding a case-base to your project... [See Slide 85]



## Doing a Retrieval test [2]

And adding some instances of the concept “car” aka “cases” to the case base... See slides: 82, 83  
You are ready to do a retrieval test.

The retrieval GUI is shown  
And explained again on the  
following slide.



# Retrieval GUI

## Reminder



Select a case base from the available case bases within your project via this dropdown menu

Change: Allows you to select a new symbol from the list of symbols that are defined for the Attribute

Special value: Allows you to select a special value (like "unknown") for the attribute. This is useful to formulate sparse queries.

List view of the cases present in the selected case base

The concept car is selected as the concept to start a retrieval for.

This input form lets you specify concrete values for all attributes. By specifying attribute values you formulate your query to the system.

Start retrieval: Start the retrieval process on the selected case base with the specified query.

This list shows the result set of a retrieval. The overall similarity is shown and also used to sort the retrieved cases by their similarity to the case specified in the retrieval query

The screenshot shows the myCBR Retrieval GUI. The 'Case base' dropdown is set to 'CaseBase0'. The 'Query' form contains the following values:

- Body: sedan
- CCM: 1000
- Color: black
- Doors: 4
- Gas: gasoline
- Manufacturer: mercedes-benz
- Miles: \_unknown\_
- Model: \_unknown\_
- Power: \_unknown\_
- Price: 15000
- Speed: 150
- Year: 1990

Each attribute has a 'Change' link and a 'Special Value' dropdown. The 'Special Value' dropdowns are set to 'none' for most attributes and '\_unknown\_' for Miles, Model, and Power.

The 'Start retrieval' button is at the bottom of the query form.

The 'List view of the cases' shows a list of retrieved cases with their similarity scores:

- 424\_mercedes-benz - 0.38
- 765\_mercedes-benz - 0.38
- 549\_mercedes-benz - 0.36
- 734\_mercedes-benz - 0.36
- 529\_mercedes-benz - 0.36
- 876\_mercedes-benz - 0.36
- 143\_mercedes-benz - 0.36
- 704\_mercedes-benz - 0.35
- 770\_mercedes-benz - 0.35
- 684\_mercedes-benz - 0.35
- 63\_mercedes-benz - 0.35
- 853\_mercedes-benz - 0.35
- 142\_mercedes-benz - 0.34
- 310\_mercedes-benz - 0.33
- 747\_mercedes-benz - 0.32
- 674\_audi - 0.3
- 181\_audi - 0.3
- 556\_audi - 0.3
- 564\_audi - 0.3
- 455\_audi - 0.3
- 742\_audi - 0.27
- 936\_audi - 0.27
- 206\_bmw - 0.27
- 297\_mercedes-benz - 0.26
- 779\_mercedes-benz - 0.26

The 'Similarity Measures' table shows the overall similarity and attribute values for the retrieved cases:

	424_mercedes-...	765_mercedes-...	549_mercedes-...	734_mercedes-...
Similarity	0.38	0.38	0.36	0.36
Body	sedan	sedan	sedan	sedan
CCM	2300	3200	2200	2200
Car Code	424	765	549	734
Color	black	black	black	black
Doors	4	4	4	4
Gas	gasoline	gasoline	diesel	diesel
Manufacturer	mercedes-benz	mercedes-benz	mercedes-benz	mercedes-benz
Miles	18726	89768	18684	42550
Model	c_230_kompres...	e_320	e_220_diesel	c_220_diesel
Power	192	224	95	95

## Doing a Retrieval test [3]

Seen in our example here: We did a retrieval with the query 'Color = red and Price = 5500'.

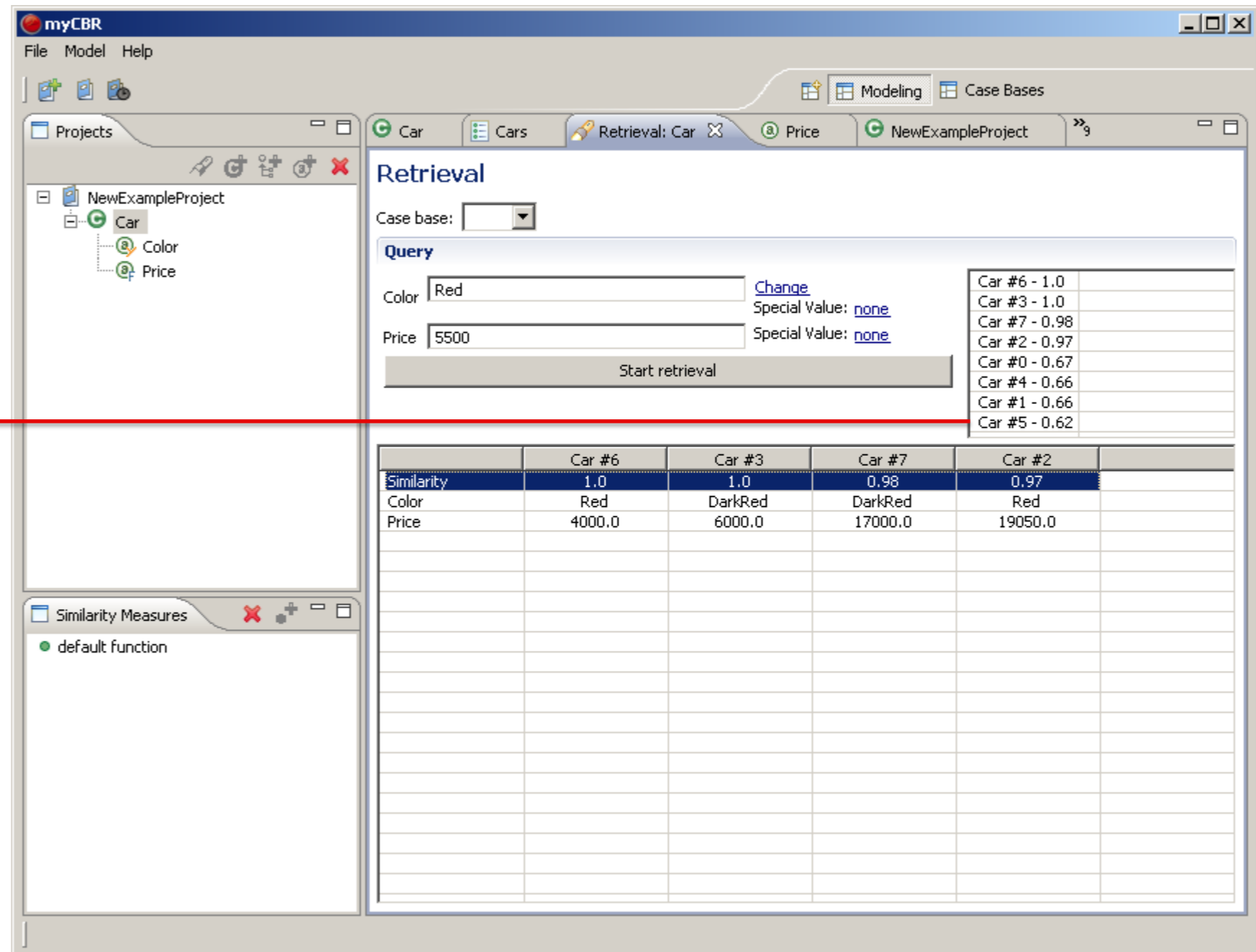
As you can see the cases in our case base are not very discriminable with respect to their similarity to the query.

This is an effect caused by the attribute 'Price' which we defined with a value range of 0 to 1.000.000.

As we only have cases (aka cars) with a price up to 30.000 the value range for the attribute "Price" is far to wide for our data (cases) at hand right now.

How can we change our model to reflect this knowledge, gathered by retrieval testing?

See how, in the following section: Padding your knowledge model.



The screenshot shows the myCBR software interface. The 'Retrieval' window is active, displaying a query with the following attributes:

- Color: Red
- Price: 5500

Below the query, there is a 'Start retrieval' button. To the right of the query, a table lists the similarity scores for five cars:

Car #	Similarity
Car #6	1.0
Car #3	1.0
Car #7	0.98
Car #2	0.97
Car #0	0.67
Car #4	0.66
Car #1	0.66
Car #5	0.62

Below the similarity scores, a table displays the attributes of the retrieved cases:

	Car #6	Car #3	Car #7	Car #2
Similarity	1.0	1.0	0.98	0.97
Color	Red	DarkRed	DarkRed	Red
Price	4000.0	6000.0	17000.0	19050.0

The 'Similarity Measures' window at the bottom shows the 'default function' selected.

# myCBR Getting started: Refining your knowledge model



## Doing a Retrieval test (again) [4]

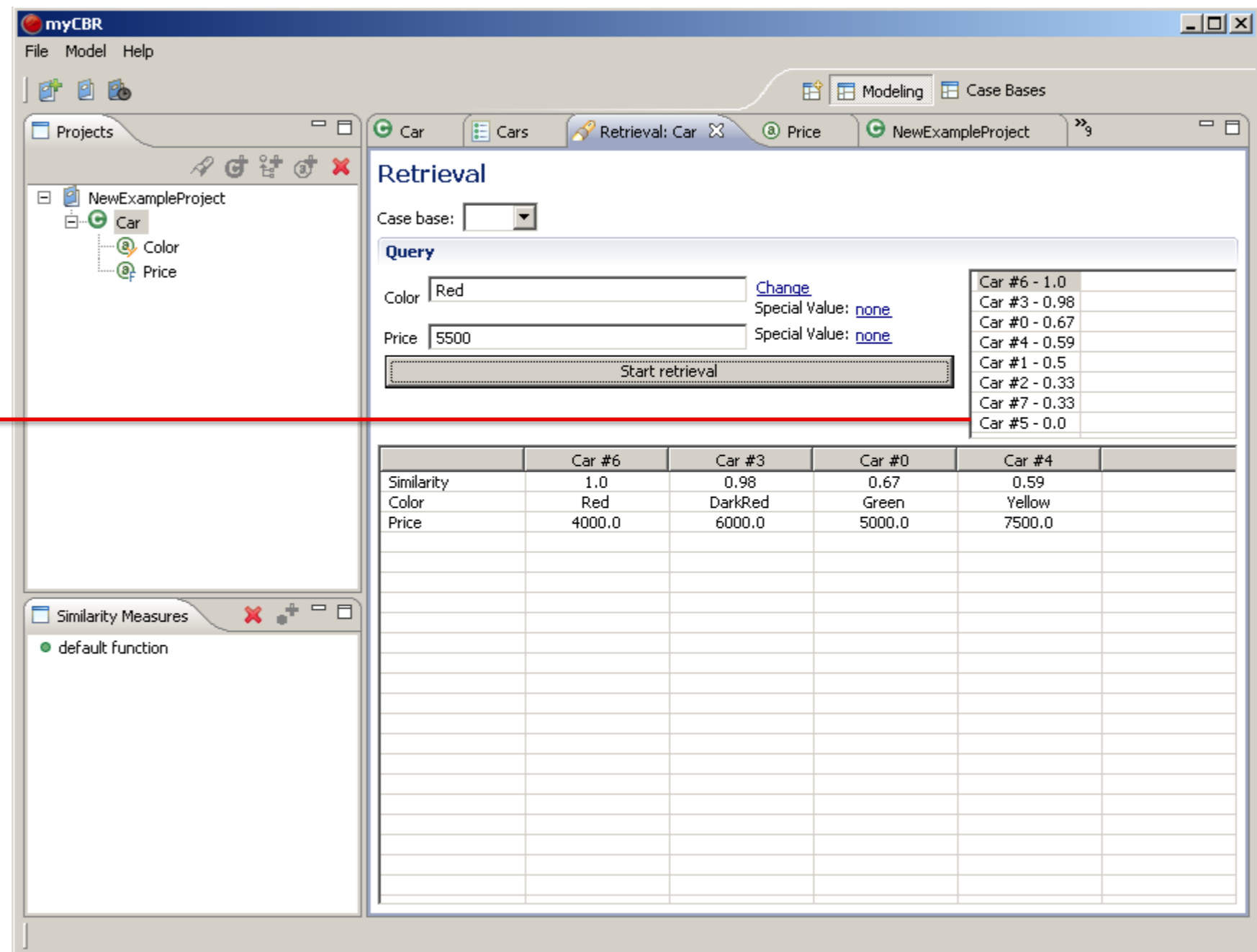
As we have seen our knowledge model is not yet accurate enough, as it was revealed by our first retrieval test at the end of the previous section). To amend the far to high value for “Price” we adapt price to 50000 and the cases become more distinguishable

Changing the value range of the attribute “Price” to 0 to 50.000 and then redoing the retrieval with the same query as in our first retrieval test results in a far more distinguishable calculation of the case’s global similarities.

As you can see to the right, the similarities of the cases to the query are now more distinct and spread over the interval [0...1] than within the first retrieval test result depicted here again:

Car #6 - 1.0
Car #3 - 1.0
Car #7 - 0.98
Car #2 - 0.97
Car #0 - 0.67
Car #4 - 0.66
Car #1 - 0.66
Car #5 - 0.62

Thus, by adapting the value range of the attribute “Price”, based upon results from our first retrieval test, we enhanced the quality of our knowledge model and thus the retrieval results it provides.



The screenshot shows the myCBR software interface. The main window is titled 'Retrieval' and contains a 'Query' section with the following fields:

- Case base:
- Color:  [Change](#)
- Price:  [Special Value: none](#)
- [Special Value: none](#)
- [Start retrieval](#)

The 'Similarity Measures' window shows the 'default function' selected.

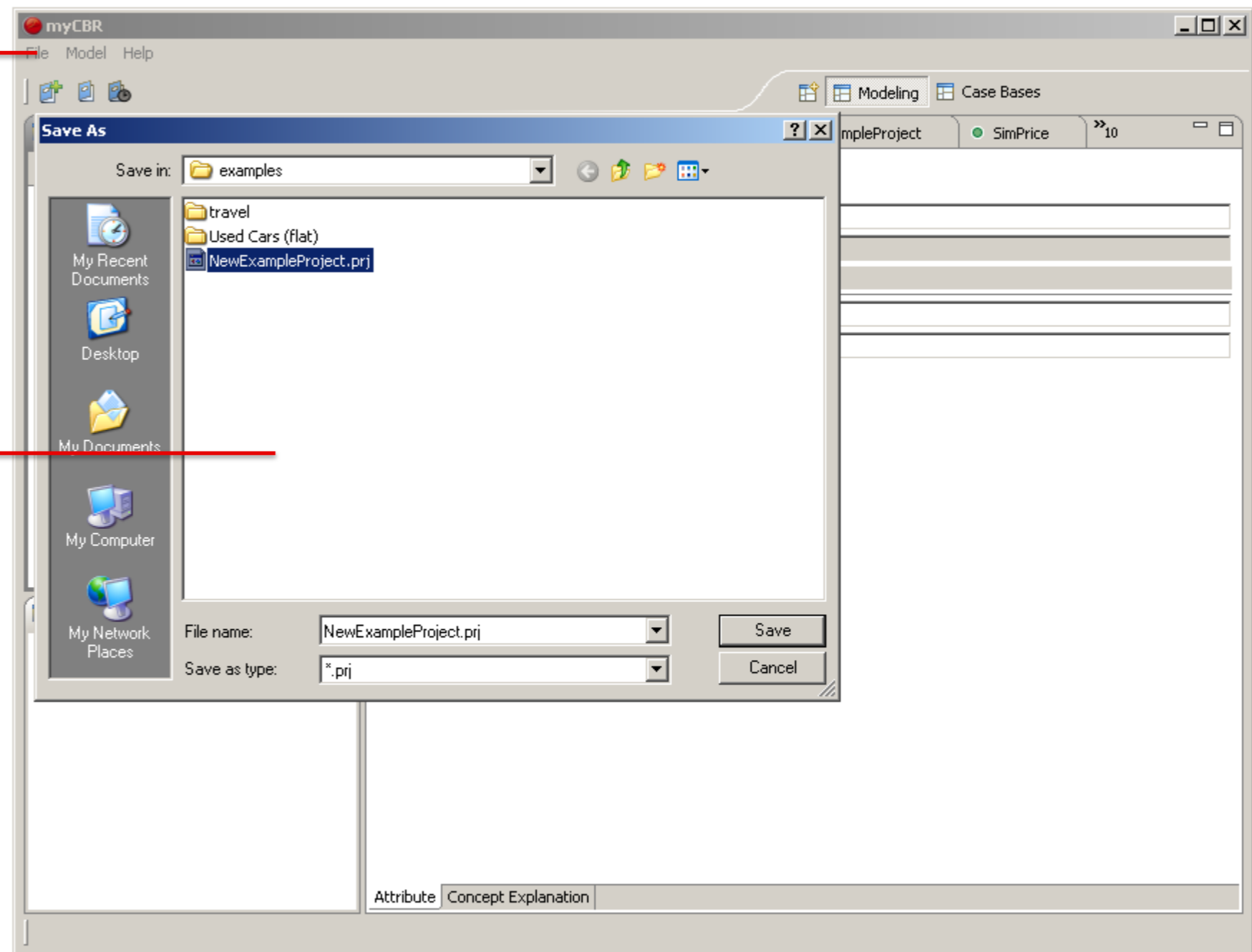
The 'Retrieval' window also displays a table of similarity results for various cars:

	Car #6	Car #3	Car #0	Car #4
Similarity	1.0	0.98	0.67	0.59
Color	Red	DarkRed	Green	Yellow
Price	4000.0	6000.0	5000.0	7500.0

# After successful refinement: Save your model (project)

Select the project you want to save then in the "File" dialog select "Save" or "Save as".

In the dialog select the place to save your project in, name it and click on "Save".



# Goals of knowledge model refinement

***The goals of knowledge model refinement are:***

## ***Enhance the performance of your models retrieval***

- ✱ Remove unnecessary attributes
- ✱ Reduce value ranges
- ✱ Streamline similarity measures
- ✱ Trim your case base (remove redundant, rarely used cases)

## ***Enhance the accuracy of your model***

- ✱ Refine similarity measures
- ✱ Add or diversify attributes to your concept

## General tips for knowledge model refinement

### ***Enhance the performance of your models retrieval while not missing a thing:***

- ✿ Try to identify attributes that doesn't have great impact on your retrieval, then remove them
- ✿ Reduce your similarity measures to the absolute necessary, as their computation takes most effort
- ✿ Monitor the frequency of your cases being retrieved, remove redundant or rarely used cases
- ✿ User-test your case base(s) to see if you have already integrated most of the cases a user can come up with in your domain context

### ***Enhance the accuracy of your model while keeping it lean:***

- ✿ Add or delete locale similarities from your global similarity measure to keep it lightweight but also precise enough
- ✿ 'Sharpen' your similarity measures by ranging the value ranges to ranges to be encountered in the day to day use of your model
- ✿ User-test the modelling of your concepts with users unfamiliar to your model to, this might yield valuable feedback to be integrated in your model

# The cycle of knowledge model refinement

The cycle of knowledge model refinement offers the opportunity for iterative knowledge model optimisation even at runtime of your application. As the CBREngine respectively your knowledge model and data are modularised from your application (separated) you can optimise it and 're plug' thus reintegrate the optimised version to your application and instantly benefit from the improvements within your live application. This way you can choose if you want to optimise your CBR Engine until you reach certain quality measures or want to constantly have it evolve during being used. Integrating the

