# Rapid Prototyping of CBR Applications with the Open Source Tool myCBR

Armin Stahl[1] and Thomas R. Roth-Berghofer[2]

[1] German Research Center for Artificial Intelligence (DFKI) GmbH
Image Understanding and Pattern Recognition Department (IUPR)
`Armin.Stahl@dfki.de`

[2] German Research Center for Artificial Intelligence (DFKI) GmbH
Knowledge Management Department
Trippstadter Straße 122, 67663 Kaiserslautern, Germany
`Thomas.Roth-Berghofer@dfki.de`

**Abstract.** Although Case-Based Reasoning (CBR) claims to reduce the effort required for developing knowledge-based systems substantially compared with more traditional Artificial Intelligence approaches, the implementation of a CBR application from scratch is still a time consuming task. In this paper we present a novel, freely available tool for rapid prototyping of CBR applications that focuses on the similarity-based retrieval step, like for example case-based product recommender systems. By providing easy to use model generation, data import, similarity modeling, explanation, and testing functionality together with comfortable graphical user interfaces, the tool enables even CBR novices to rapidly create their first CBR applications. Nevertheless, at the same time it ensures enough flexibility to enable expert users to implement advanced CBR applications.

## 1 Introduction

The development of a quite simple Case-Based Reasoning application already involves a number of steps, such as collecting case and background knowledge, modeling a suitable case representation, defining an accurate similarity measure, implementing retrieval functionality, and implementing user interfaces. Compared with other AI approaches, CBR allows to reduce the effort required for knowledge acquisition and representation significantly, which is certainly one of the major reasons for the commercial success of CBR applications. Nevertheless, implementing a CBR application from scratch remains a time consuming software engineering process and requires a lot of specific experience beyond pure programming skills.

Although CBR research has a history of about 20 years now, and in spite of the broad commercial success of CBR applications in recent years, today only few CBR software tools for supporting the development process are available. Software products used for implementing large-scale commercial applications are typically very complex, consist of various modules, and require quite a long time to get familiar with the various functionalities and configuration possibilities. Another problem are the high licensing costs of these products and also if the vendors provide cheap or even free research licences, it is usually impossible to just download the software without carrying out an annoying registration procedure. This makes these products little attractive for research, teaching, small commercial projects, or first feasibility studies.

For these purposes, more easily available and less complex CBR tools are required. Unfortunately, such solutions are nearly missing today at all. One exception is the Open Source *JColibri*[1] system, which provides a framework for building CBR systems based on state-of-the-art Software Engineering techniques [1]. The key idea of the system is to combine software reuse with the more general AI paradigm of separating the reasoning algorithms from the domain model.

In this paper we present the novel Open Source CBR tool *myCBR*[2] developed at the German Research Center for Artificial Intelligence (DFKI). The key motivation for implementing *myCBR* was the need for a compact and easy-to-use tool for building prototype CBR applications in teaching, research, and small industrial projects with minimal effort. Moreover, the tool should be easily extendable in order to facilitate the experimental evaluation of novel algorithms and research results. Many ideas for the implementation of *myCBR* came from the old *CBR-Works* system[3] [2] but which is not available any more.

The current version of *myCBR* still focuses on the similarity-based retrieval step of the CBR cycle [3], because that is still the core functionality of most CBR applications. A popular example of such retrieval-only systems are case-based product recommender systems [4]. While the first CBR systems were often based on simple distance metrics, today many CBR applications make use of highly sophisticated, knowledge-intensive similarity measures [5]. On the one hand, such extremely domain specific similarity measures enable to improve the retrieval quality substantially. However, on the other hand, they increase the development effort significantly.

The major goal of *myCBR* is to minimize the effort for building CBR applications that require knowledge-intensive similarity measures. Therefore, it provides comfortable graphical user interfaces for modeling various kinds of attribute-specific similarity measures and for evaluating the resulting retrieval quality. In order to reduce also the effort of the preceding step of defining an appropriate case representation, it includes tools for generating the case representation automatically from existing raw data.

In the next Section we give an overview of the basic concept and system architecture of *myCBR*. In Section 3 and 4 we then describe the technical approaches, how rapid prototyping of CBR applications is supported by *myCBR*. In Section 5 we conclude with a summary and an outlook on future plans for improving and extending *myCBR*.

[1] http://gaia.fdi.ucm.es/projects/jcolibri
[2] http://www.mycbr-project.net
[3] CBR-Works has been developed at the University of Kaiserslautern in cooperation with empolis knowledge management GmbH, former tecinno GmbH.

## 2 The myCBR Architecture

From its conception, *myCBR* was designed with improved communication between the system and the user—knowledge engineer and end-user—in mind. The novice as well as the expert knowledge engineer is supported during the development phase of a *myCBR* project through intelligent support approaches and advanced GUI functionality.

The foundation of every CBR system are certain knowledge representation formalisms required to describe the content of the individual CBR knowledge containers, namely the *vocabulary*, the *similarity measure*, the *adaptation knowledge*, and the *case knowledge* [6]. Since knowledge representation is a key issue for most Artificial Intelligence (AI) Systems, various software tools for supporting knowledge engineering tasks are already existing today.

One of the most popular and widely used systems is certainly the Java-based Open Source ontology editor Protégé[4] [7]. A major reason for the success of Protégé is its flexible extensibility by providing a plug-and-play environment that enables users to add and distribute new modules easily. This makes Protégé a flexible base for rapid prototyping and application development in various application domains.

In order to avoid a reinvention of the wheel, we have chosen Protégé as the modeling platform for *myCBR*. In our point of view, the use of Protégé brings two main advantages: First, the effort for implementing data structures and user interfaces for representing the vocabulary and the case knowledge can be saved. Second, it allows to add CBR functionality to existing Protégé applications with minimal effort. Due to the large community of Protégé developers and users, in the long term this may help to spread the use of CBR technology in other AI communities.

The basic architecture of *myCBR* is illustrated in Figure 1. During the development phase of an CBR application, *myCBR* runs as a plug-in within Protégé. This plug-in consists of the following modules:

**Modelling tools:** These tools extend the existing functionality of Protégé for creating domain models and case instances and add the missing functionality for defining similarity measures.

**Retrieval GUI:** The retrieval GUI provides powerful features for analyzing the quality of the defined similarity measures. Moreover, it can also serve as the user interface of first prototypical CBR applications.

**Retrieval engines:** For executing the similarity-based retrieval, different retrieval engines are provided.

**Explainer:** A dedicated explanation component provides modelling support information as well as explanations of retrieval results for quicker roundtrips of designing and testing (see also Section 4).

After installing and activating the *myCBR* plug-in, the user interface of Protégé is extended with additional tabs to access the *myCBR* modules. Figure 2 shows, for

[4] http://protege.stanford.edu/

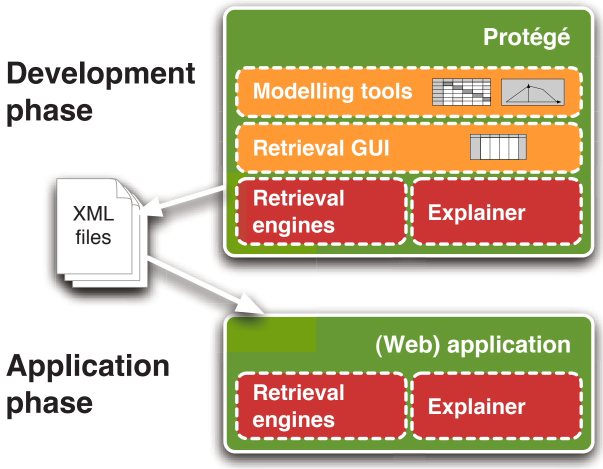


**Fig. 1.** The system architecture of *myCBR*

example, how the *myCBR* editor for configuring class specific similarity measures integrates into the Protégé environment consisting of class and slot browsers.

As a result of the modeling and development phase, the complete domain and similarity model together with the case base can be exported to XML files.

Although the *myCBR* Protégé plug-in already allows to create a full and running application, in many projects custom-made user interfaces and an integration of the CBR system into existing infrastructure is required. For this purpose, after developing a CBR application using the Protégé plug-in, *myCBR* can also be used as a stand-alone Java module, to be integrated in arbitrary applications, for example, JSP[5]-based web applications. In this application phase, the retrieval engines of *myCBR* just read the XML files generated during the previous development phase and perform the similarity-based retrieval.

End-users of the final *myCBR*-enhanced application can be further supported by providing explanations about the retrieval process.

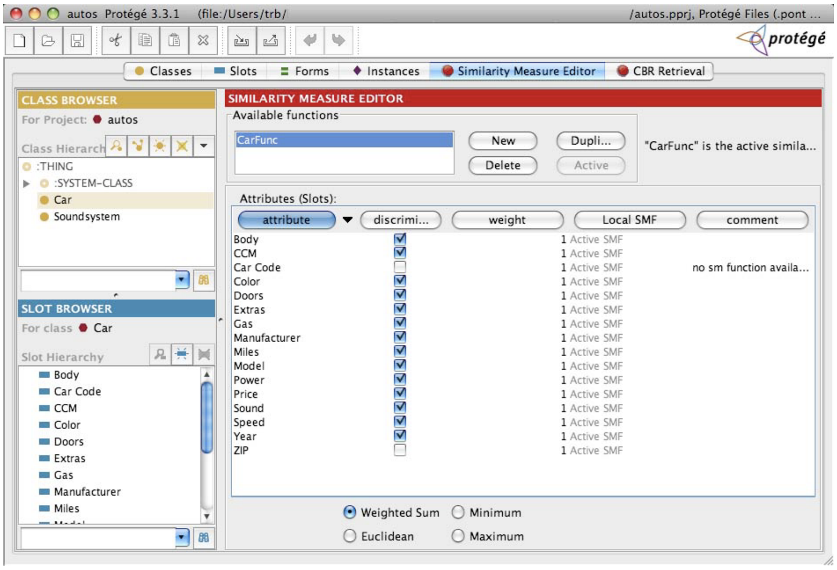
## 3 Developing CBR Applications with *myCBR*

In this section we describe in more detail how *myCBR* supports rapid prototyping of CBR applications. This includes the generation of case representations, the definition of similarity measures, the testing of retrieval and use of explanation functionality, and finally the implementation of stand-alone applications.

### 3.1 CSV Data Import and Automatic Model Generation

The starting point of many CBR projects is the collection of initial case data. The existence of at least some case examples is usually a precondition for modeling an accurate case representation and corresponding similarity measures.

[5] Java Server Pages.



"CarFunc" is the active simila...

| attribute | discrimi... | weight | Local SMF | comment |
|---|---|---|---|---|
| Body | ✓ | | 1 Active SMF | |
| CCM | ✓ | | 1 Active SMF | |
| Car Code | | | 1 Active SMF | no sm function availa... |
| Color | ✓ | | 1 Active SMF | |
| Doors | ✓ | | 1 Active SMF | |
| Extras | ✓ | | 1 Active SMF | |
| Gas | ✓ | | 1 Active SMF | |
| Manufacturer | ✓ | | 1 Active SMF | |
| Miles | ✓ | | 1 Active SMF | |
| Model | ✓ | | 1 Active SMF | |
| Power | ✓ | | 1 Active SMF | |
| Price | ✓ | | 1 Active SMF | |
| Sound | ✓ | | 1 Active SMF | |
| Speed | ✓ | | 1 Active SMF | |
| Year | ✓ | | 1 Active SMF | |
| ZIP | | | 1 Active SMF | |

[✓ Weighted Sum] Minimum
Euclidean Maximum

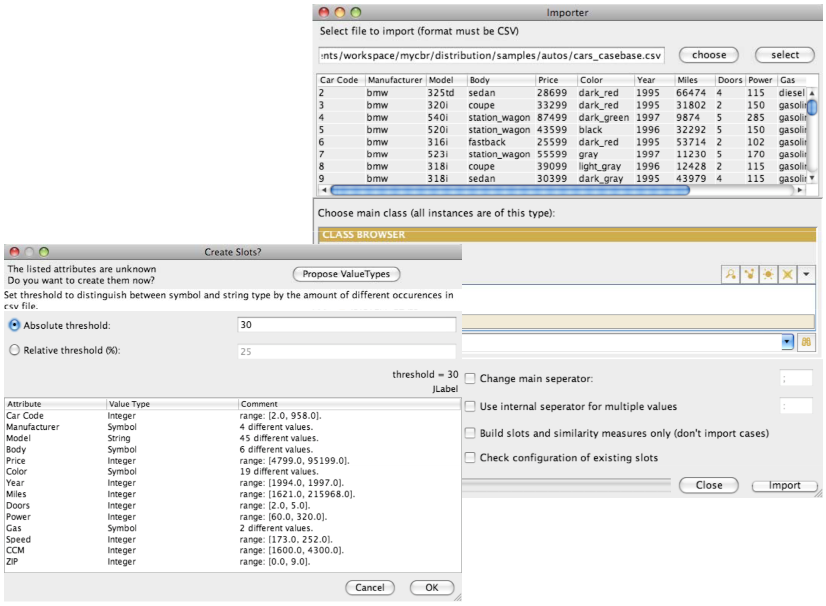
**Fig. 2.** The *myCBR* Protégé plug-in

*myCBR* is mainly intended for structural CBR applications that make use of rich attribute-value based or object-oriented case representations. Of course, since attribute values may also contain large parts of pure text, textual CBR applications are also supported by *myCBR*, but are not the main focus of the system. Although Protégé provides powerful graphical user interfaces for modeling attribute-value based and object-oriented representations, their manual definition remains a time consuming task. It includes the definition of classes and attributes (called "slots" in Protégé) and the specification of accurate value ranges required for a meaningful similarity assessment.

In order to facilitate the definition of case representations, *myCBR* provides a powerful CSV[6] data import module (see Figure 3.). CSV files are widely used to store attribute-value based raw data in pure ASCII format. For example, in the Machine Learning community example data sets are usually exchanged by using CSV files[7]. Using the CSV importer, the user has the choice to import data instances into an existing Protégé data model, or to create a new model automatically based on the raw data. In the latter case, *myCBR* generates a Protégé slot for each data column of the CSV file automatically. In order to achieve maximal flexibility, the CSV importer provides the following features:

**Slot creation:** The importer analyzes the whole CSV data in order to determine accurate value ranges for the slots automatically. For textual data, the user can specify a threshold on the number of unique values, in order to

[6] Comma Separated Values.
[7] See, for example, http://archive.ics.uci.edu/ml/



Importer

Select file to import (format must be CSV)

:nts/workspace/mycbr/distribution/samples/autos/cars_casebase.csv

| Car Code | Manufacturer | Model | Body | Price | Color | Year | Miles | Doors | Power | Gas |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | bmw | 325td | sedan | 28699 | dark_red | 1995 | 66474 | 4 | 115 | diesel |
| 3 | bmw | 320i | coupe | 33299 | dark_red | 1995 | 31802 | 2 | 150 | gasoli |
| 4 | bmw | 540i | station_wagon | 87499 | dark_green | 1997 | 9874 | 5 | 285 | gasoli |
| 5 | bmw | 520i | station_wagon | 43599 | black | 1996 | 32292 | 5 | 150 | gasoli |
| 6 | bmw | 316i | fastback | 25599 | dark_red | 1995 | 53714 | 2 | 102 | gasoli |
| 7 | bmw | 523i | station_wagon | 55599 | gray | 1997 | 11230 | 5 | 170 | gasoli |
| 8 | bmw | 318i | coupe | 39099 | light_gray | 1996 | 12428 | 2 | 115 | gasoli |
| 9 | bmw | 318i | sedan | 30399 | dark_gray | 1995 | 43979 | 4 | 115 | gasoli |

Choose main class (all instances are of this type):

CLASS BROWSER

Change main seperator:

Use internal seperator for multiple values

Build slots and similarity measures only (don't import cases)

Check configuration of existing slots

Create Slots?

The listed attributes are unknown
Do you want to create them now?

Set threshold to distinguish between symbol and string type by the amount of different occurences in csv file.

[✓ Absolute threshold:] 30

Relative threshold (%): 25

threshold = 30

jLabel

| Attribute | Value Type | Comment |
|---|---|---|
| Car Code | Integer | range: [2.0, 958.0]. |
| Manufacturer | Symbol | 4 different values. |
| Model | String | 45 different values. |
| Body | Symbol | 6 different values. |
| Price | Integer | range: [4799.0, 95199.0]. |
| Color | Symbol | 19 different values. |
| Year | Integer | range: [1994.0, 1997.0]. |
| Miles | Integer | range: [1621.0, 215968.0]. |
| Doors | Integer | range: [2.0, 5.0]. |
| Power | Integer | range: [60.0, 320.0]. |
| Gas | Symbol | 2 different values. |
| Speed | Integer | range: [173.0, 252.0]. |
| CCM | Integer | range: [1600.0, 4300.0]. |
| ZIP | Integer | range: [0.0, 9.0]. |

**Fig. 3.** The CVS data importer

control the generation of symbol and string slots. If a data column contains less unique values than specified, the slot becomes symbolic (with all found values as allowed symbols), otherwise it will be specified as a string slot.

**Model Update:** If a domain model is already existing, the CSV importer may update the model according to the data in the given CSV file. Then missing slots are created and value ranges of existing slots are updated once the data contains values that do not fit into the predefined ranges. This can also be done in a semi-automatic manner in order to investigate the differences between the data and the existing model in more detail.

**Creation of Instances:** The user can choose whether he wants to import the data by creating corresponding Protégé data instances or whether he wants to create the domain model only.

**Specification of Column Separators:** Since the use of column separators (comma, semicolon, etc.) is not standardized in CSV files, the user can specify the used separator prior to the import. By supporting a second level separator, *myCBR* is also able to import set attributes (attributes with multiple values).

After the CSV data has been imported, the user may further modify the generated case model (e.g. extend it to an object-oriented representation) in order to meet the application specific needs. The final case model together with the case base is stored by *myCBR* in XML files.

## 3.2 Modeling Similarity Measures

After having generated the case representation either by hand or by using the CSV importer, the main task for creating a CBR application with *myCBR* is the definition of an appropriate similarity measure. Here, *myCBR* follows the *local-global approach* which divides the similarity definition into a set of *local similarity measures* for each attribute, a set of *attribute weights*, and a *global similarity measure* for calculating the final similarity value. This means, for an attribute-value based case representation consisting of $n$ attributes, the similarity between a query $q$ and a case $c$ may be calculated as follows:

$$Sim(q,c) = \sum_{i=1}^{n} w_i \cdot sim_i(q_i, c_i)$$

Here, $sim_i$ and $w_i$ denote the local similarity measure and the weight of attribute $i$, and $Sim$ represents the global similarity measure. *myCBR* is also able to deal with more structured, object-oriented representations and supports suited global similarity measures as described in [8].

The editor for specifying global similarity measures was already shown in Figure 2. Besides the use of a weighted sum, the user can also choose another amalgamation function, i.e. the Euclidean distance. However, the most similarity knowledge is encoded in the attribute specific local similarity measures. For testing purposes and to ensure high flexibility, for both global and local similarity measures the user can define and manage a set of different measures. The measures that are currently marked as active are finally used for the retrieval.

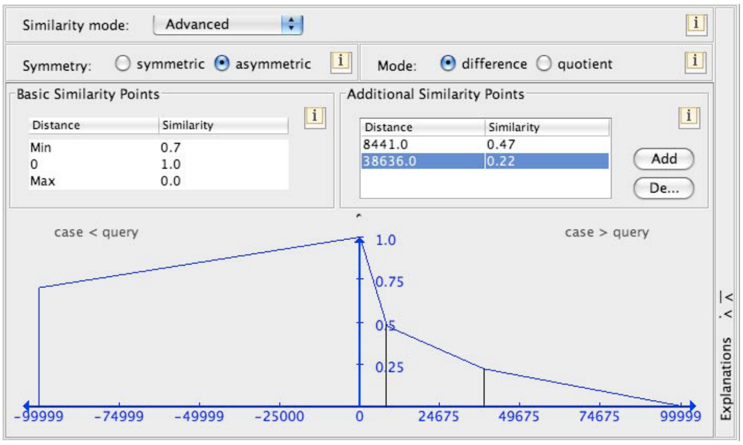
In the following sections we give an overview of the various approaches for modeling local similarity measures depending on the value type of the underlying attribute.

**Similarity Editors for Numeric Attributes.** For numerical attributes, the similarity computation is typically based on a mapping between the distance of the two values to be compared and the desired similarity value:

$$sim_i(q_i, c_i) = f(d(q_i, c_i))$$

This means, the similarity modeling focuses on the definition of an accurate mapping function $f$ for a given distance function $d$ [5]. For $d$, *myCBR* provides two alternatives, either the absolute difference $d(q_i, c_i) = c_i - q_i$ or the quotient $d(q_i, c_i) = \frac{c_i}{q_i}$ of the two values. The latter one allows to model similarities depending on a kind of relative distance, however, its application is restricted to strict positive value ranges.

For modeling the mapping function $f$, *myCBR* provides two editing modes. In the *standard mode*, the user can choose between some typical and adjustable functions (e.g. step or asymptotic decreasing functions). In the *advanced mode*, arbitrary mapping functions can be linearly approximated by specifying a set of sampling points. These sampling points can be easily generated and manipulated by using drag and drop functionality in a graphical editor (see Figure 4).


Similarity mode: Advanced

Symmetry: symmetric [✓ asymmetric]

Mode: [✓ difference] quotient

Basic Similarity Points

| Distance | Similarity |
|---|---|
| Min | 0.7 |
| 0 | 1.0 |
| Max | 0.0 |

Additional Similarity Points

| Distance | Similarity |
|---|---|
| 8441.0 | 0.47 |
| 38636.0 | 0.22 |

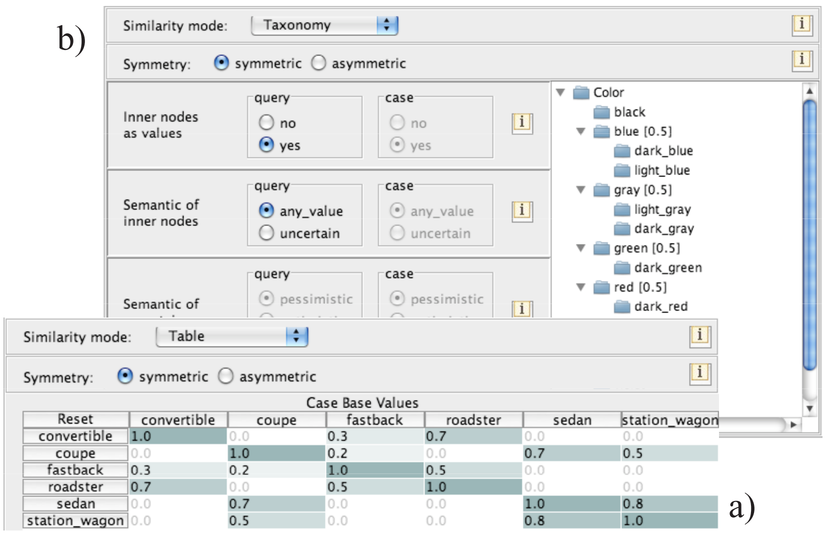
**Fig. 4.** The advanced similarity editor for numerical attributes

**Similarity Editors for Symbolic Attributes.** For symbolic attributes, several possibilities to model the similarity are supported. The most general and flexible way is the definition of a *similarity table* where all pairwise value combinations together with their similarities are enumerated explicitly (see Figure 5a). In order to make the editing as comfortable as possible, *myCBR* performs similarity highlighting (similarity values are visualized by different cell colors) and supports multiple cell selection.

However, for larger value sets the definition of similarity tables remains a time consuming and annoying task. Therefore, *myCBR* supports more comfortable approaches for defining similarities on symbolic values. The first one is the definition of a total *order* on symbols which allows to model the similarity like for numerical values by just using their position in the order. The second and more sophisticated approach is the arrangement of symbols in a *taxonomy* by using comfortable drag and drop functionality (see Figure 5b). Once the taxonomy and its application specific meaning is specified, it can be deployed to perform automatic similarity calculations (for details of this approach see [9]).

The user may start with the order or taxonomy approach to obtain a first similarity measure very quickly. In order to ensure maximal flexibility, *myCBR* supports the refinement of the similarity measure by switching to the table mode. Now the user may change some of the precalculated similarity values for considering his application specific needs.

**Similarity Editors for String Attributes.** Although textual CBR is not the main focus of the *myCBR* system, it provides flexible similarity measures for string processing. First, the user may choose between word or character-based processing modes. Depending on the selected mode, various approaches and configurations to specify the actual similarity calculation are provided, e.g. exact and partial matches, trigram matching, or regular expression based comparisons.


b)

Similarity mode: Taxonomy

Symmetry: [✓ symmetric] asymmetric

Inner nodes as values — query: no [✓ yes]; case: no [✓ yes]

Semantic of inner nodes — query: [✓ any_value] uncertain; case: [✓ any_value] uncertain

Semantic of — query: [✓ pessimistic]; case: [✓ pessimistic]

- Color
  - black
  - blue [0.5]
    - dark_blue
    - light_blue
  - gray [0.5]
    - light_gray
    - dark_gray
  - green [0.5]
    - dark_green
  - red [0.5]
    - dark_red

a)

Similarity mode: Table

Symmetry: [✓ symmetric] asymmetric

Case Base Values

| Reset | convertible | coupe | fastback | roadster | sedan | station_wagon |
|---|---|---|---|---|---|---|
| convertible | 1.0 | 0.0 | 0.3 | 0.7 | 0.0 | 0.0 |
| coupe | 0.0 | 1.0 | 0.2 | 0.0 | 0.7 | 0.5 |
| fastback | 0.3 | 0.2 | 1.0 | 0.5 | 0.0 | 0.0 |
| roadster | 0.7 | 0.0 | 0.5 | 1.0 | 0.0 | 0.0 |
| sedan | 0.0 | 0.7 | 0.0 | 0.0 | 1.0 | 0.8 |
| station_wagon | 0.0 | 0.5 | 0.0 | 0.0 | 0.8 | 1.0 |

**Fig. 5.** Similarity editors for symbolic attributes: similarity table (a) and taxonomy editor (b)

**Similarity Editors for Set Attributes.** Attributes that allow multiple values (either numerical or symbolic) are a powerful concept for representing weakly structured knowledge. However, the similarity calculation for such set attributes is much more complex compared with single values. This concerns the computation complexity as well as conceptional issues. In general, the semantic of the comparison of set values is extremely application specific. For example, a set of values may have a kind of "and" or a kind of "or" semantic. Moreover, the size of the query and case sets may have different influences on the similarity.

*myCBR* provides various options to configure similarity measures for set attributes. Depending on the chosen settings, the mapping between query values and case values is calculated differently. For example, one might want to match each query value with be best suited case values or vice versa. Moreover, query/case values that could not be matched to a case/query value (e.g. because the query contains more values than the case) may have a different impact on the final similarity. Once the desired mapping is determined, the final similarity computation is based on the basic similarity measures defined for the atomic values of the sets. Depending on the data type, here the previously described editors can be used.

**Script-Based and External Similarity Measures.** In order to obtain maximal flexibility, for all kind of data types two additional similarity modes are provided:

**Script:** *myCBR* includes a Jython[8] binding and corresponding editors that allow the user to write own similarity measures in an easy to learn scripting language.

[8] Jython is a Java-based scripting language with the same syntax than Python; for details see http://www.jython.org/Project/index.html

**External:** This similarity mode allows the user to call external programs (e.g. written in C/C++) for calculating similarities. This can be in particular useful, if computation intensive calculations are required or if data types not supported by Protégé are involved (e.g. images). In this case, the underlying attribute in the case representation may provide an URL to the external data source used by the external program to access the data.

**Dealing with Missing Values.** Missing attribute values (either in the query or in the cases) are always a crucial issue during the similarity computation because they prevent the computation of regular local similarities. Depending on the application domain, missing values can have quite different meanings. For example, in a product recommender system missing query attributes typically represent "don't care" statements of the customer, while missing case attributes correspond to unknown or not existing properties of the products.

In *myCBR* missing values are always represented as *special values*. The default special value is "_undefined_", however, the user is able to specify own special values additionally. In order to cover the application specific requirements, the influence of each special value on the similarity computation can be configured individually.

### 3.3 Testing of Retrieval Functionality

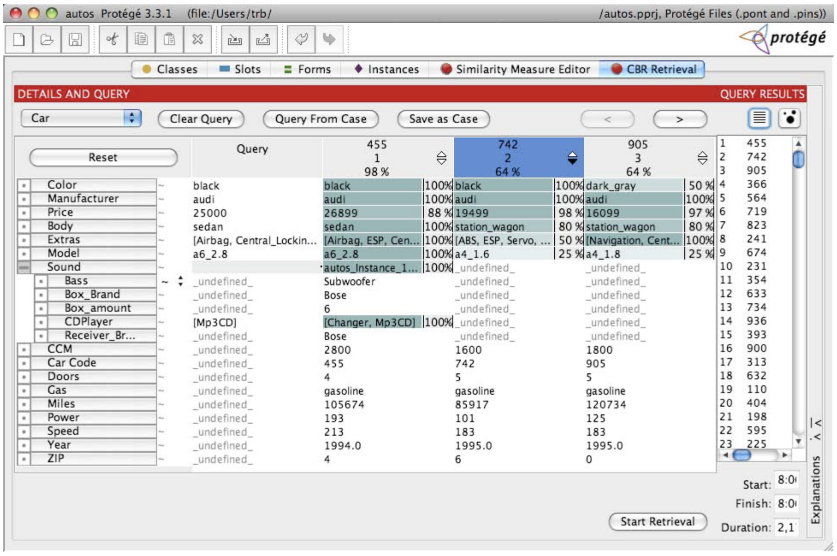
The definition of an optimal similarity measure is often a difficult and tricky task which requires repeatedly testing and fine tuning. For this purpose, *myCBR* includes a comfortable graphical user interface for performing retrievals and for analyzing the corresponding results in detail (see Figure 6). On the right hand side of the window an overview of the entire retrieval result is shown. In the center part of the GUI the query is opposed to a configurable number of retrieved cases. By providing similarity highlighting and explanation functionality (cf. Section 4), *myCBR* supports the efficient analysis of the outcome of the similarity computation.

The current version of *myCBR* provides two retrieval algorithms, a simple sequential retrieval and a basic case retrieval net [10].

### 3.4 Building a Stand-Alone Application

After having created and tested the CBR functionality using the *myCBR* Protégé plug-in, one may want to deploy that functionality in the scope of a particular application without relying on the Protégé framework. A typical use case for CBR systems are web-based applications, for example, to implement recommendation functionality in e-Commerce applications.

*myCBR* provides a Java API which allows easy integration of the retrieval functionality into arbitrary Java applications without requiring a Protégé installation. Using JSP a few lines of code are sufficient to implement a simple web-based CBR application with custom-made user interfaces. An example of such a web-based application is shown in Figure 7.



| | Query | 455<br>1<br>98 % | | 742<br>2<br>64 % | | 905<br>3<br>64 % | |
|---|---|---|---|---|---|---|---|
| Color | black | black | 100% | black | 100% | dark_gray | 50 % |
| Manufacturer | audi | audi | 100% | audi | 100% | audi | 100% |
| Price | 25000 | 26899 | 88 % | 19499 | 98 % | 16099 | 97 % |
| Body | sedan | sedan | 100% | station_wagon | 80 % | station_wagon | 80 % |
| Extras | [Airbag, Central_Lockin... | [Airbag, ESP, Cen... | 100% | [ABS, ESP, Servo, ... | 50 % | [Navigation, Cent... | 100% |
| Model | a6_2.8 | a6_2.8 | 100% | a4_1.6 | 25 % | a4_1.8 | 25 % |
| Sound | | autos_Instance_1... | 100% | _undefined_ | | _undefined_ | |
| Bass | _undefined_ | Subwoofer | | _undefined_ | | _undefined_ | |
| Box_Brand | _undefined_ | Bose | | _undefined_ | | _undefined_ | |
| Box_amount | _undefined_ | 6 | | _undefined_ | | _undefined_ | |
| CDPlayer | [Mp3CD] | [Changer, Mp3CD] | 100% | _undefined_ | | _undefined_ | |
| Receiver_Br... | _undefined_ | Bose | | _undefined_ | | _undefined_ | |
| CCM | _undefined_ | 2800 | | 1600 | | 1800 | |
| Car Code | _undefined_ | 455 | | 742 | | 905 | |
| Doors | _undefined_ | 4 | | 5 | | 5 | |
| Gas | _undefined_ | gasoline | | gasoline | | gasoline | |
| Miles | _undefined_ | 105674 | | 85917 | | 120734 | |
| Power | _undefined_ | 193 | | 101 | | 125 | |
| Speed | _undefined_ | 213 | | 183 | | 183 | |
| Year | _undefined_ | 1994.0 | | 1995.0 | | 1995.0 | |
| ZIP | _undefined_ | 4 | | 6 | | 0 | |

| | |
|---|---|
| 1 | 455 |
| 2 | 742 |
| 3 | 905 |
| 4 | 366 |
| 5 | 564 |
| 6 | 719 |
| 7 | 823 |
| 8 | 241 |
| 9 | 674 |
| 10 | 231 |
| 11 | 354 |
| 12 | 633 |
| 13 | 734 |
| 14 | 936 |
| 15 | 393 |
| 16 | 900 |
| 17 | 313 |
| 18 | 632 |
| 19 | 110 |
| 20 | 404 |
| 21 | 198 |
| 22 | 595 |
| 23 | 225 |

Start: 8:0
Finish: 8:0
Duration: 2,1

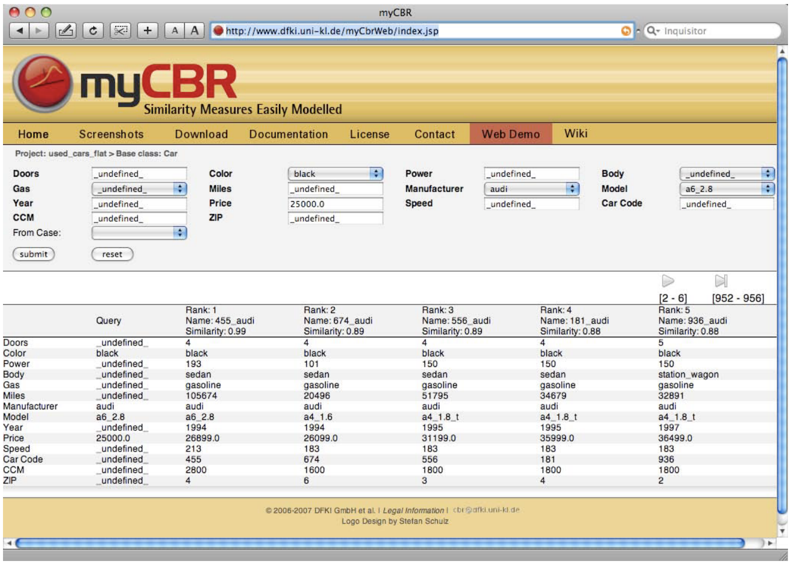
**Fig. 6.** Retrieval result with attribute values sorted in descending order of similarity values. Note the decreasing highlighting of cells corresponding to local similarity.

During the stand-alone operation of *myCBR* the XML files generated by the Protégé plug-in serve as source for obtaining the similarity model, the configuration options, and the case base. If certain maintenance operations are necessary, the XML files may be updated by using the Protégé plug-in again, or application specific modules may change the XML files directly, for example, to store new or to delete obsolete cases.

## 4 Explanation Functionality

Ease-of-use as well as approachability of any software system is improved by increasing its understandability, which in turn can be supported by appropriate explanation capabilities [11]. We follow Schank [12] in considering explanations the most common method used by humans to support understanding and their decision making. In everyday human-human interactions explanations are an important vehicle to convey information in order to understand one another. Explanations enhance the knowledge of the communication partners in such a way that they accept certain statements. They understand more, allowing them to make informed decisions.

This communication-oriented view leads to the following explanation scenario comprising three participants (Figure 8). First, the originator that is a system or an agent that provides something to be explained, e.g., the solution to some problem, a technical device, a plan, a decision etc. In our case, the originator comprises the modelling tools and the retrieval engines of *myCBR*. Second, the user who is the addressee of the explanation. Third, the explainer who presents


myCBR
http://www.dfki.uni-kl.de/myCbrWeb/index.jsp
Inquisitor
myCBR
Similarity Measures Easily Modelled
Home
Screenshots
Download
Documentation
License
Contact
Web Demo
Wiki
submit
reset
[2 - 6]
[952 - 956]
© 2006-2007 DFKI GmbH et al. | Legal Information | cbr@dfki.uni-kl.de
Logo Design by Stefan Schulz

Project: used_cars_flat > Base class: Car

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Doors | _undefined_ | Color | black | Power | _undefined_ | Body | _undefined_ |
| Gas | _undefined_ | Miles | _undefined_ | Manufacturer | audi | Model | a6_2.8 |
| Year | _undefined_ | Price | 25000.0 | Speed | _undefined_ | Car Code | _undefined_ |
| CCM | _undefined_ | ZIP | _undefined_ | | | | |
| From Case: | | | | | | | |

| | Query | Rank: 1<br>Name: 455_audi<br>Similarity: 0.99 | Rank: 2<br>Name: 674_audi<br>Similarity: 0.89 | Rank: 3<br>Name: 556_audi<br>Similarity: 0.89 | Rank: 4<br>Name: 181_audi<br>Similarity: 0.88 | Rank: 5<br>Name: 936_audi<br>Similarity: 0.88 |
|---|---|---|---|---|---|---|
| Doors | _undefined_ | 4 | 4 | 4 | 4 | 5 |
| Color | black | black | black | black | black | black |
| Power | _undefined_ | 193 | 101 | 150 | 150 | 150 |
| Body | _undefined_ | sedan | sedan | sedan | sedan | station_wagon |
| Gas | _undefined_ | gasoline | gasoline | gasoline | gasoline | gasoline |
| Miles | _undefined_ | 105674 | 20496 | 51795 | 34679 | 32891 |
| Manufacturer | audi | audi | audi | audi | audi | audi |
| Model | a6_2.8 | a6_2.8 | a4_1.6 | a4_1.8_t | a4_1.8_t | a4_1.8_t |
| Year | _undefined_ | 1994 | 1994 | 1995 | 1995 | 1997 |
| Price | 25000.0 | 26899.0 | 26099.0 | 31199.0 | 35999.0 | 36499.0 |
| Speed | _undefined_ | 213 | 183 | 183 | 183 | 183 |
| Car Code | _undefined_ | 455 | 674 | 556 | 181 | 936 |
| CCM | _undefined_ | 2800 | 1600 | 1800 | 1800 | 1800 |
| ZIP | _undefined_ | 4 | 6 | 3 | 4 | 2 |

**Fig. 7.** A *myCBR* web demo application (see also http://www.myCBR-project.net)

the explanation to the user. This agent is interested in transferring the intention of the originator to the user as correct as possible. The explainer chooses the kind of the explanation [13] and is responsible for the computational aspects as well as for organising a dialog if needed. Originator and explainer need to work together rather tightly to improve the communication with the user. The originator needs to provide the appropriate information in order to allow the explainer constructing appropriate explanations.

In order to support the communication scenario described above, *myCBR* provides two general kinds of explanations: forward and backward explanations. *Forward explanations* explain indirectly, presenting different ways of optimizing a given result and opening up possibilities for the exploratory use of a device or application. *Backward explanations* explain the results of a process and how they were generated. Details and technical aspects of how the explanation component works are available in [14].

In order to increase transparency of and trust in the retrieval process [15], *myCBR* creates an explanation object for each case during similarity calculation. This tree-like data structure stores global and local similarity values as comments for each attribute. These retrieval details are presented to the user in the retrieval GUI (Figure 6) either as tool tips or in abbreviated form along with the case's attribute value, e.g., the price of car offer 455 (26,899) is 88% similar to the requested car price (25,000). Another valuable feature is the option to find the most similar cases with respect to a single attribute by simply clicking on the attribute name (row head). In attribute rich cases one might also want to sort the local similarity values of one case in ascending or descending order. This can simply be achieved by clicking on the respective case name (column head).

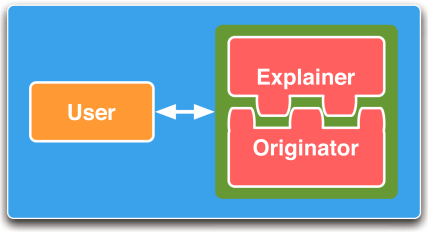


**Fig. 8.** Participants in explanation scenario

While developing a CBR system an important question is whether a similarity measure leads to the appropriate cases for a given query. Forward explanations (not depicted in the screenshots) help predicting the behavior of the system during modeling time and explain the interdependencies between the similarity measure and the case base. For this, a central explanation component analyzes the case base and gathers statistical information. The distribution of values in the case base can already be quite helpful and may reveal parts of similarity measures that are in fact never used (assuming that the case base covers most of possible queries). Or they reveal missing border cases, which is important for exception treatment.

## 5 Conclusion and Outlook

In this paper we have presented a novel, freely available CBR tool that supports rapid prototyping of advanced retrieval-based CBR applications. By providing powerful but still easy-to-use model generation, data import, similarity modeling, explanation, and testing functionality, *myCBR* enables even CBR novices to rapidly create their first CBR applications.

Nevertheless, at the same time the support of object-oriented case representations, advanced similarity editors, various configuration options, integration of a scripting language, and the possibility to call custom-made external modules ensures very high flexibility in order to fit also the requirements of expert users and complex application domains.

In focusing on the similarity-based retrieval step, *myCBR* differs from the JColibri system which aims to cover the entire CBR cycle in a flexible way. However, JColibri does not provide comparable graphical user interfaces for defining knowledge-intensive similarity measures but requires to program them by hand. In the future, an integration of both Open Source systems in order to benefit of the advantages of both might be worth to be considered.

*myCBR* is still an ongoing project and several extensions of the system are already planned or are even already under development. In order to facilitate the work with more structured, object-oriented case representations and to improve the interoperability with existing IT infrastructure, one of the next steps is the implementation of an interface for accessing relational database management systems. This interface will provide an advanced data importer which enables automatic generation of object-oriented case representations similar to the CSV importer. Moreover, this interface will allow to retrieve cases directly from a database instead of relying on XML files for storing case bases.

Another planned extension is the implementation of a rule engine for providing adaptation and completion rules [16]. This would make *myCBR* to a full-fledged CBR system beyond pure similarity-based retrieval. Last but not least, we plan to integrate our approaches to automatically learn similarity measures based on given user/application feedback [17].

We also encourage other researchers to try out *myCBR* in their own research and teaching projects and to contribute to the further development by implementing their own extensions and experimental modules.

## Acknowledgements

The authors would like to thank Daniel Bahls, Andreas Rumpf, and Laura Zilles for their great implementation work and all the valuable discussions during the development of the *myCBR* system. This work was partially funded by the federal state Rhineland-Palatinate under the project ADIB (Adaptive Provision of Information).

## References

1. Bello-Tomás, J., González-Calero, P.A., Díaz-Agudo, B.: JColibri: An Object-Oriented Framework for Building CBR Systems. In: Proceedings of the 7th European Conference on Case-Based Reasoning. Springer, Heidelberg (2004)
2. Schulz, S.: CBR-Works - A State-of-the-Art Shell for Case-Based Application Building. In: Proceedings of the 7th German Workshop on Case-Based Reasoning (GWCBR 1999) (1999)
3. Aamodt, A., Plaza, E.: Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications 7(1), 39–59 (1994)
4. Bridge, D., Göker, M.H., McGinty, L., Smyth, B.: Case-based recommender systems. Knowledge Engineering Review 20(3) (2006)
5. Stahl, A.: Learning of Knowledge-Intensive Similarity Measures in Case-Based Reasoning, Dissertation.de, vol. 986 (2004)
6. Richter, M.M.: The Knowledge Contained in Similarity Measures. In: ICCBR 1995 (1995)
7. Gennari, J.H., Musen, M.A., Fergerson, R.W., Grosso, W.E., Crubézy, M., Eriksson, H., Noy, N.F., Tu, S.W.: The evolution of Protégé an environment for knowledge-based systems development. J. Hum.-Comput. Stud. 58(1), 89–123 (2003)
8. Bergmann, R., Stahl, A.: Similarity Measures for Object-Oriented Case Representations. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488. Springer, Heidelberg (1998)
9. Bergmann, R.: On the Use of Taxonomies for Representing Case Features and Local Similarity Measures. In: Proceedings of the 6th German Workshop on Case-Based Reasoning (GWCBR 1998) (1998)

10. Lenz, M.: Case Retrieval Nets as a Model for Building Flexible Information Systems. Ph.D. Thesis, Humboldt University Berlin (1999)
11. Roth-Berghofer, T.R.: Explanations and Case-Based Reasoning: Foundational issues. In: Funk, P., González-Calero, P.A. (eds.) Advances in Case-Based Reasoning, pp. 389–403. Springer, Heidelberg (2004)
12. Schank, R.C.: Explanation Patterns: Understanding Mechanically and Creatively. Lawrence Erlbaum Associates, Hillsdale (1986)
13. Roth-Berghofer, T., Cassens, J., Sørmo, F.: Goals and kinds of explanations in case-based reasoning. In: Althoff, K.D., Dengel, A., Bergmann, R., Nick, M., Roth-Berghofer, T. (eds.) WM 2005: Professional Knowledge Management, Kaiserslautern, Germany, DFKI GmbH, pp. 264–268 (2005)
14. Bahls, D.: Explanation support for the case-based reasoning tool MYCBR. Project thesis, University of Kaiserslautern (2008)
15. Roth-Berghofer, T.R., Cassens, J.: Mapping goals and kinds of explanations to the knowledge containers of case-based reasoning systems. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 451–464. Springer, Heidelberg (2005)
16. Bergmann, R., Wilke, W., Vollrath, I., Wess, S.: Integrating General Knowledge with Object-Oriented Case Representation and Reasoning. In: Proceedings of the 4th German Workshop on Case-Based Reasoning (GWCBR 1996) (1996)
17. Stahl, A., Gabel, T.: Using Evolution Programs to Learn Local Similarity Measures. In: Proceedings of the 5th International Conference on CBR. Springer, Heidelberg (2003)