Distributed CBR using XML

Conor Hayes, Pádraig Cunningham, Michelle Doyle

Department of Computer Science
Trinity College Dublin
Conor.Hayes@cs.tcd.ie

Abstract. CBR (case-based reasoning) has considerable potential for developing intelligent assistants for the World Wide Web. Several case-based intelligent applications already exist on the web but these applications follow a *thin client* model with the intelligence located at the server side. In this paper we explore the advantages of making these applications more distributed. We illustrate the kinds of application where the dialogue with the case-base is *long-lived* and network latency or server load would suggest that some of the processing should be distributed to the client side. We present an architecture for such a distributed CBR system and describe how a case-representation language based on XML can facilitate this distribution. The advantages of adopting an XML case-representation language are interoperability and flexibility of reuse and these are discussed in the second half of the paper.

1. Introduction

Several web based intelligent assistants that use CBR (case-based reasoning) are already in existence (see the web site on Case-Based reasoning on the Web[1] for a list). This illustrates the knowledge engineering advantages of CBR as an effective reasoning strategy for weak theory problems; that is, problems where it is difficult to elicit first principle rules from which solutions may be deduced. A characteristic of these early applications is that they involve implementations of existing CBR technology in a web context – the client has a remote dialogue through the browser with the CBR application at the server side.

The ideas behind our current research have three strands:

1. To extend the incremental CBR (I-CBR) approach to network applications, particularly in areas where predictive feature are expensive or difficult to come by.
2. To examine a distributed architecture for such a system. Since both the client-server interaction is long lived and the information theoretic technique of I-CBR is computationally expensive, response time may be poor.
3. To situate the first two strands as part of a process of creating open standards for case based network computing, case base storage and possible interoperability with non-CBR systems.

There is currently a growing number of *thin client* applications on the web. By 'thin client' we mean an application with presentation logic and simple error handling only at the client end while the server side handles all the business logic and the logic required to integrate the two ends (Wilcox 1997). We argue that this set-up may not be suitable for all online applications, particularly in those situations where the server must be contacted several times as part of an incremental process.

In this paper we present an architecture for distributed CBR that allows some of the case-base processing to be performed on the client side. The objective of this distribution is to improve overall response times for the user. In addition we introduce CBML, an XML application for data represented as cases. The Extensible Mark-up Language, XML, is a simplified subset of SGML which was developed by the W3C XML Working Group to facilitate easy transmission of structured data over existing network protocols (Bray, Paoli & Spergberg-McQueen 1998).

While Distributed CBR has response-time advantages for the user, a case representation language based on XML has advantages of interoperability and ease of reuse. XML is a standard for content on the Internet with XML parsers freely available in Java and in C++. This means that knowledge and data marked up in CBML is readily reusable by other applications such as Intelligent Agents.

In section 2 we describe the type of web-based CBR application where interaction is long lived and networking problems can result in poor response times for the user. In section 3 we present our

[1] http://wwwagr.informatik.uni-kl.de/~lsa/CBR/wwwcbrindex.html

architecture for Distributed CBR that addresses this problem and, in section 4 we introduce the CBML case-representation that will be used to transport cases across the network. Section 5 discusses the benefits of developing a standard mark up language for network based CBR applications.

2. Web Based CBR Applications

In this section we wish to establish the idea that a dialogue with a case-based assistant can be long-lived. This can occur because the user may engage the case-based assistant with only a rough idea of his requirements. These requirements are refined as the user interacts with the system.

Consider the following example of a web based travel advisor system. Two typical cases are shown in Table 1. These cases are taken from the Travel Agents Case-Base[2]. In the scenario we will consider here the user comes to the system knowing that he wants a car-based holiday for three people in three star accommodation. These requirements produce a target case with just three slots filled. If this is passed to the case-retrieval mechanism several tens or hundreds of cases will be retrieved from the thousands of cases available. The user will be asked to refine the query to narrow down the search. This can be done by inviting the user to provide *any* extra information (Kriegsmann & Barletta, 1993) or by indicating to the user the piece of information that will be most discriminating, i.e. most efficient in reducing the set of candidate cases (Smyth & Cunningham, 1995; Cunningham, Smyth & Bonzano, 1998). In the scenario we evaluate here we will consider this second incremental model of CBR (I-CBR) where the retrieval engine indicates to the user the most discriminating feature to provide next. For instance, the system might ask the user to select a Holiday Type from several types offered.

Table 1. Two cases from the travel case-base and a target case.

	Journey149	**Journey162**	**Target Case**
HolidayType:	Recreation	Wandering	-
Price:	922	2588	-
NumberOfPersons:	3	3	3
Region:	BlackForest	Thuringia	-
Transportation:	Car	Car	Car
Duration:	7	14	-
Season:	August	July	-
Accommodation:	ThreeStars	ThreeStars	ThreeStars
Hotel:	"Berghotel Kandel, Black Forest	Hotel Finsterbergen, Thuringia	?

This process continues until a consistent set of cases remains, i.e. a *discriminating* set of user requirements has been determined - or until no cases are available to meet the users requirements. In which situation the user will be invited to backtrack and relax some requirements. The key point here is that the process involves a *long-lived* interaction with the case-base. If the system is implemented as a thin-client with case-base processing at the back end then network latency and server load may produce poor response times for the user. Two existing commercial systems that have these characteristics of long-lived interaction are the OP Amp selection assistant from Analog Devices[3] and the Configuration Agent from Cisco[4].

In the next section we describe an architecture for distributed CBR that allows some of the case-base processing to be distributed to the front end. This will eliminate some of the delay due to network latency, reduce the load on the server and make use of available machine cycles at the client side.

3. Distributed CBR

Our architecture for Distributed CBR is shown in Figure 1. The current thin-client alternative to this has a Browser based interface at the front-end that connects to the server at the back-end; all the case-

---

[2] available at http://wwwagr.informatik.uni-kl.de/~bergmann/casuel/casebases.html

[3] http://imsgrp.com/analog/query.htm

[4] http://www.cisco.com/pcgi-bin/front.x/config_root.pl

base processing is performed at the back-end. In the distributed architecture the CBR engine is downloaded to the client side to allow for the later stages of processing to be performed there.

The detail of the operation of the distributed system is best explained in the context of the travel example presented in section 2. The interface allows the user to describe his requirements. This is marshalled into a partial case-description that is passed to the CBR Front-end as a Query Context. Initially this will be passed to the CBR Back-end to find matching cases. If too many potential matches are found the CBR engine will identify which feature of the matched cases is the most discriminating. This is then passed to the user interface as a Refining Question. The response to this request for extra information is passed to the back-end as a refined Query Context. This process is continued until such time as the Query Context is sufficiently discriminating. At this point, matching cases are passed to the user interface.

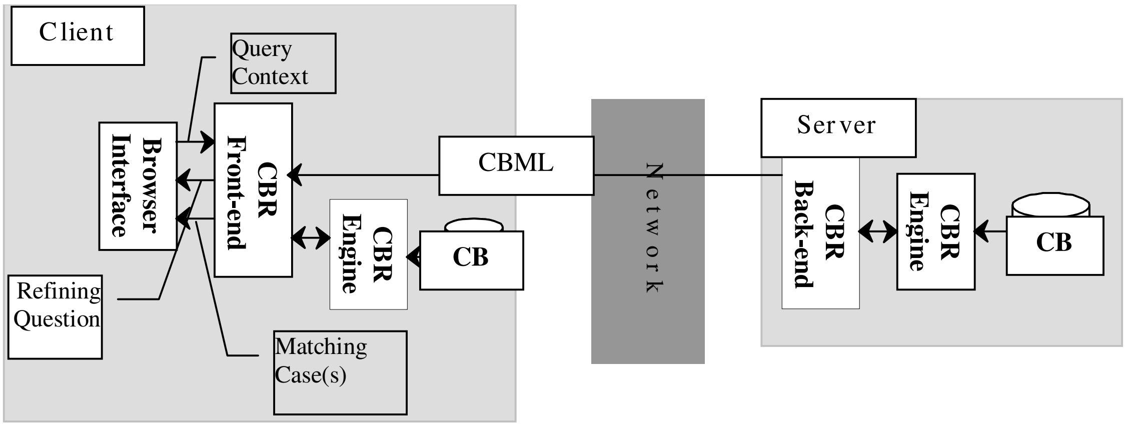

Figure 1. Architecture for Distributed CBR

In this process, as the Query Context is refined, the set of potentially matching cases reduces. The advantage of the distributed architecture is that once this set is sufficiently small it can be passed to the front end where processing can be completed without further interaction across the network. The decision as to when precisely to do this depends on the size of the cases and the response time across the network.

In a Java implementation in an Intranet context, the reduced case-base could be passed as serialised objects to the client side. This would have the advantage that the cases would not have to be re-parsed and loaded into memory after downloading. Unfortunately this will not work across firewalls on the Internet. Consequently we are developing the XML based case representation that can be passed as plain text using the http protocol (which can pass through firewalls via proxy servers).

4. Case-Representation in XML

The background to this section of the paper situates itself in two areas of research. The first is concerned with case representation, a fundamental aspect of the CBR paradigm. What are at issue are methodologies of representing cases in a manner that allows for efficient retrieval, easy maintenance and that provides for transmission over a network. The second research area is connected with devising open standards for intelligent web applications. Such standards provide for inter agent communication and the movement of data through a network from application to application, entailing a truly distributed computing environment.

4.1 Introduction to CBML

As previously mentioned, we are developing a case representation language in XML for use with our system. This language is currently named CBML (Case-Based Markup Language). In this section the design goals of CBML are briefly discussed and some example cases marked up in CBML version 1.0 are given. A detailed discussion of the language is beyond the scope of this paper.

Our main goal was to develop a representation language that could be used by all CBR systems that exchange information across an Internet or Intranet. This language was developed in XML, a meta-language for representing structured data over network systems. The reasons for this will be discussed further in section 5. Another criterion was that the language would provide similar functionality to the CASUEL case representation language (INRECA 1997); therefore its design, as it evolves, will be based in part on CASUEL. However CASUEL is an object-oriented language that makes use of inheritance, and this is a feature that we have not included in version 1.0 of CBML for simplicity. Future versions of CBML will probably be object-oriented so that more complex cases can be represented. As it stands, CBML is a simple, flat feature-value representation. A case-base marked up in CBML consists of two main files[5] - one describing the structure of a case in this domain, the other containing the cases themselves.

The first file (`CaseStruct.xml`) contains information about the features in the case-base - their type constraints, weights etc. The extract below shows three features from the previously mentioned Travel Agents case base, now marked up in CBML version 1.0.

```
<?XML version="1.0"?>
<!DOCTYPE domaindef SYSTEM "CaseStruct.dtd">

...

<slotdef name = "JourneyCode">
        <type a_kind_of="integer"/>
</slotdef>

<slotdef name = "Holiday">
        <type a_kind_of="symbol">
         <range><enumeration>Arbitrary Active Adventure Bathing City Diving
                            Education Language Recreation Skiing Shopping
                            Surfing Wandering</enumeration>
         </range>
        </type>
</slotdef>

<slotdef name = "Duration">
        <type a_kind_of="integer">
         <range>
                 <interval><start value="1"/><finish value="56"/></interval>
         </range>
        </type>
</slotdef>

...

</domaindef>
```

The first two lines of the above example simply state that the version of XML being used is 1.0, and that the DTD (document type definition) can be found in the file `CaseStruct.dtd`. A full explanation of DTDs is outside the scope of this section, but in essence, a DTD allows you to define the tags to be used in your language. In the DTD you also indicate the permitted contents of each tag (either character data, or another nested tag), and its allowed attributes. In the above example we can see that `<slotdef>`, `<type>`, `<range>` and `<enumeration>` are some of the tags defined in CBML. The `<enumeration>` tag contains character data while the `<range>` tag contains either the `<interval>` tag or the `<enumeration>` tag. The `<slotdef>` tag has an attribute called `name`. An XML document for which there is a DTD, and which conforms to that DTD, is termed "valid". A partial DTD for the `<slotdef>`, `<type>` and `<enumeration>` tags is given below (taken from `CaseStruct.dtd`).

```
:
<!ELEMENT slotdef (type, weight?, constraint?)> # consists of one type tag, weight tag
                                                # and constraint tag are both optional
<!ATTLIST slotdef name  ID #REQUIRED>           # slotdef has attribute "name"
<!ELEMENT type (range?)>                        # type consists of optional range tag
                      # type has attribute "a_kind_of" with certain allowed values
<!ATTLIST type a_kind_of  (integer|symbol|ordered_symbol|string|real) "symbol">
:

<!ELEMENT enumeration (#PCDATA)>

:
```

In the example overleaf, we see partial contents of the second file, `CaseBase.xml`. Because the DTD is so short, it is enclosed within the file instead of being referenced as an external file. This example gives a brief indication of what cases marked up in CBML will look like.

[5] There is a third file, the DTD (document type definition) for the case structure file. This will be explained later

```
<?XML VERSION="1.0"?>
<!DOCTYPE cases [<!ELEMENT cases (casedef+)>
                 <!ELEMENT casedef (attributes, solution)>
                 <!ATTLIST casedef casename ID #REQUIRED>
                 <!ELEMENT attributes (attribute+)>
                 <!ELEMENT attribute (#PCDATA)>
                 <!ATTLIST attribute name CDATA #REQUIRED>
                 <!ELEMENT solution (#PCDATA)>
]>

<cases>
<casedef casename="n1">
<attributes>
        <attribute name="JourneyCode">1</attribute>
        <attribute name="HolidayType">Bathing</attribute>
        <attribute name="Price">2498</attribute>
        <attribute name="NumberOfPersons">2</attribute>
        <attribute name="Region">Egypt</attribute>
        <attribute name="Transportation">Plane</attribute>
        <attribute name="Duration">14</attribute>
        <attribute name="Season">April</attribute>
        <attribute name="Accommodation">TwoStars</attribute>
</attributes>
<solution>Hotel White House, Egypt</solution>
</casedef>
...
</cases>
```

5. Advantages of CBML

The advantages of using CBML as a standard are rooted in the strengths afforded XML. XML is an extensible mark up language that has been designed to facilitate the traffic of complex hierarchical data structures over network protocols. Several XML applications have already been produced for the exchange of data particular to specific disciplines and industries.[6] For instance, The Open Trading Protocol (OTP) has been developed for retail trade over the web by the OTP Consortium, an interest group for internet commerce (OTP Consortium 1998).

Once an XML document has been parsed with its DTD, any XML compliant application can "understand" the semantics of the data contained within. Thus data can be represented and exchanged independent of the software at either end of transmission. Jon Bosak, Sun's Online Information Technology Architect and Chair of the W3C XML Working Group views XML as a technology complementary to the platform independent philosophy of the Java language, extending the computing potential of the latter particularly in the scenario of distributed client side processing and web agents (Bosak 1997).

As an XML application, CBML will provide an SGML compliant standard for storing and exchanging case bases. A case base will simply be just another form of data representation, with its own particular DTD. Thus it will be available for processing to other XML compliant applications, whether CBR based or not. This hopes to avoid the current scenario where case representation is application dependent. For instance, case data delivered to the desktop will be available for local computation by a variety of applications. The data can be read by the browser, then delivered to a local application for further viewing or processing, or the data can be manipulated through script or other programming languages using the XML Document Object Model. In the distributed model outlined earlier, cases are delivered to the client end for further processing. Once a successful match is found the case solution can be delivered to a local application for testing.

Data represented in the form of structured cases can be viewed in multiple ways. A local case base can be presented in a variety of ways through the employment of style sheets. This allows for multiple visual representations of hierarchical case structures. Such views would prove important for case editing tools.

XML enables granular updating. Thus, in a distributed CBR environment, servers do not have to dispatch an entire subsection of the case base to the client every time there is a change. The server holds a profile of what has already been sent to the client and only resends the changed element of that

[6] Summer Institute of Linguistics web page. *XML: Proposed Applications and Industry Initiatives.* http://www.sil.org/sgml/xml.html#applications

dispatch. Furthermore XML facilitates late binding of presentation: Using a CBML format, would allow centralized case data to be quickly updated.

6. Conclusion

We have presented a distributed architecture for CBR motivated by a need to move processing to the client side in order to improve interactive response times. We have introduced CBML, an XML application, as a protocol for enabling distributed CBR.

Implementing a Case Based Markup Language opens the door to distributed CBR computing over any network. The web-based format of XML would allow local case-bases to be updated in a granular fashion, only sending changed elements from the server to the client. Since data delivered to the user's desktop can be viewed in multiple ways, a client GUI receiving CBML data could offer the human reasoner, a visual representation of the local case base, with the possibility of simple editing. Furthermore, a client CBML application could possibly process any case base it downloads, once it is deemed valid by its parser. Such a scenario is the ideal outcome of the philosophy of open standards informing the development of XML. To look further into the future, the use of such open standards would be a key to inter-application communication of the web, paving the way for distributed hybrid tools over the Internet. Indeed, the lack of an open standard for representing semantic data up to this point has been a stumbling block for Agent Engineering on the Web (Petrie 1996).

Using cases based on the proposed CBML standard poses the possibility of allowing mobile agents access to large repositories of case storage. Within this view of web based agent engineering, a networked CBR application, though self sufficient within its task domain, is in fact one node in a potentially broader application network.

7. References

1. Bosak, Jon. (1997) *XML, Java, and the future of the Web,* Sun Microsystems, available at http://sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm . Published in W3 Journal, No.4: Fall 1997: XML: Principles, Tools, and Techniques
2. Bray, T,. Paoli, J., Sperberg-McQueen, C. M. Eds. Extensible Mark-up Langauge*, W3 Consortium recommendation paper* . Feb. 1998, http://www.w3.org/TR/1998/REC-xml-19980210
3. Cunningham P., Smyth B., Bonzano A., (1998) An Incremental Retrieval Mechanism for Case-Based Electronic Fault Diagnosis, to appear in *Knowledge Based Systems*.
4. INRECA consortium.(1994). Casuel: A Common Case Representation Language, available at http://wwwagr.informatik.uni-kl.de/~bergmann/casuel/CASUEL_toc2.04.fm.html
5. Kriegsmann M, and Barletta R, (1993) Building A Case-Based Help Desk Application, *IEEE Expert,* 8 18-26.
6. Microsoft, Frequently Asked Questions About Extensible Mark-up Language (XML). The URL is http://www.microsoft.com/xml/xmlfaq.html
7. Petrie, Charles J.,(1996) Agent-Based Engineering, The Web and Intelligence. *IEEE Expert Vol. 11, No. 6. December 1996.* Available at http://cdr.stanford.edu/NextLink/Expert.html
8. Smyth B and Cunningham P, (1995) A Comparison of Incremental Case-Based Reasoning and Inductive Learning in *Advances in Case-Based Reasoning*, Lecture Notes in Artificial Intelligence, Haton J-P, Keane M, and Manago M, eds., Springer Verlag, 151-164.
9. Summer Institute of Linguistics web page. *XML: Proposed Applications and Industry Initiatives.* http://www.sil.org/sgml/xml.html#applications
10. The Open Trading Protocol Consortium Internet (1998) *Internet Open Trading Protocol Specification, parts 1& 2.* Available for download at http://www.otp.org:8080/
11. W3C Working Draft 09-Dec 1997. Document Object Model Specification. URL: http://www.w3.org/TR/WD-DOM-971209/
12. Wilcox, Steve, (1997) Designing Thin Java Client Applications for Network Computers. *Aviteck, LLC 1997*. http://java.sun.com:81/javareel/isv/Aviteck/ThinClientWP.html