Rapid Prototyping of CBR Applications

with the Open Source Tool myCBR

Armin Stahl, Thomas R. Roth-Berghofer

In this paper a novel, freely available tool for rapid prototyping of Case-based Reasoning (CBR) applications is presented. 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 applications.

1 Introduction

Compared with other AI approaches, CBR allows reducing the knowledge acquisition and representation effort significantly. Nevertheless, implementing a CBR application from scratch still 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 and are connected with high licensing costs. 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 [3].

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. Moreover, the tool should be easily extendable in order to facilitate the experimental evaluation of novel algorithms and research results.

The current version of *myCBR* still focuses on the similarity-based retrieval step of the CBR cycle [1], because this is still the core functionality of most CBR applications. While early CBR systems often were based on simple distance metrics, today many CBR applications make use of highly sophisticated, knowledge-intensive similarity measures [8]. The major goal of *myCBR* is to minimize the effort for building CBR applications that require such 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 this article we give an overview of the basic concept and system architecture of *myCBR* and we describe how rapid prototyping of CBR applications is supported by *myCBR*. A more detailed description of myCBR can be found in [9].

2 The myCBR Architecture

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*. 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[3] with its easily extendable plug-and-play environment.

In order to avoid a reinvention of the wheel, we have chosen Protégé as the modeling platform for *myCBR*. This brings two main advantages: First, the effort for implementing data structures and user interfaces for representing the vocabulary and the case knowledge can be avoided. Second, it allows to add CBR functionality to existing Protégé applications with minimal effort.

The basic architecture of *myCBR* is illustrated in Figure 1. During the development phase of a CBR application, *myCBR* runs as a plug-in within Protégé. This plug-in consists of several modules. The *modeling tools* extend the existing functionality of Protégé for creating domain models and case instances, and add the missing functionality for defining similarity measures. The *retrieval GUI* provides powerful features

[1] http://gaia.fdi.ucm.es/projects/jcolibri
[2] http://www.mycbr-project.net
[3] http://protege.stanford.edu/

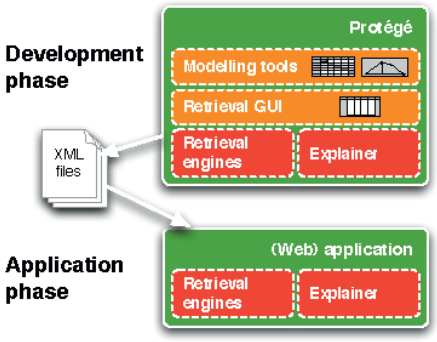

Figure 1: The system architecture of *myCBR*

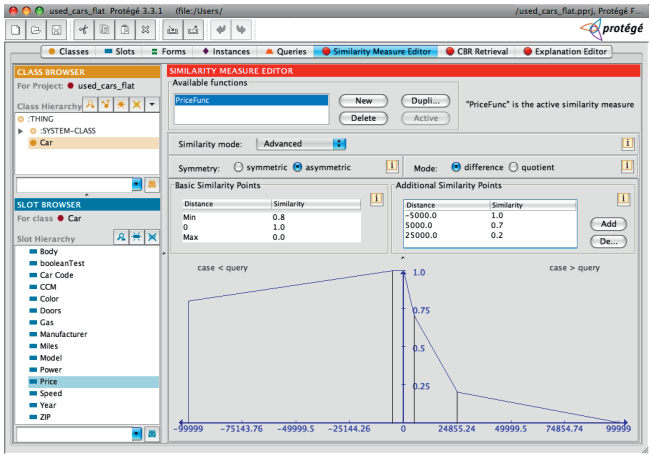

Figure 2: Modeling of similarity measures with myCBR

for analyzing the quality of the defined similarity measures. Moreover, it can also serve as the user interface of first prototypical applications. For executing the similarity-based retrieval, different *retrieval engines* are provided. A dedicated explanation component provides modelling support information as well as explanations of retrieval results for quicker roundtrips of designing and testing.

After installing and activating the *myCBR* plug-in, the user interface of Protégé is extended with additional tabs to access the *myCBR* modules. 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[4]-based web applications.

3 Developing *myCBR* prototypes

In this section we describe how *myCBR* supports rapid prototyping of CBR applications.

3.1 CSV Data Import and Automatic Model Generation

The foundation of every CBR application is an accurate case representation. 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 module for importing data from CSV[5] files. 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.

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. A commonly used global similarity measure is, for example, a weighted sum of local similarities. But *myCBR* is also able to deal with more structured, object-oriented representations and supports suited global similarity measures [5].

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.

For numerical attributes, the similarity computation is typically based on a mapping between the distance d of the two values q_i and c_i to be compared and the desired similarity value, i.e. $sim_i(q_i, c_i) = f(d(q_i, c_i))$, where sim_i denotes the local similarity measure of attribute i. This means, the similarity modeling focuses on the definition of an accurate mapping function f for a given distance function d [8]. For modeling the mapping function f, *myCBR* provides comfortable graphical user interfaces (see Figure 2).

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 the similarities of all pairwise value combinations are enumerated explicitly. However, for larger value sets the definition of similarity tables remains a time consuming and annoying task. Therefore, *myCBR* supports more comfortable approaches based on either a total *order* on the symbols or by arranging symbols in a *taxonomy*. Once the order or taxonomy and its application specific meaning is specified, it can be deployed to perform

[4] Java Server Pages
[5] Comma Separated Values

automatic similarity calculations (for details of this approach see [4]).

Although textual CBR is not the main focus of *myCBR* it also provides flexible similarity measures for string processing like partial or trigram matching, or regular expressions.

Attributes that allow multiple values (either numerical or symbolic) are a powerful concept for representing weakly structured knowledge. *myCBR* also provides various options to configure similarity measures for set attributes.

In order to obtain maximal flexibility, for all kind of data types two additional similarity modes are provided: The *script* mode provides a Jython[6] binding and corresponding editors that allow the user to write own similarity measures in an easy to learn scripting language. The *external 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).

3.3 Testing of Retrieval Functionality

The definition of an optimal similarity measure is often a difficult and tricky task that 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. By providing similarity highlighting and explanation functionality, *myCBR* supports the efficient analysis of the outcome of the similarity computation.

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.

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 [6]. We follow Schank [7] 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. First, the originator that is a system or an agent that provides something to be explained, e.g., the solution to some problem. 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 the explanation to the user. This agent is interested in transferring the intention of the originator to the user as correct as possible.

For supporting the communication scenario described above *myCBR* provides two general kinds of explanations. *Forward explanations* explain indirectly, presenting different ways of optimizing a given result and are 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 [2].

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 create their first CBR applications rapidly.

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 more complex application domains.

myCBR is still an ongoing project and several extensions of the system like a database connection and a rule engine are already planned or are already under development. 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.

References

[1] A. Aamodt and E. Plaza. Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59, 1994.

[2] Daniel Bahls. Explanation support for the case-based reasoning tool mycbr. Project thesis, University of Kaiserslautern, 2008.

[3] J.J. Bello-Tomás, Pedro A. González-Calero, and Belén Díaz-Agudo. JColibri: An Object-Oriented Framework for Building CBR Systems. In *Proceedings of the 7th European Conference on Case-Based Reasoning*. Springer, 2004.

[6] for details see http://www.jython.org/Project/index.html

[4] R. Bergmann. On the Use of Taxonomies for Representing Case Features and Local Similarity Measures. In *Proceedings of the 6th German Workshop on Case-Based Reasoning*, 1998.

[5] R. Bergmann and A. Stahl. Similarity Measures for Object-Oriented Case Representations. In *Proceedings of the 4th European Workshop on Case-Based Reasoning*. Springer, 1998.

[6] Thomas R. Roth-Berghofer. Explanations and Case-Based Reasoning: Foundational issues. In *Proceedings of the 7th European Conference on Case-Based Reasoning*. Springer-Verlag, 2004.

[7] Roger C. Schank. *Explanation Patterns: Understanding Mechanically and Creatively*. Lawrence Erlbaum Ass., Hillsdale, NJ, 1986.

[8] A. Stahl. *Learning of Knowledge-Intensive Similarity Measures in Case-Based Reasoning*, volume 986. dissertation.de, 2004.

[9] A. Stahl and T. Roth-Berghofer. Rapid Prototyping of CBR Applications with the Open Source Tool myCBR. In *Proc. of the 9th European Conference on CBR*. Springer, 2008.

Contact

Dr. Armin Stahl
German Research Center
for Artificial Intelligence DFKI GmbH
Image Understanding and Pattern Recognition Department
Armin.Stahl@dfki.de

Dr. Thomas Roth-Berghofer
German Research Center
for Artificial Intelligence DFKI GmbH
Knowledge Management Department
thomas.roth-berghofer@dfki.de

Armin Stahl studied Computer Science at the University of Kaiserslautern with focus on artificial intelligence. From 2000 to 2003 he worked as a researcher in the work group "Artificial Intelligence - Knowledge-Based Systems" directed by Prof. Michael M. Richter where he received his Ph.D. for his research on Learning Knowledge-intensive Similarity Measures in CBR in 2003. Since 2004 he is the deputy director of the Image Understanding and Pattern Recognition Department of DFKI.

Thomas R. Roth-Berghofer studied Computer Science at the University of Kaiserslautern. From 1998 to 2002 he worked at tec:inno (now empolis) GmbH, a university spin-off specialising in bringing CBR to market. He received his Ph.D. for his work on knowledge maintenance of CBR systems from the University of Kaiserslautern where he gives also lectures on "Case-Based Reasoning" together with Armin Stahl. He is presently Senior Researcher at the Knowledge Management Research Department of DFKI where his research focusses on personal information systems with explanation capabilities.