A framework for automated construction and transformation of case-based reasoning systems [*]

Jing Ma, Arijit Sengupta and David Wilson
Computer Science Department
Lindley Hall, Indiana University
150 S. Woodlawn Ave
Bloomington, IN 47405
{jima,asengupt,davwils}@cs.indiana.edu

July 20, 1999

Computer Science Technical Report 525

Indiana University, Bloomington

[*]Adapted from Sengupta, Wilson and Leake [SWL99a] with implementation details from Ma.

Abstract

Case Based Reasoning systems have gained immense popularity over the recent years as problem-solving tools. Most case based reasoning systems, however, are developed essentially from scratch using proprietary systems and applications on a limited number of platforms. Although methods have been proposed to describe the structure of a case based reasoner, none of these have been very successful outside their application domains. In this paper, we first describe common methods for automating CBR system construction. We then describe a general model for common CBR implementation, and describe in detail a framework of platform-independent construction of systems based on this model. We discuss an implementation of such a system using Java, and finally describe ways systems can be developed using this framework.

1 Introduction

Achieving widespread case-based reasoning support for corporate memories will require flexibility in integrating implementations with existing organizational infrastructure and resources. We need to investigate not only that particular implementations have addressed some pieces of the puzzle, but also general implementation infrastructure on which all systems can build. CBR systems as currently constructed tend to fit three general implementation models, defined by broad implementation constraints on representation and process, which reflect evolutionary developments in CBR practice.

Traditionally, *task-based* implementations have addressed system goals based only on the constraints imposed by the reasoning task itself. Most research systems, for example, focus on particular (often idiosyncratic) methods and representations optimized to address a specific reasoning task, either to demonstrate the effectiveness of the method or to meet specific task goals.

Recently, there has been an increasing and successful trend of incorporating CBR into enterprise systems (e.g [Wat97, SW98]) to leverage corporate knowledge assets by knowledge management (e.g. [BFA99]). *Enterprise* implementations reflect the additional implementation constraints imposed on CBR systems as part of an overall enterprise architecture (see [KS96]). In our view, the most important implementational constraint in this context is that typically such CBR integrations must operate in conjunction with database systems, the mainstay of corporate knowledge activity. This will usually mean inter-operation with the more prevalent relational database systems (e.g. [GW98, KS96, APMH95]), but may also include object database systems (e.g. [Ell95]). Thus enterprise CBR implementations provide for and make use of database functionality. Note that not all "enterprise CBR systems" will have an enterprise implementation in this sense.

Currently, CBR systems are emerging that take advantage of recent developments in knowledge representation and sharing on the world-wide web (e.g. [Shi98, GW98, DFH$^+$98]). *Web-based* implementations reflect additional constraints imposed on CBR systems by conforming to structured document representation standards for web/network communication, in particular XML—Extensible Markup Language [BPS98]. Note that our type distinction here is based on the construction of the reasoning system itself, not on how it presents information. Thus a task-based implementation might have a web interface, and a web-based implementation might not.

These implementation characterizations are intended to be useful, not perfect. They represent implementation targets in constructing corporate memories, and varying task aspects and contexts may prefer one to another. Thus it is important to understand (1) how the models compare, (2) their individual construction, (3) their combination, and especially (4)

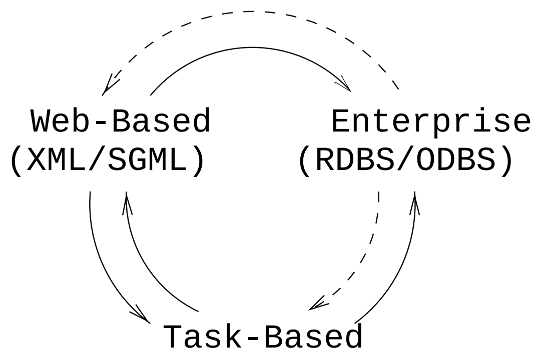

Figure 1: Relating CBR implementation types

how one may be constructed by transforming another. Transformations are useful when new task criteria suggest a model that differs from current implementation (conversion), and when differing models are used in different aspects of a combined system (combination, e.g. database storage, web communication, task-based front end). This paper outlines a framework of practical constructions and transformations, represented in figure 1, that we expect will play an important role in building and maintaining case-based corporate memories.

2 Implementation Models

Our implementation characterizations can be applied at many levels of typical CBR systems, and here we find it useful to differentiate CBR process (retrieval, adaptation, evaluation) and (case) representation. Although we recognize the importance of complex object and object-oriented database models, as well as Standard Generalized Markup Language (SGML, [ISO86]), here we restrict our discussion to relational database models and XML.

Task-Based: Task-Based implementations account for the bulk of current CBR practice. These systems allow for highly tuned and efficient metrics and representations, but it may prove difficult to reuse them outside of the system context. Some efforts have used standardized representations to ameliorate these difficulties (e.g. [MBC$^+$94]), but this is not widespread.

Enterprise: Integrating CBR implementations with enterprise database systems imposes standardization constraints that are almost universal in the enterprise community. Representations must accord with the table model of relational database systems (RDBS), while process must adopt Structured Query Language (SQL) conventions. CBR systems gain the strengths of the underlying RDBS, such as security, concurrency control, backup/recovery, and scalability. Moreover, integration allows the use of enterprise data both for normal corporate tasks (e.g. reporting), as well as for reasoning. However, because SQL has been designed to provide certain performance guarantees, it is limited in power, so refined metrics may be difficult to construct. Also, while complex cases are representable, they may be difficult to model in manual construction.

Web-based: XML is emerging as the vehicle for knowledge representation on the web. It provides a medium that allows (1) definition of customized representational markup languages and (2) application independent exchange of these complex hierarchical representations over existing web/network channels. XML also allows for customizable display of information using the associated Extensible Style Language[1]. While XML is currently viable for use (e.g. for transfer and parsing), it is a fairly new standard, so support (e.g. for browsing) is limited though growing. Its usability for applications such as CBR is also still evolving relatively rapidly [HC99]. Thus some benefits are immediately available for individual systems, but developing standard representations for community knowledge sharing will

[1]http://www.w3.org/TR/1998/WD-xsl-19981216

be a crucial task for widespread use in the field. Since XML is primarily a representation standard, it is not tightly coupled with process as are databases, so task-based applications are generally required for process. However, direct structured query mechanisms, analogous to SQL, are under development [Sen98, W3C98].

3 Realizing Implementations

The realization of a framework for automatic implementation transformation involves outlining process and representation for each model, as well as defining and exemplifying the inter-model transformations. This section outlines the enterprise and web-based models (we omit the multifarious task-based model that abounds throughout the literature), and section 4 describes the transformations.

3.1 Enterprise/RDBS

Constructing an enterprise implementation involves associating a case structure with a corresponding relational database schema. Figure 2 shows an Entity-Relationship (ER) model for typical CBR systems, where stored data represents cases (problems) which result in proposed decisions (solutions), and their outcomes (evaluations). Almost any general CBR system can be represented using this ER model by appropriately identifying the different components of the problem space. Once represented within the model, the system can be fully implemented in a RDBS. The construction is straightforward for feature-vector case structures, where a single table row corresponds to a case, and the implementation will consist of at most four linked tables, one each for the three entities, and one for the relationship. For more complex case structures, relational normalization techniques are used to model the data.

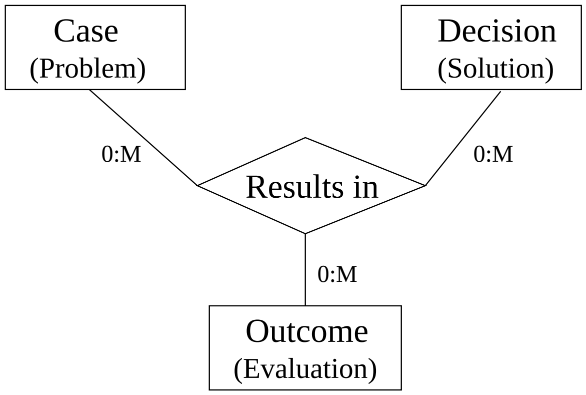

Figure 2: Entity Relationship diagram for a typical case-based reasoning process

Database systems can also be used for CBR process, for example by implementing k-nearest neighbor (k-nn) retrievals. A number of novel data structures have been proposed in the database literature for efficient implementation of k-nn algorithms (e.g. [BBKK97]), but standard database systems do not currently offer such support. However, if the similarity metric can be expressed as a numeric-valued function, database cases can be retrieved as ordered by the similarity results. Thus we view database/CBR process as taking place on at least three levels:

1. *Simple Storage*: The database is used only as a storage medium. All cases are retrieved and processed by an external system. This combines the storage benefits of the database systems with task-based processing power, but requires a full task-based implementation. The basic query to the database in this case is:

   ```
   SELECT * FROM case_table
   ```

2. *Simple Retrieval*: A simple selection is performed based on conditions applied from the target, and the resulting subset is processed externally. This shifts part of the processing task to the database system, but may require considerable modeling effort

to pre-compute similarity as in [Shi98], or to relax query specifications as in [GW98, DL97]. The basic query here is:

```
SELECT * FROM case_table WHERE conditions
```

3. *Metric Retrieval*: A metric function is used to order the rows based on a similarity value, `metric(c)`—a function of the target case c. This uses only the database system itself to perform a full k-nn analysis. This method is inefficient, since it must both compute and sort with every record and loses the efficiency of optimized database indexing. Thus it has been rejected in the past [SKS93], but could prove useful for some implementations, since it does not require additional processing for retrieval. We have used this method with good response time in a prototype application containing 4709 cases with 24 numeric-valued features. The basic query is:

```
SELECT * FROM case_table ORDER BY metric(k)
```

To take full advantage of database capabilities, a pre-selection of the cases in the case-base could be performed using simple retrieval before evaluating metric retrieval, to reduce (if possible by exact/ranged matching) the number of retrieved cases.

3.2 Web-based/XML

Based on the entity-relationship model of CBR in figure 2, we can also describe the structure of a full CBR system using an XML document type definition (DTD). The following DTD was used for the current implementation:

```
<!ELEMENT CBR   (DATA, PROCESS?)>
<!ATTLIST CBR   NAME CDATA #REQUIRED>
```

```
<!ELEMENT DATA  (PROBLEM, SOLUTION, EVAL?, RESULT?)>
<!ELEMENT PROBLEM (ATTRIBSET)>
<!ELEMENT SOLUTION (ATTRIBSET)>
<!ELEMENT EVAL   (ATTRIBSET)>
<!ELEMENT RESULT (ATTRIBSET)>
<!ELEMENT ATTRIBSET (ATTRIB | ATTRIBSET)+>
<!ATTLIST ATTRIBSET NAME CDATA #REQUIRED>
<!ELEMENT ATTRIB (#PCDATA)>
<!ATTLIST ATTRIB TYPE CDATA #REQUIRED
                 REQD (REQD|NOTREQD) "NOTREQD"
                 ID   (ID|NOTID) "NOTID"
                 SIZE CDATA #IMPLIED
                 RANGE CDATA #IMPLIED
                 CONSTRAINT CDATA #IMPLIED>
<!ELEMENT PROCESS (METRIC+, ADAPT*)>
<!ELEMENT METRIC (#PCDATA)>
<!ATTLIST METRIC  NAME CDATA #REQUIRED
                  LANG CDATA #IMPLIED
                  TYPE CDATA #IMPLIED>
<!ELEMENT ADAPT  (#PCDATA)>
<!ATTLIST ADAPT   NAME CDATA #REQUIRED
                  LANG CDATA #IMPLIED
                  TYPE CDATA #IMPLIED>
```

XML documents conforming to this CBR DTD describe the structure (i.e. meta-data) of particular CBR systems. Components of the case base are expressed as relations (attribute sets) and their constituent attributes. Complex hierarchies are supported by allowing sub-relations inside a relation (i.e., an ATTRIBSET inside another ATTRIBSET in the DTD). In contrast to other DTDs for CBR [Shi98, HCD98], we allow representation of both process (similarity, adaptation, evaluation) and case representation, either together or individually. For example, we are currently working on an implementation that incorporates MathML[2], an XML DTD for describing mathematical structure and content, to represent similarity metrics.

[2]http://www.w3.org/TR/REC-MathML/

Using the XML model: An instance of the above DTD describes the actual case structure, which is used by a separate XML application to generate the proper structural definition (a separate DTD) of the case data. The actual case data can then be defined as conforming instances of the generated DTD. This two-step process has the following advantages:

1. *Consistency*: By generating the case data DTD from the CBR system markup, we ensure that no separate check is necessary to assert the consistency of the data with the reasoning system.

2. *Validation*: Document type definitions in which the system attributes are represented as generic identifiers (tags) instead of XML attributes allows the case data to be validated against its DTD to ensure its integrity.

While XML has no particular associated process for retrieval, evolving query language implementations such as DSQL in DocBase [Sen98] and XML-QL [W3C98] will enhance the applicability of XML as a web-based CBR implementation model.

As a toy example of this process, consider the following instance of the above DTD describing a simple linear systems analyzer (LSA), which includes (i) a flat feature vector describing a linear system (the problem space), (ii) a solution vector that includes the data structure used to solve the system and (iii) a result vector including the performance values.

```
<!DOCTYPE CBR SYSTEM "cbr.dtd">
<CBR NAME="LSA">
<DATA>
   <PROBLEM>
     <ATTRIBSET NAME="Features">
        <ATTRIB TYPE="int" ID="ID">FeatureID</ATTRIB>
        <ATTRIB TYPE="float">squareness</ATTRIB>
        <ATTRIB TYPE="float">diagonalness</ATTRIB>
```

```
      </ATTRIBSET>
    </PROBLEM>
    <SOLUTION>
      <ATTRIBSET NAME="Sol">
         <ATTRIB TYPE="int">DS</ATTRIB>
      </ATTRIBSET>
    </SOLUTION>
    <EVAL>
      <ATTRIBSET NAME="Eval">
         <ATTRIB TYPE="float">Perform</ATTRIB>
      </ATTRIBSET>
    </EVAL>
</DATA>
<PROCESS>
<METRIC NAME="Similarity" LANG="SQL">
select FeatureID,sqrt(power((squareness - <? param squareness ?>),2)
                  power((diagonalness - <? param diagonalness ?>),2))
                  as distance
from Features order by distance ascending
</METRIC>
</PROCESS>
</CBR>
```

The system described in the above DTD contains a feature vector containing only two features, a solution including the data structure used, and a result vector with performance values. Based on this instance, a DTD for the actual case data is generated, which has the following structure:

```
<!ELEMENT LSA (Features, Sol, Result, )>
<!ELEMENT Features (Featurestup*)>
<!ELEMENT Featurestup (FeatureID, squareness, diagonalness)>
<!ELEMENT FeatureID (#PCDATA)>
<!ELEMENT squareness (#PCDATA)>
<!ELEMENT diagonalness (#PCDATA)>
<!ELEMENT Sol (Soltup*)>
<!ELEMENT Soltup (SolID, DS)>
<!ELEMENT SolID (#PCDATA)>
<!ELEMENT DS (#PCDATA)>
<!ELEMENT Eval (Evaltup*)>
<!ELEMENT Evaltup (EvalID, Perform)>
<!ELEMENT EvalID (#PCDATA)>
```

```
<!ELEMENT Perform (#PCDATA)>
<!ELEMENT Result (Resulttup*)>
<!ELEMENT Resulttup (FeatureID, SolID, EvalID)>
```

The XML data conforming to the above DTD will represent the actual case data for this CBR system. Such data can be automatically tagged using a text conversion process, or can be generated from other structured content.

4 Transforming Implementations

Perhaps as important as the implementations themselves is the transformation of one implementation to another. This is useful in two situations: When new task criteria prefer a model that differs from current implementation, and when differing implementation models are used in different aspects of a combined system (e.g. database storage, web communication, task-based front end). Here we outline the transformations in the framework.

4.1 Web-Based → Enterprise

An XML representation of case structure can be converted to a database system using an XML application that processes XML markup tags/content and generates appropriate Data Definition Language (DDL) statements to create tables in a relational database. Consider the following fragment of a CBR system description, relating a person to their automobile:

```
<ATTRIBSET NAME="Person">
  <ATTRIB ID="ID" REQD="REQD" TYPE="longint">SSN
  </ATTRIB>
  <ATTRIB TYPE="char" SIZE="20" REQD="REQD">Name
  </ATTRIB>
  <ATTRIBSET NAME="Auto">
    <ATTRIB TYPE="char">Make</ATTRIB>
    <ATTRIB TYPE="int">Year</ATTRIB>
  </ATTRIBSET>
</ATTRIBSET>
```

By parsing this XML fragment and mapping the XML structure to relational table structure, the patterns can be translated into the following relational DDL statements:

```
create table Person (SSN longint not null,
                     Name char(20) not null);
create table Auto (Person_SSN longint not null,
                   Make char(50), Year int);
```

For complex case structures, the application can adopt a simple foreign key strategy by augmenting a substructure with the key of the parent structure. In order to facilitate a possible future back-translation, this application should also update a database catalog (organized list) with the role of each created tables in the CBR model. A similar transformation application can be used to transform XML case data to fill the generated tables.

The toy example on the LSA system described earlier would result in DDL statements like the following:

```
create table Features ( FeatureID int unique, squareness float(15)  ,
diagonalness float(15))
go
create table Sol (SolID int, DS int)
go
create table Eval (EvalID int, Perform float(15))
go
create table Result (FeatureID int, SolID int, EvalID int)
go
```

The above structure can be improved by identifying the types of relationships, and by incorporating single-attribute attributesets directly into the Result vector, which will result in a more compact description of the system, as follows:

```
create table Features ( FeatureID int unique, squareness float(15)  ,
diagonalness float(15))
go
create table Result (FeatureID int, DS int, Perform float(15))
go
```

4.2 Web-Based → Task-Based

The main task in transforming an XML implementation to a task-based implementation is to identify a mapping between XML and task-based structures. We assume that the user or developer of the task-based systems will have the necessary tools and information to create case data in the task-based model. Taking CASUEL [MBC+94] as an example, an application like the one described in section 4.1 can generate appropriate CASUEL declarations from the XML structure. This process is similar to the Web-Based→Enterprise generation process, except that the generated statements are in CASUEL instead of SQL.

CASUEL has a somewhat of an object-oriented modeling approach. Although it does not support a complete object-oriented language, it does support abstract typing and inheritance. However, going from XML to CASUEL, hence, is fairly straight forward, by creating classes for every attribute set, and for a nested attribute set, simply making the subclass a member of the parent class. If the data is available in XML as well, it can also be represented in the CASUEL data format using the same format. The above toy example of the LSA will be represented in CASUEL as follows:

```
defdomain LSA;

...

defclass Features a_kind_of class;
slots FeatureID, squareness, diagonalness.
```

```
defclass Sol a_kind_of class;
slots SolID, DS.

defclass Eval a_kind_of class;
slots EvalID, Perform.

defclass Result a_kind_of class;
slots FeatureID, SolID, EvalID.
```

4.3 Enterprise → Web-Based/Task-Based

Transforming an existing database model into a conforming XML model or task-based model is more involved. Because the database lacks explicit case structure (when using more than a single table), transformation applications need to understand the role of various database objects in the CBR representation. Maintaining a catalog of the database objects and their roles, as suggested in section 4.2, should significantly reduce the amount of reasoning required prior to transformation. This process of role determination can be performed in several ways:

1. *Manual interaction:* The system may ask a user to assist in the process of determination of the roles of each of the objects,

2. *Catalog information:* The system may use a catalog that includes the roles of each of the objects,

3. *Mining:* The system may use data mining techniques to determine appropriate database objects and their roles.

The dashed lines in figure 1 represent the extra information requirements for these transformations.

4.4 Task-Based → Web-Based/Enterprise

Converting from task-based to an XML or database format also depends on the actual task-based model, and the availability of tools that can assist in such transformation. For example, cases represented using CASUEL can be mapped into the corresponding XML schema or a database format using an application built on top of a CASUEL parser.

5 Java Implementation

This section describes the architecture as well as the actual implementation of the transformation routines.

5.1 Languages, platform and tools

We used the Java programming language for the implementation of the transformation routines. One important consideration behind the use of an object-oriented language was that they ensure easy extensibility using inheritance and overloading. We used external applications for the purpose of parsing XML and CASUEL. The XML parser (XP) is courtesy James Clark (http://www.jclark.com), which provides a Java class library for parsing XML documents into an event structure that can be processed for many different purposes. The transformation routines from CASUEL (currently under development) use the CASUEL parser from INRECA, which is written in C with lex and yacc. We have plans for rewriting the parser entirely using a Java-based parser generator.

The prototype system was designed to run over the World-Wide Web independent of platform, a useful outcome for using Java as a programming language. We are currently developing a version that can be run directly over the web using a world-wide web browser.

Because of the necessity of file operations in this system, we need to consider workarounds for the security restrictions in Java applets.

5.2 Supporting tools

The primary supporting application in the current version of the prototype is XP, an XML parser. An XML instance is parsed through the parser, and the events generated from the result of parsing is used for further processing purposes. XP reads through the XML input and generates events corresponding to the start and end tags found in the XML. Each of these events can be processed in sequence, while populating a data structure internally (described below). The parser also includes names and values of the tags and attributes in the event objects. This is very useful for the cases where further information on the tag are available in one or more attributes.

5.3 The transformation process

This section describes the process of the XML to enterprise transformation. This is part of the application interface which walks the user through the complete transformation process. Currently only the XML to Enterprise transformation is fully functional. Implementations of the other transformations are to be completed within the recent future.

In the graphical user interface, after the user selects web-based system as the starting point, the interface changes to an editing mode where the user can enter an instance of the CBR DTD. This can be either pasted into the edit buffer, or written from scratch. Internally, the XML instance is opened by `EntityManagerImpl.openFile()`, then parsed by `ParserImpl.parseDocument()`, both methods in the XP parser library. Events are generated by ParserImpl while it parses through the XML instance. Some of these events include (i) `StartElement()`, invoked when a start tag is identified, and (ii) `EndElement()`, invoked when an end tag is identified.

The transformation engine then builds up an data structure representing the hierarchy of elements in the XML tree, using the events generated from the parser. In this data structure, there are two primary type of nodes `AttributeSet`, representing an attribute set, and `Attribute`, representing an attribute in the attribute set. These two nodes were implemented using Java classes with the same name, and were used to store the physical data in the structure.

For the implementation of the transformation, three instances of `AttributeSet` were used, one each corresponding to the problem, solution and evaluation. Thus, in response to an `attribset` tag, an instance of the `AttributeSet` class is created, and similarly an instance of the `Attrib` class is created corresponding to the `attrib` tags. The hierarchy of these classes is maintained according to the XML structure. The algorithm that builds the data structure uses a simple stack-based algorithm, to ensure the proper handling of the most recently open `attribset` tag.

The actual conversion is performed by writer classes that implement the `outputString()` method, which generates events corresponding to the type of output that is intended. For example, for enterprise output, the `outputString()` method is implemented by the `SQLWriter` class which generates appropriate SQL output corresponding to the current element of the data structure.

6 User-interface

A user-interface to drive the creation and transformation of the systems was also developed in Java. The interface was developed using a simple "wizard" model. The user is given options to select the type of system she wants to start with. For the selected type of system, the wizard then identifies different methods for specifying the system structure, and incorporation of the data. Once the data is incorporated into the system, the wizard can then present various methods for actual implementation, including implicit conversions if necessary. In the end, the system allows the user to pose simple case retrievals.

The basic operation of the transformation wizard is explained in the control-flow diagram in figure 3. At the beginning of the process, the system identifies the type of the existing system. It is possible to start without any system, in which case the user needs to design one during the process. For an XML-based system, the user needs to describe the system using an instance of our CBR DTD. This instance is then automatically translated into a database schema. If a database connection is available, the system can also build the database objects for which the DDLs are generated. If the original system has a database implementation, an XML instance corresponding to that database structure is generated. For a CASUEL-based system, both XML and database schema are generated (CASUEL to database is performed by CASUEL to XML and then XML to database). At the end of these steps, the system has the description of the system in XML, and an implementation in a database, which is used to upload and enter data, as well as perform case retrieval operations.

The Java implementation of this conversion wizard is presently under construction, and can be obtained from http://www.cs.indiana.edu/∼asengupt/cbdb/.

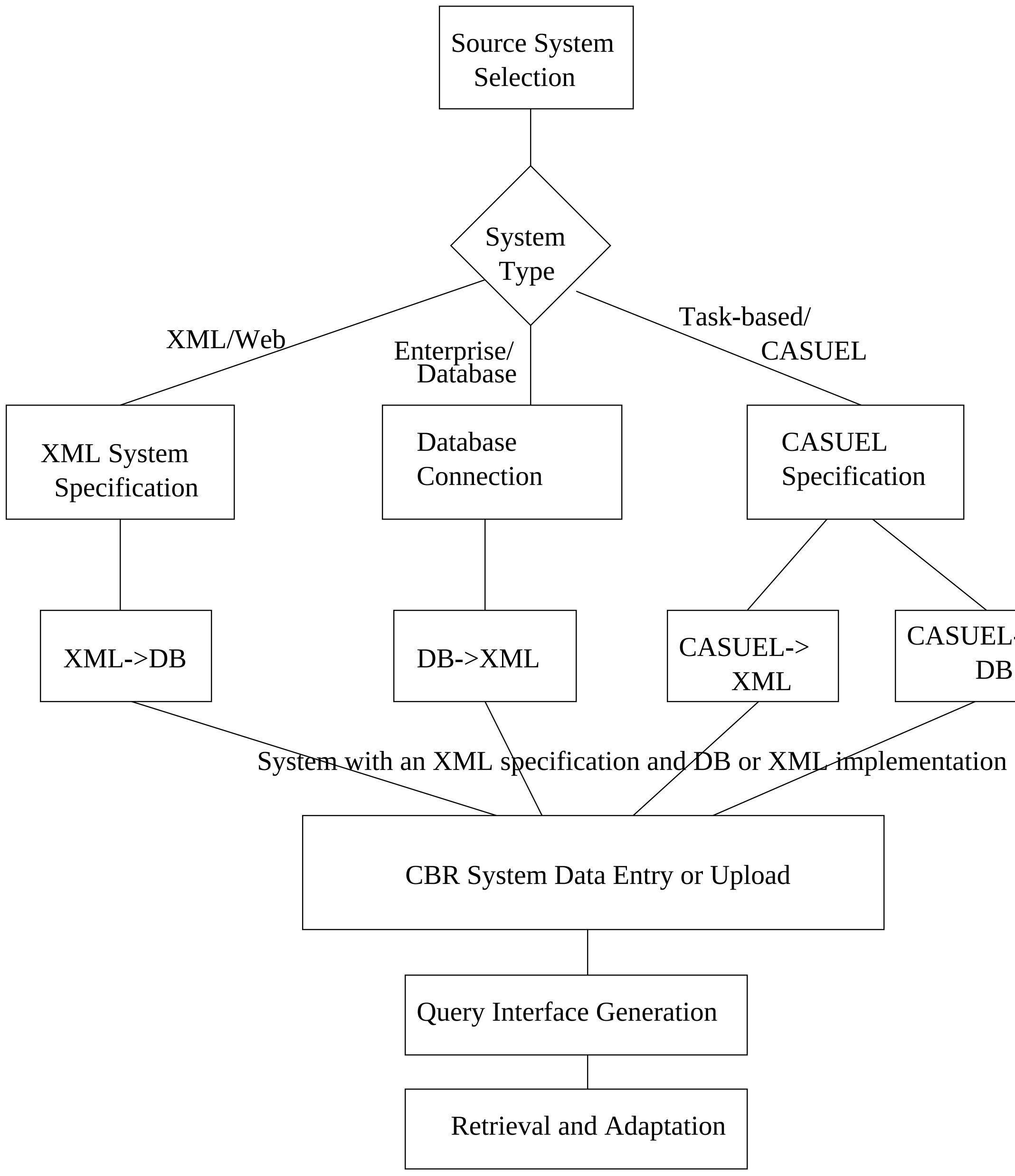

Figure 3: Control-flow of the CBR conversion wizard

7 Conclusion

We have presented a categorization of current CBR implementation models into three classes, and shown how this view leads to practical support for building and maintaining case-based corporate memories. The general transformations from one implementation model to another allow for the conversion of existing implementations and facilitate the combination of implementation types to meet new and changing task requirements. We also view these methods as a natural extension and generalization of mining databases to aid in system construction.

Based on this framework, we present three challenges to the community: (1) to create community standard XML representation specifications for CBR, (2) to build a set of standard methods/libraries for translating between these XML representations and standard database representations, and (3) to develop standard CBR functionality within database systems. This requires shifting some attention to building infrastructure for the field in areas that are constrained enough to be feasible and consequential enough to be worthwhile. Community standards for representation have long been sought in many areas of AI. The structural foundations provided by web-based and enterprise media, coupled with the impetus for recognition as a community in developing successful case-based corporate memories provides an environment ripe for achieving this goal for CBR. By providing general representational frameworks that already have ties to the world of practical application, as well as the tools to integrate them with one another and with traditional practice, the CBR community shapes the building blocks for constructing the next generation of successful research and industrial CBR systems. As CBR practice evolves, we expect the different implementation types to become increasingly integrated, and we hope to facilitate that transformation.

8 Acknowledgements

This technical report is an extended version of [SWL99a], with details of the system implementation. Much of the motivation for this paper is also derived from [SWL99b], in which the primary idea of the conversion model is proposed.

References

[APMH95] Jonathan RC Allen, David WR Patterson, Maurice D. Mulvenna, and John G. Hughes. Integration of case based retrieval with a relational database system in aircraft technical support. In *Proceedings of ICCBR-95*. Springer, 1995.

[BBKK97] Stefan Berchtold, Christian Bohm, Daniel Keim, and Hans-Peter Kriegel. A cost model for nearest neighbor search in high-dimensional data space. In *Proceedings of the 16th ACM PODS Conference*, 1997.

[BFA99] Irma Becerra-Fernandez and David W. Aha. Case-based problem solving for knowledge management systems. In *Proceedings of FLAIRS-99*. AAAI Press, 1999. To Appear.

[BPS98] Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. *Extensible Markup Language 1.0*, 1998. http://www.w3.org/TR/1998/REC-xml-19980210.

[DFH+98] Michelle Doyle, Maria Angela Ferrario, Conor Hayes, Pádraig Cunningham, and Barry Smyth. CBR Net:- smart technology over a network. Technical Report TCD-CS-1998-07, Trinity College Dublin, 1998.

[DL97] Jirapun Daengdej and Dickson Lukose. How case-based reasoning and cooperative query answering techniques support RICAD? In *Proceedings of ICCBR-97*, pages 315–324. Springer, 1997.

[Ell95] Jeremy Ellman. An application of case based reasoning to object oriented database retrieval. In Ian Watson, editor, *First United Kingdom Workshop on Case-Based Reasoning*. Springer, 1995.

[GW98] Dan Gardingen and Ian Watson. A web based case-based reasoning system for HVAC sales support. In *Applications & Innovations in Expert Systems VI*. Springer, 1998.

[HC99] Conor Hayes and Pádraig Cunningham. Shaping a CBR view with XML. In *Proceedings of ICCBR-99*, 1999. To Appear.

[HCD98] Conor Hayes, Pádraig Cunningham, and Michelle Doyle. Distributed CBR using XML. In *Proceedings of the KI-98 Workshop on Intelligent Systems and Electronic Commerce*, number LSA-98-03E. University of Kaiserslauten Computer Science Department, 1998.

[ISO86] International Organization for Standardization, Geneva, Switzerland. *ISO 8879: Information Processing – Text and Office Systems – Standard Generalized Markup Language (SGML)*, 1986.

[KS96] H. Kitano and H. Shimazu. The experience sharing architecture: A case study in corporate-wide case-based software quality control. In *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. AAAI Press, 1996.

[MBC$^+$94] Michael Manago, Ralph Bergmann, Noël Conruyt, Ralph Traphöner, James Pasley, Jacques Le Renard, Frank Maurer, Stefan Wess, Klaus-Dieter Althoff, and Sylvie Dumont. CASUEL: A common case representation language. INRECA Consortium. Available on the World-Wide Web at http://wwwagr.informatik.uni-kl.de/~bergmann/casuel/CASUEL_toc2.04.fm.html, 1994.

[Sen98] Arijit Sengupta. Toward the union of databases and document management: The design of DocBase. In *Proceedings of COMAD-98*. Tata McGraw Hill, 1998.

[Shi98] Hideo Shimazu. A textual case-based reasoning system using XML on the world-wide web. In *Proceedings of the Fourth European Workshop on Case-Based Reasoning*. Springer, 1998.

[SKS93] Hideo Shimazu, Hiroaki Kitano, and Akihiro Shibata. Retrieving cases from relational data-bases: Another stride towards corporate-wide case-base systems. In *Proceedings of IJCAI-93*, 1993.

[SW98] Markus Stolpmann and Stefan Wess. *Intelligente Systeme für E-Commerce und Support*. Addison Wesley, 1998.

[SWL99a] Arijit Sengupta, David Wilson, and David Leake. Constructing and transforming cbr implementations: Techniques for corporate memory management. In *Proceedings on ICCBR Workshop on Practical Case-Based Reasoning Strategies for Building and Maintaining Corporate Memories*, Munich, Germany, July 1999.

[SWL99b] Arijit Sengupta, David C. Wilson, and David B. Leake. On constructing the right sort of CBR implementation. In *Proceedings of the IJCAI-99 Workshop on Automating the Construction of Case Based Reasoners*, 1999. To Appear.

[W3C98] W3C. *XML-QL: A query language for XML*, 1998. http://www.w3.org/TR/1998/NOTE-xml-ql-19980819/.

[Wat97] I. Watson. *Applying Case-Based Reasoning: Techniques for Enterprise Systems.* Morgan Kaufmann, San Mateo, CA, 1997.