Brown, M., Watson, I. &Filer, N. (1995). Separating the Cases from the Data; Towards More Flexible Case-Based Reasoning. In, Case-Based Reasoning Research and Development Veloso, M., & Aamodt, A. (Eds.) Lecture Notes in Artificial Intelligence 1010 Springer-Verlag Berlin.

# Separating the Cases from the Data; Towards More Flexible Case-Based Reasoning

**Mike Brown[✞], Ian Watson[✧] and Nick Filer[✞]**

[✞] *Department of Computer Science, University of Manchester, Manchester, UK*
*michaelb@computer-science.manchester.ac.uk*
*nick@computer-sciences.manchester.ac.uk*

[✧] *AI-CBR, University of Salford, Salford, UK*
*ian@ai-cbr.org*

**Abstract**

The number of successful, small-scale and purpose-built applications of CBR is growing rapidly. However, CBR has so far not been widely used as a methodology for reusing the large-scale data repositories typically maintained by a corporation. To facilitate this, cases must no longer be considered as concretely represented at the data level, but as virtual views of the underlying data. This paper argues that the basic requirement to support virtual cases are mapping functions between different data representations. It is argued that the use of mapping functions can increase flexibility in a number of ways. Multiple CBR applications can exploit a single data repository. Similarly, a single case representation can span multiple data repositories. Support for communication between different CBR applications as well as the evolution of case representation within a single application are also catered for by the same methodology. The paper concludes with a brief description of existing technology for support of the development and use of mapping functions.

**Keywords**: virtual cases, indirect retrieval, case-based reasoning

# Separating the Cases from the Data; Towards More Flexible Case-Based Reasoning

**Mike Brown✞, Ian Watson✧ and Nick Filer✞**

✞ *Department of Computer Science, University of Manchester, Manchester, UK*
*michaelb@computer-science.manchester.ac.uk*
*nick@computer-sciences.manchester.ac.uk*

✧ *Department of Surveying, University of Salford, Salford, UK*
*i.d.watson@surveying.salford.ac.uk*

**Abstract**

The number of successful, small-scale and purpose-built applications of CBR is growing rapidly. However, CBR has so far not been widely used as a methodology for reusing the large-scale data repositories typically maintained by a corporation. To facilitate this, cases must no longer be considered as concretely represented at the data level, but as virtual views of the underlying data. This paper argues that the basic requirement to support virtual cases are mapping functions between different data representations. It is argued that the use of mapping functions can increase flexibility in a number of ways. Multiple CBR applications can exploit a single data repository. Similarly, a single case representation can span multiple data repositories. Support for communication between different CBR applications as well as the evolution of case representation within a single application are also catered for by the same methodology. The paper concludes with a brief description of existing technology for support of the development and use of mapping functions.

## 1. Introduction

Case-Based Reasoning (CBR) has rapidly reached a state of maturity as a practical solution to relatively small-scale problems. Many successful industrial applications have been developed and a growing number of commercial shells are available to aid in application development [Althoff et al. 95]. A typical application will use CBR for a single purpose and a standardised case template will be employed to represent the data relevant to that application purpose. CBR ought to be a technology that is also well suited to the development of very large-scale Knowledge-Based Systems (KBS) as it has certain inherent advantages over other

KBS techniques [Watson & Marir 94]. The storage of knowledge as modular cases rather than as some coherent body of general-purpose rules can greatly reduce the initial knowledge acquisition costs as well as improving maintainability [Vargas & Raj, 93]. In addition, CBR promotes *cognitive economy* [Brown 92] as it generally requires less computational effort to adapt the solution to a similar past problem than to derive a solution to a new problem from first principles.

Current attempts at applying CBR technology on a large-scale have involved a commitment to knowledge engineering, with the cases and the indexing structure that organises those cases being specifically created for the purposes of the CBR application [Edelson 92]. However, a more commercially appealing approach is to use CBR as a technique for exploiting *existing* data repositories [Shimazu et al. 93], so that the data held therein can be re-used. In this paper, it is argued that many CBR applications should be constructed on top of a single large-scale data repository, where each CBR application is tailored to a specific purpose to which that data can be put. It will be argued that, to support this type of reuse of a data repository, the cases within each application must be *virtual views* of the underlying data.

## 2. Virtual Case Representation

In this section, a new approach to describing cases is discussed. Cases are no longer considered as being direct manifestations of the under-lying data but as virtual views of that data. To support this, a mechanism for mapping between different data representations is required. As will be described, this can lead to much enhanced flexibility in the use of CBR in comparison to existing technology.

### 2.1 Limitations of Current Case Representation

The prevailing approach in CBR tools is to treat cases as instances of some predetermined and static case template (e.g., as in CBR Express). In more sophisticated tools, cases can be structured into a hierarchy of case *fragments*, but there is still conformity in the structure of the hierarchy and in the representation of each fragments (e.g., as in KATE, ReCall, or ReMind). The use of standardised cases is popular because it greatly simplifies much of the CBR process, such as indexing, matching and retrieval. The representation of a case should directly reflect the purpose for which that case is to be used [Hammond 89]. Even where multiple uses of a single case are considered, a single case representation is usually assumed and subsets of the case features are selected by each of a number of

predetermined dimensions [Alterman & Wentworth 89; Pazzani, 89]. It follows that adherence to a standardised case representation is well suited if an application requires a single, problem-solving perspective. This is true in many small-scale practical applications which, perhaps, explains why currently available commercial tools rely on standardised cases. However, tailoring a case representation to the requirements of a single application is not feasible in large-scale applications where knowledge re-usage for a wide range of purposes will inevitably be required.

Another prevailing and limiting assumption is that the actual data that comprises a case-base is stored in the same format as the cases, i.e. the cases are inseparable from the data itself. In applications where the case data originates in pre-existing databases, this requires that either the cases conform to the format in which the data is stored, or a translation program is written to extract relevant data and store it within a new case base. This causes a problem of data redundancy. Frequent calls to the translator will be required to ensure that the case base is up-to-date. Moreover, if multiple CBR applications are built on top of a single database, there is no scope for direct communication of data between the CBR systems as each operates on its own localised case base.

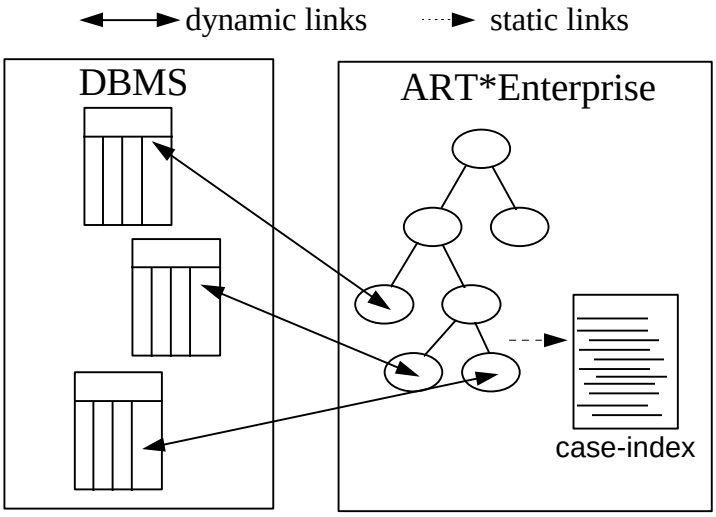


**Figure 1. ART*Enterprise Data Integration**

This is the situation with the current generation of CBR tools. The more sophisticated have good data import facilities (e.g., ESTEEM, KATE, and ReMind). The exception to this approach is ART*Enterprise which is able to map ART objects to data stored in relational tables (see Figure 1). The attributes of these object will be updated if the data in the relational tables changes or vice versa. However, although ART*Enterprise has CBR functionality and can match and retrieve ART objects it requires an index to be created. This index is static and will not be automatically changed if the underlying data changes.

What is proposed here is that the data required to populate a case base is *derived by mappings* attached to a standardised case format; hence, the cases are only indirectly linked to the raw data on which they are based. This indirection allows enhanced flexibility in a number of different ways including:

- Multiple views of cases representing the same data
- Cases that span multiple data repositories
- Support for co-operative CBR
- Evolution in case representation

## 2.2 The Need for Mapping between Cases and Raw Data

To genuinely reflect different uses of the same episodic data, there is a need to support heterogeneous case bases where several orthogonal cases may represent views of the data concerning a single episode [Goel et al., 91]. As a hypothetical example, consider the situation of a large construction company. To a first approximation, it is assumed that the company stores a detailed account of each building contract within a single unified data repository. This data repository is updated with new data as part of the routine operation of the company.

This hypothetical situation is typical for many companies in all areas of industry; the company has a massive amount of data which has the potential to be put to valuable usage. As noted in section 1 CBR is well suited to this exploitation. For example, CBR may be a useful guide to a construction company in tendering for new contracts. This task typically requires a complex estimation based on a wide variety of cost factors, from inevitable expenditure (such as on labour and materials) through to less certain factors (such as the potential risk of legal costs from litigation resulting from carrying out the contract). CBR allows this cost estimation to be made by taking the costing for a previous contract and making adjustments based on the significant differences found between the two cases.

CBR can also be applied for *risk factor analysis*. For example, by recording the circumstances relevant to previous situations in which a building company has been sued (e.g. due to failure to meet a contract deadline or due to environmental damage caused by the construction work) CBR may be used to predict the recurrence of such problems in the future. This is just a variant of standard CBR approaches to failure avoidance (e.g. [Hammond 90]).

Clearly, the feature sets relevant to the purposes of contract tendering and litigation prediction will have little over-lap, hence different case structures are

required for each application. However, the cases for each application should derive from the company's existing data repository. This is achieved by the establishment of a set of *mapping functions* between the general data schema for the data repository and the specific data schema for each case base. Hence, cases are in fact *virtual views* of the underlying data. Nevertheless, to an external CBR application, these cases should behave as concrete data structures.

The mapping functions required to establish each feature value of a virtual case are often complex and must deal with the problem of *semantic heterogeneity* [Brown et al. 95; Reddy et al. 94]. For example, in a tendering case, a single feature value may be required representing the combined sub-contractor cost for a particular building contract. In practise, establishing this value may require a selective search through the records of all sub-contractors concerned with the contract and a summation of the moneys paid these sub-contractors for the period of the contract.

In order for cases to be truly virtual but act as concrete data structures, the mapping functions to the underlying data repository should be navigated each time the case base is used, rather than using the mappings functions to generate a local and independent data repository into which the cases are stored as happens in ART*Enterprise. This implies that the population of cases within a virtual case base must be automatically generated. This requires part of the mapping to establish unambiguous constraints on what defines an individual case, with respect to the data stored in the underlying repository. The result of this *signature mapping* is a set of *case signatures*. Each case signature is a set of data values that are *sufficient* to uniquely identify that case. As an example, there may be exactly one tendering case for each building contract undergone by a construction company, hence a sufficient case signature is the name of each contract. In contrast, there may multiple litigation cases for each building contract, hence a more complex case signature is required (e.g. citing the plaintiff and defendant for the case, and the court ).

Unlike other feature values, the case signature *does* require localised storage. This is in order to provide the case-based application with a handle by which to refer to each case (hence enabling virtual cases to appear concrete). Periodic use of the signature mapping to check the validity of individual case signatures will be necessary to ensure that the virtual case base is kept up-to-date.

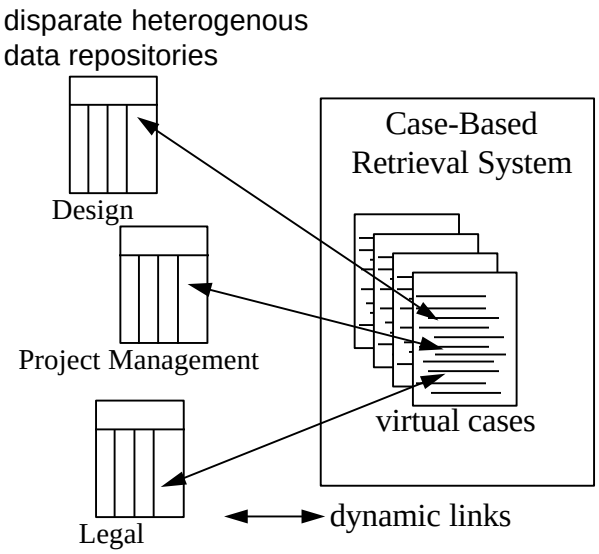


**Figure 2. Deriving A Single Case Base from Multiple Data Repositories**

In reality, the required mappings between data and virtual cases will be more complex because the assumption of a single unified data repository will not hold. A single virtual case may derive its feature values from multiple data repositories (as shown in Figure 2). For example, it is reasonable to assume that the variety of cost factors required to constitute a tendering case are in fact derived from multiple data repositories, covering distinct aspects of a construction company's operation (e.g., data repositories in design, project management and legal departments).

The complexity, in terms of implementation, for dealing with multiple data repositories is manifest in the signature mapping function. The case signature that is produced must be sufficient to distinguish the portion of data within *each* data repository that is required by the virtual case. For example, the personnel records held by the construction company may not store data concerning which contract each employee was employed on at any one time. Hence, a case signature comprising just the name of a contract would be insufficient to enable relevant sub-contractor records to be selected in the calculation of costs. However, the data repository which holds the records concerning each building contract may store which sub-contractor worked on each contract. Hence, the signature mapping function should access the contract records repository and derive case signatures including; the name of the contract, the list of sub-contractor names for that contract and other information (e.g. contract dates) required to enable all the distributed data for each virtual case to be unambiguously accessed.

## 2.3 The Need for Inter-Case-Base Mapping

The assumption that each virtual case base represents an *independent* viewpoint for the underlying data repositories may not always hold. In some circumstances, it may be more appropriate for one case-based application to derive values for part of its associated cases indirectly from the virtual cases used by another application, rather than using a direct mapping to the data repositories themselves. For example, there are two possible mapping routes by which the legal cost of a tendering case can be derived:

- Directly from the data repository; the signature of a case selected by the tendering application is used to search through all legal records to find the appropriate data.
- Indirectly via another virtual case base; the signature of a case selected by the tendering application is used to match against signatures in the litigation case base. The mappings from the selected litigation cases are then used to search the legal records for the appropriate data.

The indirect data access via route has two pragmatic advantages. Firstly, the inter-case-base mapping function may be easier to encode than ones that directly access the data repository; assuming an inter-case-base mapping can match case signatures between the two virtual case bases, then the existing direct mappings used for the legal risk assessment application can be exploited. Secondly, in performing an inter-case-base mapping additional information concerning the virtual case signature is inferred (e.g. the names of plaintiff and defendant in the legal case). This additional information may allow a more selective search of the underlying data repository.

The above discussion has described the need for inter-case-base mapping for two case bases created for orthogonal purposes. The need is even greater if more than one case base exists for the same purpose. The support for multiple reasoning agents *co-operating* is an appealing approach to problem solving in complex domains [Andreas et al. 92]. For example, multiple CBR applications could be used to access the legal risk of a new contract, each application concentrating on a different legal aspect (e.g. breech of contract, environmental pollution, etc.). Often complex legal cases will cover several different legal aspects, hence the data representing that case will be shared amongst multiple virtual case bases for the legal risk assessment application. Mappings will be required to maintain *equivalence* relationships between the case bases. Hence, if a single CBR application identifies a particular case as being important with respect to a current

contract, then the need to select the equivalent cases in other virtual case bases can be communicated via the inter-case-base mapping functions, hence, effecting the reasoning performed by other CBR applications.

### 2.4 The Need for Intra-Case-Base Mapping

A final potential source of flexibility is in the support of intra-case-base mappings. Intra-case-base mappings will be required where the representation of a case for a single application can be predicted to evolve over time. The need for a change in case representation may be prompted by changes in the underlying data repositories or because the requirements of the associated CBR applications change. Typically, new types of data in the underlying data repositories may prompt the creation of a completely new case feature. Conversely, types of data may become obsolete, requiring an existing case feature to be deleted. Finally, the format for the value for a particular case feature may change (e.g. from integer to floating point).

The most convenient (though not necessarily most efficient) way to respond to an evolving case representation is to maintain all versions of the case representation as well as intra-case-base mappings between them. Trivial mapping functions (i.e., simple equivalence relationships) are predominantly used to connect parts of a case representation that are unchanged between case versions. This allows existing complex mapping functions to the underlying data repository to be exploited by new case representation versions. New mapping functions (either between case representations or to the underlying data repository) are only inserted where there is change in the representation. The disadvantage is that multiple mapping functions may be called before the appropriate data is accessed, hence access time may be slightly increased. However, the major advantage is that multiple versions of the CBR application, requiring differing case representations, can be enabled to use the same virtual case base

## 3. Towards Virtual Case Bases

It has been argued in this paper that a new style of implementation for CBR systems is required where the representation of cases is separated from the underlying data. In this scheme, cases are virtual and act as application-specific views of the data. It has been argued that this methodology is essential for the future use of CBR technology to support large-scale, *corporate-wide* knowledge

bases, where CBR must rely on use of existing data repositories rather than specifically generated case data.

By making cases virtual, enhanced flexibility is achieved in a number of ways, including; a single data repository can be exploited by multiple CBR applications without introducing data redundancy, a single case can span multiple data repositories, and, evolution in the case representation for a single CBR application can be tolerated.

It has been argued that the fundamental requirement to produce the enhanced flexibility of virtual case bases is provision of sets of mapping functions between different data representations that can be navigated at access time. Technology needs to be developed for two distinct aspects:

1. Interactive support for the encoding of each mapping.
2. Automatic management of mapping functions during data access.

Significant advances in addressing these problems have recently been made at the University of Manchester as part of a non-CBR research project concerning the development of a CAD framework. CAD frameworks provide an environment into which separate software tools can be integrated [Filer et al. 94] so that various management tasks, such as managing the communication of data between tools, can be automated. Different tools can have different data representations, hence the same types of problem are encountered as involved in supporting virtual case bases.

Based on this work, a number of ways have been identified in which a knowledge engineer can be supported in the construction of a mapping. Firstly, a set of generic operators is provided that can enable a knowledge engineer to describe structural constraints upon the mapping [Brown 95]; such as the equivalence relationships that exist between attributes of different virtual cases. Some automated assistance for the task of establishing structural constraints is also possible. For example, finding similarities in the naming of constructs within two data representations is a powerful heuristic method for establishing where there is overlap in the semantics of the two representations [Mir 93]. Finally, although the automated encoding of individual mapping functions does not seem feasible [Brown et al 95], the established structural constraints ought to be used to generate a skeletal function body. For example, structural constraints will determine which items of data a particular case feature value is derived from, hence identifying the appropriate input parameters for the associated mapping

function code. One area for future work is to amalgamate these various components into a single integrated environment to providing a methodology by which a knowledge engineer can construct mappings.

Finally, the ideal way of supporting multiple CBR applications accessing multiple data formats is through some central data access module. This module should provide a set of procedures simple and general enough to provide data access for the variety of data representations used by the CBR applications and individual data repositories. The existing GPIC system [Filer et al. 94] for database access has been designed and implemented to fulfil these requirements. The proposed future development of GPIC is to provide facilities for storing mapping functions within the access module and providing automated management of the invocation of these functions. This will provide the ideal basis for support for construction of virtual case bases and, hence, by the arguments of this paper, an appropriate technology for the development of large-scale, multi-purpose CBR applications.

## 4. Conclusions

Our experience of applying CBR to knowledge reuse within industrial sectors such as the construction industry has highlighted a severe limitation in the current generation of CBR tools. If the uptake of CBR is to become exponential over the next few years, as Alex Goodall states in his preface to Althoff et al's recent report on CBR tools [Althoff et al. 95], it is essential that CBR tools can use data in an organisation's existing data repositories. The creation of local data models, as in ART*Enterprise, is welcomed but is only a partial solution. This paper has set out a theoretical framework for deriving *virtual cases* from data held in disparate heterogeneous data repositories by using mapping functions. Although theoretical it is based on sound research already carried out by Manchester University. The authors of this paper are now seeking research funding to develop a workbench that will combine conventional direct data retrieval with virtual cases and indirect retrieval using the methodology described here.

## 5. References

Alterman R. and Wentworth M (1989). Determining the Important Features of a Case. In, Proc. of the Case-Based Reasoning Workshop, pp.197-202, Morgan Kaufmann Publishers.

Althoff, K-D, Auriol, E., Barletta, R. & Manago, M. (1995*). A Review of Industrial Case-Based Reasoning Tools*. AI Intelligence, Oxford, UK. ISBN 1 898804 01 X

Andreas S, Schlageter G and S Kirn (1992) *Problem Solving in Federative Environments: The FRESCO Concept of Cooperative Agents*. In, The New Generation of Information Systems: From Data to Knowledge. M. P. Parazoglou and J Zeleznikow. Springer-Verlag

Brown M (1995). *Generic Operators for Schema-to-Schema Mappings*. Project Report CAD Group, The University of Manchester,

Brown M (1992) *Case-Based Reasoning: Principles and Potential*. In , AI Intelligence, Oxford, pp. 1 - 23 Editor, Alex Goodall

Brown M, Moosa Z, N Filer, Heaton J and J Pye (1995). *Close Integration of a CAD Vendor's Framework into the Jessi-Common-Frame Using a Flexible and Adapatble Procedural Interface*. In, Proc. of The Int. Workshop on Concurrent/Simultaneous Engineering Frameworks and Applications, Lisboa, Portugal, 5-7 April

Edelson D C (1992) *When Should a Cheetah Remind You of a Bat? Reminding in Case-based Teaching*. In, Proceedings of AAAI-92, pp. 667-672.

Filer N, Brown M and Moosa Z (1994). *Integrating CAD Tools into a Framework Environment Using a Flexible and Adaptable Procedural Interface*. In, Proc. of EURO-DAC '94. IEEE-CS Press. pp. 200-205,

Goel A K, Kolodner J L, Pearce M and R Billington. (1991) *Towards a Case-Based Tool for Aiding Conceptual Design Problem Solving*. IN, Proc. of the Case-Based Reasoning Workshop, may 8-10, pp. 109-120.

Hammond K J (1989) *On Functionally Motivated Vocabularies: An Apologia*. In, Proc. of the Case-Based Reasoning Workshop pp. 52-56. Morgan Kaufmann Publishers}

Hammond K J (1990) *Explaining and Repairing Plans That Fail*. Artificial Intelligence. Vol. 45. pp.173-228.

Mir S (1993) *Heuristic Reasoning for an Automatic Commonsense Understanding of Logic Electronic Design Specifications*. PhD Thesis, The University of Manchester, UK,

Pazzani M J (1989) *Indexing Strategies for Goal Specific Retrieval of Cases*. In, Proc. of the Case-Based Reasoning Workshop. pp.31-35. Morgan Kaufmann Publishers

Reddy M P, Prasad B E, Reddy P G and Gupta A. (1994) A *Methodology for Integration of Heterogeneous Databases*. in, IEEE Transactions on Knowledge and Data Engineering. Vol. 6, No.6, pp. 920-933

Shimazu H, Kitano H and A Shibata (1993) *Retrieving Cases from Relational Data-Bases: Another Stride Towards Corporate-Wide Case Base Systems*. IJCAI-93, Proc. of the 13th Int. Joint Conf. on Artificial Intelligence, Aug 28 - Sept 3, pp.909-914, Morgan Kaufmann Publishers.

Vargas, J.E., & Raj, S. (1993). *Developing maintainable expert systems using case-based reasoning*. Expert Systems, 10(iv): pp.219-25.

Watson, I. & Marir, F. (1994). *Case-Based Reasoning: A Review*. The Knowledge Engineering Review, Vol.9 No.4 pp.327-54.