CBR-Works
A State-of-the-Art Shell for
Case-Based Application Building

Stefan Schulz
TECINNO GmbH, Sauerwiesen 2,
D-67661 Kaiserslautern, Germany
schulz@tecinno.com

Abstract. Nowadays, a proper tool for Case-Based Reasoning has to fulfill a wide range of tasks beyond simple retrieval. This paper gives a brief overview of the abilities and features of the tool CBR-Works which provides support for the design process of a Case-Based application as well as for maintenance and retrieval. CBR-Works also provides the ability to reuse existing data from common database systems and may act as server for distributed access to a case base, including retrieval and case base management.

1 Introduction

Case-Based Reasoning (CBR) becomes more and more popular for companies, improving and enhancing their customer and sales support by introducing "intelligent applications" [5]. Using a Case-Based application not only provides stored product catalogs or experience knowledge (the *cases*) to customers of a company. But also, by capturing problems and solutions a corporate memory is built, so the knowledge is no longer distributed in the workers minds but accessible to everyone in a company.

Besides collecting cases, applying Case-Based Reasoning necessitates a CBR-Tool supporting retrieval of matching cases as well as modeling and maintaining of the case base. Companies store information about their products in common database systems. Hence, as the amount of stored data is rather large, the CBR-Tool's ability of easy (re)using those information is important.

Another fundamental characteristic of a CBR-Tool is to cover the complete cycle of Case-Based Reasoning ([1], [4]), i.e., retrieving cases similar to a user's specification, reusing a retrieved case as proposed solution, testing a solved case for success during the revisioning process, and retaining a new solution given in form of the revised case by including the experiences (the case) into the existing case base.

CBR-Works is a shell for Case-Based application building. Besides the retrieval of cases, it supports modeling the cases' structure and maintaining the case base. Its consultation mechanism also covers the whole CBR-Cycle from retrieving to revising.

Though CBR-Works is designed as a complete environment, it may also act as a CBR-Server for several clients by the use of CQL (Case Query Language [9]). Last but not least, CBR-Works offers an open interface to build a Case-Based application from existing data stored in common database systems.

This paper gives a brief overview of the abilities and features of CBR-Works. It will introduce the tool's elements that are used for building an application. To illustrate the building process, a simplified PC-Domain is used as depicted in fig. 1. This example will be used in the following chapters.

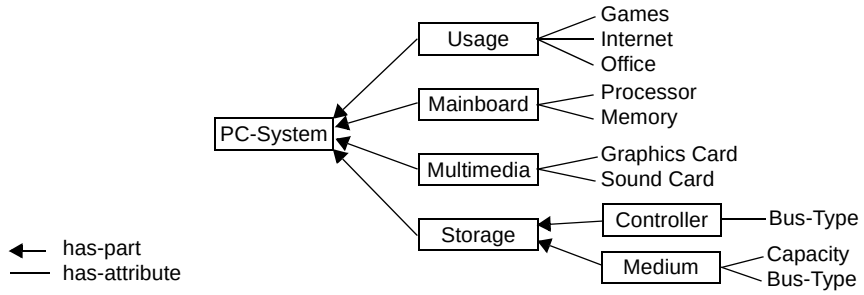

Fig. 1. Structure of a simplified PC-Domain's case

The following two sections describe the common elements used for building a case base in CBR-Works. Section 3 gives a concise description on maintenance in CBR-Works. In chapter 4 the interface for reusing data is tersely discussed. This is followed by an overview on how to consult a case base in section 5. Finally, perspectives are given in chapter 6 on further enhancements of CBR-Works.

2 Structure Modeling

CBR-Works is suited for intelligent solutions in a variety of domains and environments. Its graphical editors support the user to design complex knowledge models. An object-oriented approach (see [6], [7]) is used in CBR-Works to design the underlying structure of cases. This structure can be edited and maintained in an easy and intuitive way.

2.1 Concepts

In CBR-Works, *concepts* define the structure of the cases. They are defined in hierarchy similar to a class-model hierarchy including inheritance. Each *concept* consists of attributes which can be either atomic (defined by a *type)* or complex (has-part relationship to another *concept*).

For retrieval purposes, attributes have three additional, functional properties: one for defining its weight, i.e., its importance in respect to the other attributes of the concept, a property for defining whether an attribute is discriminant for retrieval or will be

ignored, and another property defining if an attribute is mandatory for a case to be valid. Moreover, for every attribute a question and an annotation may be given that can be used by clients when asking for the value and to refer to further information about an attribute.

In fig. 1 each rectangle may be seen as a concept. For example, Storage consists of the two complex attributes Controller and Medium, and again the latter consists of the two atomic attributes Capacity and Bus-Type.

Concept Similarity. Beside attributes, the type of similarity can be specified for every concept. The concept's similarity consists of two parts: the similarity of a concept's contents (*contents-based similarity*) and the similarity between concepts (*structure-based similarity*) (see [2] for detailed information on similarities).

The contents-based similarity of a concept is computed based on the attributes defined in the concept. It may be one of the following:

- Average: All attribute similarities contribute to the contents-based similarity by computing their average.
- Euclidean: Geometric interpretation of the contents-based similarity (distance between two concepts, based on its contents).
- Minimum: The lowest attribute similarity defines the contents-based similarity.
- Maximum: The highest attribute similarity defines the contents-based similarity.

An example for a contents-based similarity is given in fig. 2. Here, the similarity between the usage parts of two PC-Domain cases is computed using Average. The numbers are the computed similarities between two objects which are connected by a corresponding arc. The upper similarity computes as average of the lower ones.

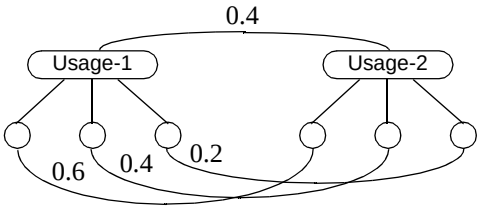

Fig. 2. Example of contents-based similarity using Average

The structure-based similarity defines similarities between concepts independent of their contents. Inside a concept-hierarchy, the similarity of concepts to each other may be explicitly or implicitly defined by using a taxonomic view of the hierarchy.

In the PC-Domain a concept-hierarchy could be defined like in fig. 3a. Assuming the initial taxonomic view of the hierarchy as base for the structure-based similarity, it computes to $\frac{\text{level of common father}}{\text{number of levels}}$. An example for a two-level taxonomy is shown in fig. 3b.

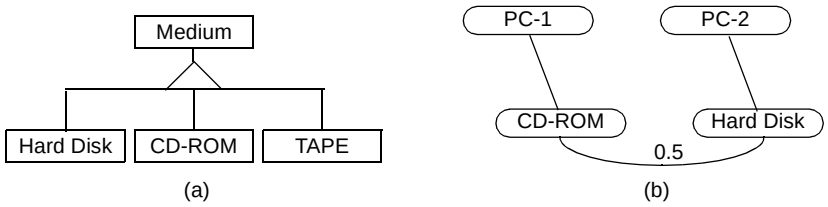

Fig. 3. Example for structure-based similarity: a) concept-hierarchy for Medium b) structure-based similarity between two PC-Domain cases where Medium is the common father

The concept's similarity computes as a weighted sum of structure-based and contents-based similarities.

Rules. Additionally, rules may be specified for each concept, either being *completion* or *adaptation* rules. Completion rules apply to cases of a case base as well as to a query whenever a new value is given for an attribute. If some attribute values depend on each other, completion rules ease handling by automatically setting appropriate values. Adaptation rules get activated only after retrieval and they are used to combine attribute values of the query and retrieved cases and to apply the result to a target case. That way, slightly modified cases are created which may fit the customers need better than the retrieved case.

Each rule, for adaptation as well as completion, consists of two parts: a *condition* part and a *conclusion* part. The condition part defines a conjunction of conditions. A condition may either be a predicate or a simple calculation over attributes (of the according concept), constants (defined using concepts or types), or local variables (computed by previous conditions). The conclusion part consists of actions being executed if all conditions of the condition part are fulfilled. An action may be an assignment of values to attributes (atomic as well as complex), a command to open a notifier (e.g., to report inconsistencies due to a given value), or changes to retrieval-influencing values (e.g., filters and weights) (see [3], [8]).

For example, to keep consistency for the Storage component of a PC-System, a completion rule may be defined to ensure that a Medium will fit to a specified Controller. If a Medium gets defined having a Bus-Type different to an already specified Controller, a notifier will open to inform the customer about this inconsistency. More complex, an adaptation rule may be defined choosing a, e.g, different, fitting Controller replacing the previously specified one.

2.2 Types

Similar to concepts, types are defined hierarchically. New types are defined by building subtypes of the existing elementary types shown in table 1. They differ in their usability: a type may be used *immediate* or *derived*. While *immediate* types cover the

whole range of possible values of a type, *derived* types get restricted in their range by defining an enumeration of elements of its elementary type or, in case of numeric types, by specifying an interval.

Table 1. Elementary Types in CBR-Works

Type	Usability	Type	Usability
Integer	immediate and derived	String	immediate and derived
Real	immediate and derived	Symbol	immediate and derived
Date	immediate and derived	Ordered Symbol	derived only
Time	immediate and derived	Taxonomy	derived only
Boolean	immediate only	Reference	derived only

Additional to the type *Symbol*, *Ordered Symbol* provides a total and *Taxonomy* a partial order over a given enumeration of values. For example, Hard Disk being defined using *Taxonomy* introduces a partial order of the values compatibility regarding Bus-Types as shown in fig. 4.

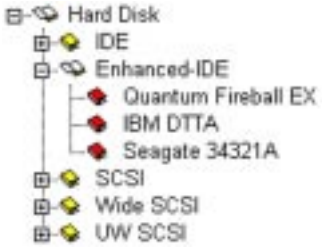

Fig. 4. Taxonomy over selected Processors

Furthermore, *constructional types* are available for defining intervals and sets using defined, elementary types. Here, intervals are restricted to ordered types where sets may be defined over any elementary type or one of its derivatives (see table 2 for restrictions).

Table 2. Constructional Types in CBR-Works

Type	Value-Type Restrictions
Set	All but Boolean
Interval	Ordered Types (e.g., Ordered Symbol, Integer, Real)

Type Similarity. For each type derived from elementary types, similarities may be defined describing major parts of the experts knowledge which is necessary for intelligent retrieval. The definition ranges from value-to-value specifications in form of a table over special, type-depending similarities (e.g., for string types) to functional specification by graphs [2]. Furthermore, an interface is given to define a programmatic similarity for any derived type. An example of functional similarity is given in

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.