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.

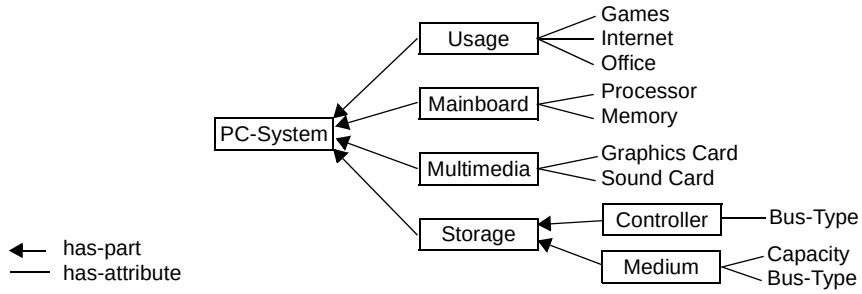

Fig. 1. Structure of a simplified PC-Domain's case

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.

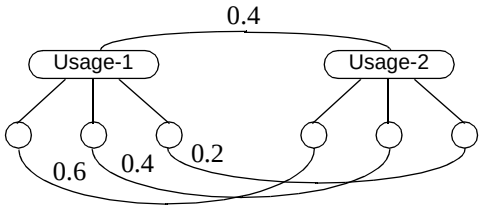

Fig. 2. Example of contents-based similarity using Average

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.

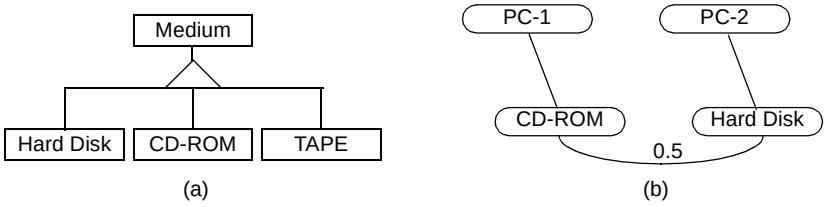

Fig. 3. Example for structure-based similarity: a) concept-hierarchy for Medium b) structure-based similarity between two PC-Domain cases where Medium is the common father

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.

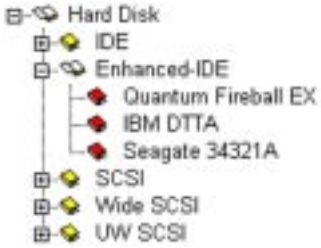

Fig. 4. Taxonomy over selected Processors

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

fig. 5 regarding a customers “feeling of an acceptable price” being different in a retrieved case to a specified value in the query. A higher price only is accepted up to a specific limit quickly dropping the higher it is. The situation is similar offering products with lower prices, as a customer usually thinks of lower quality by a lower price once the negative limit is passed.

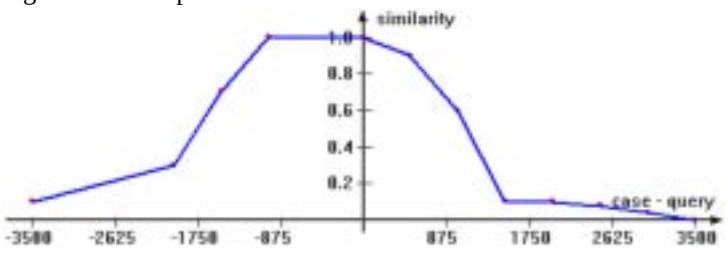

Fig. 5. Example of a similarity-function for the price of a computer

For derivatives of constructional types, predefined similarity functions are given based on intersection and inclusion of sets or intervals.

In CBR-Works, it is possible to define more than one similarity for each type as the decision which similarity to use may depend on values selected for retrieval. This decision may be formulated using completion rules for concepts.

3 Case Base Building and Maintenance

The heart of a CBR-System is the case base containing the active knowledge of the domain to be represented. Each case’s structure is defined by the underlying concept and its data represents exactly one information entity.

Cases in the Case Base. In CBR-Works, the case base consists of a number of virtual case bases each of which is founded on one of the concepts being marked as case-concept, i.e., concepts which are specified for being the structure of cases. These virtual case bases may not be seen stand-alone, but the complete set of virtual case bases is united into the CBR-Works’ case base.

In the PC-Domain, several virtual case bases may be useful, e.g., not only storing complete PC-Systems as cases but also monitor exchangeable components like Hard Disk and Mainboard cases. Hence, PC-System cases having the same Mainboard refer to the same case instead of having the same data twice in the case base (see fig. 6). As a side-effect, the effort on keeping the consistency of the case base according to changes in the specification of referred information is reduced.

A case in CBR-Works has four possible states: *unconfirmed*, *confirmed*, *protected*, and *obsolete*. Usually, new cases become *unconfirmed* being unrevised or uncomplete cases not valid for retrieval. Revised cases become either *confirmed* which allows for retrieval or *protected* which additionally protects the case from changes. Old cases, no longer valid for retrieval but probably useful for further statistics, become *obsolete*.

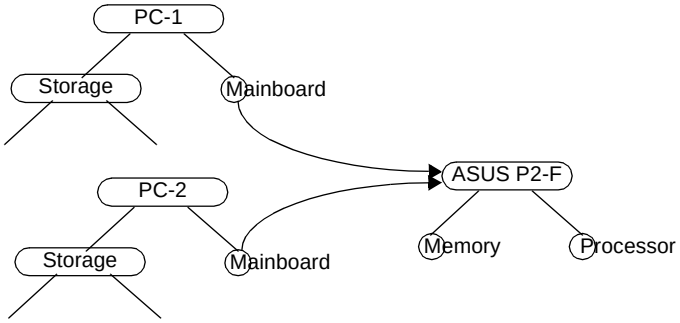

Fig. 6. Example for cases of multiple, virtual case bases. The Storage of each PC-System is defined as complex attribute belonging to the PC-System case while the Mainboard is defined as reference pointing to the according Mainboard case

Case Base Maintenance. Important for a consistent case base is the maintenance of its cases concerning validity of values and changes to the underlying model of the domain.

Therefore, CBR-Works provides several mechanisms ensuring that each case which is confirmed or protected to be valid regarding modeled type-ranges after inserting or modifying a case in the case base as well as changing the structure of the model, e.g., changing the range of a type. In the latter and similar operations, appropriate actions to the case base are selectable, being necessary to keep consistency and prevent data loss due to changes in the model, e.g., remapping values of cases when changing the type of attributes.

4 Reusing Data

Building a CBR-System from scratch is necessary and appropriate for domains that are not available in electronic form. For information being stored in, e.g., a database, a CBR-Tool must be able to reuse such data rather than having the user to remodel the domain and manually add all information to the case base.

CBR-Works supports connections to electronic information via the open database connection (ODBC) system. Hence, any source (e.g., sheet or database) which contains the domain-data can be connected to CBR-Works for import of structure and data to build up the CBR-System. Here, concepts are build from tables or views being defined by the source, and types may be generated from the contents of each column. Relations between tables are modeled by either using references or aggregation. In case of aggregation, the information given in related tables becomes part of a case. Using references necessitates the referenced concept also being marked as concept for building cases.

After building the domain model that bases on the source information, the cases are

imported into the case base using the same interface served by ODBC. Each row of a table becomes one case, including aggregated concepts built of rows from related tables and references to cases built from related table-rows (see fig. 7).

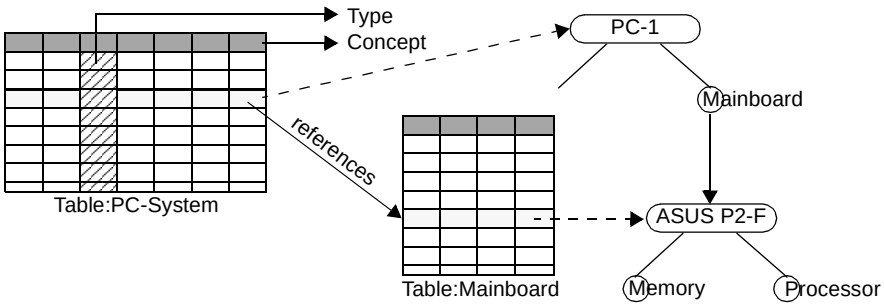

Fig. 7. Creation of a PC-System case out of a database

5 Consulting the Case Base

For querying the case base and retrieving cases from it, CBR-Works offers several interfaces for console using as well as for clients using CBR-Works as server. The so called consultation of the case base covers the whole Case-Based Reasoning Cycl. Not only providing retrieval-mechanisms but also the possibility to revise and to retain suggested or confirmed solutions in form of cases. In CBR-Works, the revision step also includes adaptation of cases using the appropriate rules.

5.1 Common Consultation

Generally, consultation happens either by using firsthand access to CBR-Works as a CBR-Console or by remotely accessing the case base with CBR-Works acting as a Server.

In addition to requested values, a query consists of further information like filters and weights for attribute-values, being hard constraints in opposition to the rather soft constraints provided by similarity-measures. Other additional information is: a threshold to lay down the minimal similarity a case may have to be valid as solution, options for completion of the query's values and adaptation of retrieved cases, and options for defining the virtual case bases to be considered.

5.2 Strategic Questioning

Besides the common consultation, strategic questioning of attribute-values interactively leads to suggested solutions. Here, algorithmic mechanisms ask for values in order to quickly reduce the number of possible solutions.

The predefined strategy of *information gain* operates on retrieved cases and com-

putes the gain of information for every undefined attribute of the query according to its ability to partition the space of solutions.

A second strategy bases on modeled *importance ranking*, where the modeler determines the order of selected questions. Questions not explicitly ordered by this ranking are handled using the strategy of information gain which is normalized to the range between zero and the lowest ranking given.

6 Perspectives

By now, CBR-Works can be seen as a CBR-Shell providing all necessary tools to model, maintain and consult a case base. Moreover, CBR-Works is able to reuse information already stored in electronic form. For simple representation of the added value and power brought in by CBR, an integrated WWW-Server with adapted generic interface supports online retrieval without additional programming.

Forthcoming versions of CBR-Works may be enhanced by including additional modules, e.g., a configuration component, which allows for adapting and combining retrieved cases automatically and interactively with a consultant until his requirements are fulfilled. This is highly interesting for the domain of electronic commerce, where products often consist of exchangeable components and therefore may be suited to a customers need. In the PC-Domain, configuration would allow to build new cases, i.e., PC-Systems, communicating with a customer as if it were a salesclerk.

In some cases, strategic questioning, as been described previously, is not adequate to lead a customer to appropriate products. Both mechanisms given by CBR-Works only allow questioning for reachable attributes. Sometimes, it might be practical to ask for information being defined lower in the hierarchy of the case structure and automatically fill in complex attributes which build the path to the questioned attribute. Furthermore, mechanisms like decision trees may be needed to establish a suitable process modeling which, also, eases to set up a proper query.

Other than enhancing CBR-Works, future approaches for building a successor to the current monolithic CBR-Shell may lead to a smart and slim solution. It will consist of various services (in its center a retrieval kernel module) using a database as storage system. This not only allows import, but direct access to existing information. That way, case base and data source automatically stay consistent. In contrast to the current mechanism, where new information, either in source or in case base, have to be updated manually by the operator.

Another aspect of this approach on slim solutions is to simplify the addition of CBR-Technology to existing software, e.g., internet shops, in form of a stand-alone library which is built for a specific domain. As a result, not only thin clients but also thin servers will be available without carrying all CBR-System information but those needed for consultation and management of the case base.

7 References

1. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches, AICom - Artificial Intelligence Communications, IOS Press, Vol. 7: 1 (March 1994) 39-59
2. Bergmann, R., Stahl, A.: Similarity Measures for Object-Oriented Case Representations, Proceedings of the European Workshop on Case-Based Reasoning, EWCBR'98
3. Bergmann, R., Wess, S., Traphöner, R., Breen, S.: Using Background Knowledge in the Integrated System: Specification and Approach, ESPRIT project 6322, Deliverable, Kaiserslautern (1994)
4. Kolodner, J.: Case-Based Reasoning, Morgan Kaufmann Publishers, San Mateo (1993)
5. Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.): Case-Based Reasoning Technology, From Foundations to Applications, Springer-Verlag, Berlin/Heidelberg (1998)
6. Wess, S.: Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik, Ph.D. Dissertation, University of Kaiserslautern (1995)
7. Wilke, W.: Knowledge Management for Intelligent Sales Support in Electronic Commerce, Ph.D. Dissertation, University of Kaiserslautern (1998)
8. CBR-Works 3 - Reference Manual, TecInno GmbH, Kaiserslautern (1999)
9. Introduction to the Case-Query-Language, TecInno GmbH, Kaiserslautern (1998)