CBR Net :- Smart Technology over a Network

Michelle Doyle
Maria Angela Ferrario
Conor Hayes
Dr Padraig Cunningham
Dr Barry Smyth

Department of Computer Science
Trinity College Dublin
University College Dublin

Abstract. CBR (case-based reasoning) has considerable potential for developing intelligent assistants for the World Wide Web. We examine intelligent applications already existent online and find that they can be divided into agent applications and thin client applications. Several case-based intelligent systems already exist on the web. These applications follow the thin client model with the intelligence located at the server side. In this paper we explore the advantages of distributing the processing load to the client side. We present an architecture for such a distributed CBR system for both an Internet and intranet application and describe how a case-representation language based on XML can facilitate this distribution. Adopting a standard case-representation language offers the possibility of communication between different applications, and agent-like interoperability.

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 CBR to network applications, particularly in areas where predictive features are expensive or difficult to come by.
2. To examine a distributed architecture for such a system. Client-server interaction can be long lived and the retrieval techniques are often computationally expensive. Therefore, response time could benefit from a distributed system.
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.

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

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.

After a brief overview of CBR technology (Section 2), we review intelligent systems currently used on the web (Section 3), including the thin client systems favoured by CBR implementations. In Section 4 we present our architecture for Distributed CBR as an alternative to the thin client model. We then suggest a standard case representation language based on XML as a feature of our distributed model (Section 5). Two proposed applications are introduced in Section 6: Ulysses, an Internet based event planner, and EBMT an online translation aid suitable for an Intranet.

2. Background Information On CBR

2.1 The roots of CBR : AI and Cognitive studies

> *"[The automation of] activities that we associate with human thinking, activities such as decision making, problem solving, learning…" [Bellman, 1978]*

Artificial Intelligence (AI) is a branch of computer science that investigates and demonstrates the potential for intelligent behavior in digital computer systems. From the beginning AI research (Minsky, 1961; Turing, 1950) has attempted to replicate human problem solving competence in machines.

Case-based reasoning (CBR) is a recent approach to problem solving in Artificial Intelligence (AI) research. It's an AI technique that places emphasis on the use of previous experiences to solve problems.

> *A case based reasoner solves new problems by adapting solutions that were used to solve old problems [Riesbeck and Shank, 1989]*

CBR studies also have their roots in research into Cognitive Science: the work of Schank and Abelson in 1977 is usually held to be the origins of CBR. According to Schank and Abelson, our general knowledge about situations is recorded in the brain as scripts, which allow us to set up expectations and perform inferences.

> Sheila went to a Thai restaurant. She ordered King prawns. She left the restaurant very satisfied.

According to Schank and Abelson, our knowledge about restaurants allows us, firstly, to predict that Sheila will do what is normally done in restaurants and, that she will play the role of a customer. Though it is not mentioned, we presume that she ate her meal and that she paid the bill before leaving the restaurant.
Our general knowledge (organised in scripts) allows us to explain the connections and fill in the missing details of the story. However later findings showed that this is not a complete theory of memory representation (a person may mix scenes from similar situations, e.g. going to a doctor's office and going to dentist's office).
It was observed that people commonly solve problems by remembering how they solved similar problems in the past. From this, Case based reasoning was developed as a methodology that judges a new problem by comparing it to already classified cases.

2.2 CBR as a problem solving paradigm

Case-based reasoning is a problem-solving paradigm that is in many ways different from other AI approaches. Traditional AI approaches rely on general knowledge of a problem domain, and tend to solve problems on a first-principles, or "from scratch" basis. CBR systems solve new problems by utilising specific knowledge of past experiences, and this knowledge is encoded within a corpus of previous problem solving episodes called a case-base. Another important difference is that CBR supports incremental, sustained learning, since a new experience is retained each time a problem has been solved, making it available for future problems.
This is similar to how people act when solving real life problems. For example:

> A client gives a travel agent a general description of the type of holiday that he is interested in. It is up to the travel agent to suggest the specific one that best

> meets the requirements of the customer. To do this, he looks up a catalogue of holiday packages and finds one closest to the customer's requirements. This may not be exactly what was described in the first place, maybe some details are different, but the suggestion may be close enough to satisfy the client.

CBR is a cyclic and integrated process of solving problems, learning from the experience, then solving new problems. This cyclical process is often described as comprising of four “REs”:

1. REtrieve the most similar cases.
2. REuse the cases to attempt to solve the problem..
3. REvise the proposed solution if necessary.
4. REtain the new solution as a part of a new case.

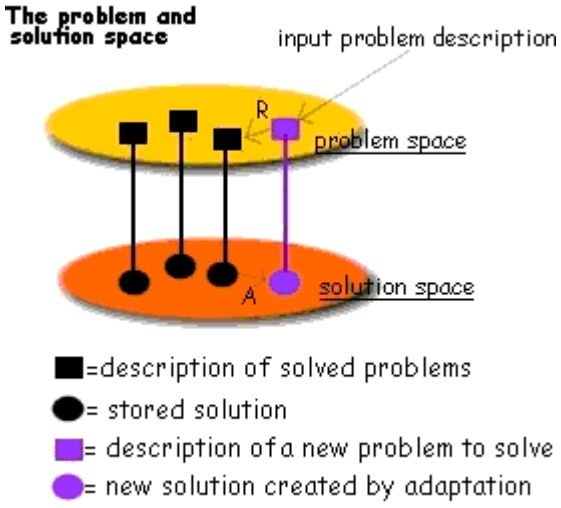
A new problem is matched against cases in the case-base, and one or more similar cases are retrieved. A solution suggested by the matching cases is then reused and tested for success. Unless the retrieved case is a close match, the solution will probably have to be revised, producing a new case that can be retained.

2.3 The CBR Cycle

A case is a piece of knowledge representing an experience. This piece of knowledge can be an account of an event, a story or some record and usually comprises:

1. The problem that describes the state of the world when the case occurred.
2. The solution that gives the derived solution to that problem.

The CBR cycle starts with the description of the new problem to solve been positioned in the problem space.
During the REtrieval (R) process, the system first *identifies the features* of the problem by processing the input descriptors and trying to understand the problem within its context. When a set of best matching cases is found, the system applies *similarity metrics* to assess the degree of similarity of the cases found and the new case submitted. Finally *the best matching case* is



selected and retrieved with its relevant solution. Often, when a case is REused, an *adaptation process* (A) is required to take account of the difference and readapt the solution of the retrieved case to the new case.
The system, then REvises (evaluates) the case solution generated by reuse in the real environment (outside the CBR cycle) and then, if successful, it *learns* from the success (case REtainment). Otherwise the system repairs the case solution using domain-specific knowledge.
Effective learning using CBR techniques requires a well defined set of methods in order to *extract* from the case which information to retain as well as the form it is to be retained in. It is necessary to know also how to *index* the case for later retrieval and *how to integrate* the new case in the memory structure.

CBR avails of indexes to make the search faster and more efficient. Indexes are computational data structures, used in many database systems to speed up the retrieval process. It's crucial that the features of each case are selected and effectively represented, identifying which ones are to be used for retrieval (indexed information) and the ones to be used when displaying the case (unindexed information).

The following table (Table 1.1) is an example case description for representing a Restaurant; it comprises the following indexed and unindexed features.

Table 1.1

Indexed:	**Unindexed**
Name: Escape Location: Bray Type: Vegetarian Style: Casual Price: Cheap Closing day: Monday Advanced booking: Yes Party-facilities: Yes Takeaway: No	Address:1 Brennan's Terrace Tel: 2866755 Seats: 58 Hours: 12.00/22.30 E-mail address: escape@res.ie URL: www.escape.ie Picture: ../images/escape.jpeg Transports: Dart; Other: bring your own wine

The telephone number is not a feature that a potential customer would consider when selecting a restaurant, so that is more likely to be classified as an unindexed feature. On the other hand, the location and the cost appear in the indexed features because they are important variables in the decision.
However, addresses and telephone numbers are still pieces of information needed by the users.

2.4 Current Implementations of CBR Technology

Areas of CBR applications: Case based reasoning has proven itself to be a methodology suited to solving weak theory problems, that is, problems where it is difficult or impossible to elicit first principle rules from which solutions may be created. In the sphere of industry, these problem areas have very often been solved by one or two human experts whose experience with similar problems have made them best qualified to provide new solutions. This effectively means that industries in which such *weak theory* problems occur are dependent on the knowledge collated by a small number of individuals, and are vulnerable should these human experts depart. The number of commercial CBR applications is extensive and growing fast. Existing CBR applications can generally be classified into two main categories (Altoff et al. 1995): classification tasks and synthesis tasks.

1. **Classification** tasks match a new case against those in the case-base in order to determine which type of case it is. The solution from the best matching case is then reused (with or without adaptation).The followings are the most common sub-categories of classification-type applications
   - **Diagnosis**: medical diagnosis, help desks (e.g.: *Compaq*: customer help desk and intelligent computer reference manual; *Cybermedia*: First Aid 97 PC diagnostics).
   - **Prediction**: forecasting system failure (e.g.: *Matra Space Corporation*: Satellite fault diagnosis)
   - **Assessment**: estimation of project costs, risk analysis for banking or insurance (e.g.: *American Express*: Credit card risk assessment)
   - **Process control**: manufacturing equipment (e.g.: *Nestlé*: Process control)
   - **Planning**: travel plans or work schedules (e.g.: *University Hospital Munich*: Medical diagnosis and personnel scheduling)

2. **Synthesis tasks** attempt to create a new solution by combining parts of previous solutions. There are fewer examples in this category due to their complexity. They involves tasks such as:
   - **Design**: the creation of a new artefact by adapting element of previous ones (i.e.: *NCR*: customer support and design of computer interfaces; *Shai*: architectural engineering)
   - **Planning**: creation of new plans from elements of old ones.
   - **Configuration**: creation of new schedules from old schedules.

2.5 Industrial Applications of CBR

It is vital in business to maximise the use of intellectual assets, to manage and store the collective knowledge of any enterprise. As mentioned earlier, efficiency in this

area is difficult if the work is defined by *weak theory* problems which require the judgement of an expert who can recall and re-adapt solutions to similar past problems. Industries in which such "weak theory" problems occur are overly dependent on the knowledge of one or two experts in the field (Watson 1998).
This knowledge management problem becomes next to impossible if the enterprise requires people to work apart from one another, but on similar problem types.
CBR is best suited to any situation in which there is the need to ensure efficiency in the communication flow and the availability of specialised knowledge to expert and non-expert users.

A wide range of enterprises avail of CBR for a variety of tasks, the ones mentioned above are just a very limited selection of what is available on the market. Although the predominant use of CBR is still within customer service/after sale assistance (*Black and Decker, London Electric, Nokia*; *Microsoft* has embedded CBR technology within the intelligent help system of its Microsoft Office 95), the number of applications in different areas is growing at an incredible pace. The entire e-commerce scenario is another area that will benefit from CBR applications

3. Intelligent Web applications

The growth of intelligent systems on the web is driven by the need to assist users in finding or configuring information relevant to them At the root of this paradigm is a fundamental of all computing: using machines to speed up, assist or intervene in processes currently carried out by humans, thus freeing people to carry out higher level tasks or creative tasks. If sales and support tasks are to be automated on the web then web applications must have some intelligence to compensate for the absence of a human agent in the process.

For instance, in the realm of electronic commerce, if the web is to more than a digital mail order catalogue it must provide a means whereby potential shoppers can choose or configure, under the advice of an intelligent monitoring system, the product or service that is suitable to them. In this context, information constitutes both the purchasers profile, the particular advice given during the transaction and the specific data or services purchased.

Intelligent Internet systems can be roughly divided into two camps: Agent based systems, and application type systems.

3.1 Agent based Systems

"An agent is a computational entity which:-

- *Acts on behalf of other entities in an autonomous fashion*
- *Performs its actions with some level of proactivity and/or reactiveness*
- *Exhibits some level of the key attributes of learning, co-operation and mobility."* (Green, Hurst, Nangle, Cunningham 1997)

Internet Agents are programs that reside on servers and access the distributed on-line information on the Internet to perform tasks on behalf of users without direct user interaction. They can be roughly categorised into the following groups:

- Search agents: Web Robots that traverse the web by following hyperlinks, cataloguing its content. They usually have a built in strategy for prioritising links to follow as they arrive at each URL.
- Information filters: Gathers content on behalf of the user according to the user profile. These agents then filter the gathered information according to user preferences.
- Service Agents: which act as information brokers, providing specialised services to users. Firefly (www.firefly.com) is a service agent which uses collaborative filtering techniques to recommend new books and music to subscribers.

3.2 Application type systems:

These are generally thin client systems, where the application is distributed over three tiers. The GUI is situated at the client end, with business logic running on a server that connects to a database. Whereas the agents previously described act independently and anonymously, the type of applications being described here are expert systems that require user input at the client end. Agent systems index and filter information already provided while thin client systems provide information in the form of solutions generated on the fly by the business logic residing on the server. In many ways this scenario harks back to the mainframe computing model, where numerous clients were hooked up to a centralised system.

3.3 Current Web CBR

CBR systems are currently implemented on the web as thin client applications.[2] Most use HTTP forms to submit query data to a server on which a CBR engine resides. Cases on the server side are stored in a variety of proprietary formats. Users submit

[2] The University of Kaiserlautern hosts a page listing current CBR web applications: http://wwwagr.informatik.uni-kl.de/~lsa/CBR/wwwcbrApps.html

generalised descriptions of their problem and the server returns specific cases whose descriptors best match the original submission.
As discussed in Section 2.5, CBR is suited to providing expert advice in domains that traditionally required the judgement of an expert who could recall and readapt solutions to similar past problems. Implementing CBR over a network makes expert advice available to all who need it irrespective of location. We examine three types of application in use on the web that benefit from the CBR paradigm:

- Assistants for electronic commerce
- After Sales Support Service
- Specialised Search systems

Assistants for electronic commerce: The facility of Case Based Reasoning to provide suggestions imprecise queries has not yet been exploited to any great extent on the Internet at present. Such systems have the potential to provide selling suggestions to clients as a shop assistant might, based on the particular needs of a client. It is the role of the shop assistant to translate a generic description given by a customer into a real item on the shelf.
In the world of high street retailing, a successful sale occurs when a customer realises that your stock is closest to what she had in mind the first place. If we apply such a scenario to an online outlet the benefits of a CBR assistant becomes clear. For instance, somebody perusing an online jazz store might be looking for 1940s swing jazz that might include several favourite players. Such an ensemble, however, may never have played together. A CBR search engine might return details of recordings that some of the jazz personnel made together during the 1940s, thus providing the jazz fan with an opportunity to purchase recordings that almost meets their initial specification.

Jazz, is a good example in this context because Jazz buffs are so particular about genre, instrumentation and personnel.[3] The number of possible configurations of player line up, instrumentation, genre and recording label would be too large to make a rule based system feasible to construct. CBR is ideally suited to dealing with systems where users may submit an infinite number of configurations, but in reality where there is a only finite number of solutions available. An example of an Industrial system currently implemented on the Internet is **Analog Device's** search system for operational amplifiers, where a user can enter the configuration required of an amplifier and a CBR engine returns the details of an amplifier best suited to the task.[4]

[3] Whereas Rock music is much less concerned with this. On line collaborative filtering techniques have been proven to work well in recommending new recordings to rock fans.

[4] http://www.imsgrp.com/analog/query.htm

This site does not yet include a facility to buy online. However, the facility of CBR to steer potential buyers towards real articles or services based on imprecise queries makes it well positioned to take advantage of the growth in the electronic commerce.

After Sales Support Service: Web help desks: For complete customer satisfaction it is necessary that a company is able to meet the requirements of the client both in terms of high-quality products and high-quality customer service. Good after-sale assistance implies efficiency in knowledge management and decentralization of expert information. The efficiency of a Help Desk service can be maximized, if supported by a type of expert system able to perform diagnostic problem solving and easy to upgrade when new problems and their solutions are encountered.
This means making continuously upgraded specialised knowledge available to expert staff and home users. These are exactly the characteristics of a CBR system, it's therefore easy to understand why more and more companies chose to implement their help desk using CBR. Once a CBR help desk is made available on-line, it becomes an automated 24-hour 365-day fast customer service, easy to interact with and accessible at any time of the day

One of the first and probably one of the most successful on-line CBR applications (Awarded at the Voice Europe 1996 Exhibition in London as "Highly commended") is the *GizmoTapper*[5], Broderbund's Software helpdesk. Up to the end of 1995, Broderbund's primary support tool for the telephone help line services was a DOS-based key word search product. It was a system that retrieved without discrimination every word mentioned in the problem description, leaving the technical support staff to interpret the huge amount of information retrieved.
The time delay was off-putting and the overall efficiency of the service was so poor that both users and technicians were highly frustrated. As the technical support staff threaten to quit while Broderbund was launching a major new game *Myst* (Christmas 1995), the company decided to implement an automated 24 hours Internet/Web-Based customer support service.

[5] http://www.broderbund.com/support/gizmo.html.

They realised that many of the customers would have preferred to access a Web-site to obtain help, instead of using the phone. A problem resolution case base was therefore offered directly to the customers leaving the telephone line free for customers without Internet access.
The cases the *GizmoTapper* uses are the same as those used by the customer support representatives on the telephone line. The service provided is consistent in both mediums and any changes in the case-base are automatically available at both ends

Broderbund, without expenses of additional personnel or extended hours, provides a high quality customer service. Many on-line help desk followed the *GizmoTapper* (*LucasArt Entertainment Co*; *Cisco and 3Com*) while the area for CBR on-line application is extending to other domains.

Online Search: The CBR online search applications are a variation of the guidance system we have described. The search space is limited to a particular domain, for instance jobs. CBR Job is a search application designed by the University of Kaiserslauten,[6] which matches the computer skills entered by a user to jobs vacancies.

3.4 The Client Server Model

This section will examine the thin client - server model which is the predominant model for intelligent applications, including CBR, on the web.

There is debate as to what exactly constitutes a *thin client* and the degree of thinness inherent in the system. In the *Model View Controller* design pattern, there is a clear separation of the presentation of the application and the application logic in the thin client architecture.[7] The *View* is the presentation component of the system. It includes the rendering of windows, panels, text fields, and other GUI widgets. In this system the *View* constitutes the work performed by the client, whereas the *Model* component resides on the server and consists of all of the application domain specific objects and behaviours otherwise known as the application or business logic. The *Controller* comprises the logic required to make the Model and View components work together. In the 3 - tier thin client architecture presented, the Controller resides on the server.

[6] http://minsk.informatik.uni-kl.de:8100/launch/JobbQueryInterface

[7] ibid. pg. 2.

Distributing the view and controller components between client and server requires a protocol framework so that client view information can be received by server controllers and vice versa. Using custom sockets for this communication has proven problematic due to firewall restrictions. A protocol framework, however, that uses HTTP to communicate between the client view and the server application avoids this restriction.

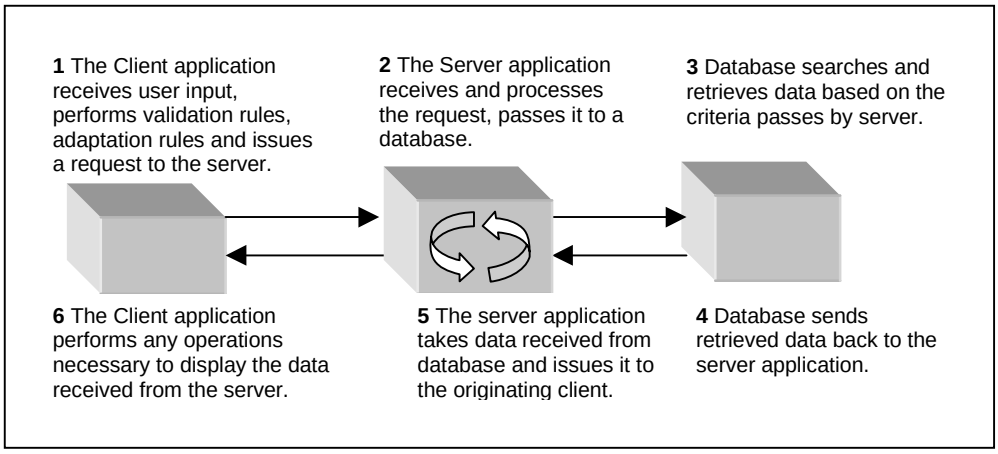


figure 1. A simple schema of the three-tier architecture used by most applet - server applications. Tier 1 is the client, Tier 2 the server application and tier 3 the database.

Java Clients : Java is increasingly being used as the front end in thin client - server applications, [8] particularly in three tier configurations with the second tier consisting of a server application which receives a request from a client, retrieves the appropriate data from a database and issues it to the client for remote display.
Thin client Applications using this schema fall broadly into three categories.

1. Visualisation.
2. Financial packages for home help or updating market data.
3. Knowledge Assistants

As part of the present research, the CBR unit is interested in applications that provide efficient knowledge management solutions. The following examples are thin client Java applications currently being used for delivering personalised client information.

[8] Phil Bartholo. 1997. *Applet Power*. http://java.sun.com:81/features/1997/oct/applets.html

In the sphere of visualisation the client applet acts as an interface for customising and displaying visual information. Because of their size, it does not make sense to download the entire range of graphics available to a visualisation applet, particularly when only a small part of which might be used at any one time. Instead, such an applet issues the server application with a request corresponding to the input data of the user. The server requests the correct image or series of images from the database and sends them to the applet for visualisation. Such a situation works for enterprises that require customised views of their products or services, such as in the area of design. An applet that is particularly effective in this respect is **Adjacency's** *Outfit Your Land Rover Applet*[9].

figure 2. The Land Rover Applet downloads images of extras specified by the user, as they are needed

The Land Rover Applet allows the customer to visually examine the range of customisable add-ons that come with each of the three land rover models provided. As only one graphic needs to be displayed initially, the applet loads quickly. Each add-on specified by the user is at once requested from the server. Once the image is received by the applet, an image of the newly adorned Land Rover is recompiled and displayed.

In the realm of personal financial management technology, **The Home Account Network**[10] have developed a demo network Java applications that allow clients to perform balance enquiries, review account statements and to transfer funds to and from accounts over the Internet. The Home Account Network uses the Open Financial Exchange (OFX)[11] standard to transfer financial data across the network. This standard was developed as an SGML application, prior to the development of XML, to enable the exchange of financial data and instructions over the internet between customers and Financial Institutions such as banks, credit unions, brokerage firms, mutual fund companies, credit card companies, among others. It allows institutions to connect directly to their customers without requiring an intermediary. As will be

---

[9] Outfit Your Land Rover. http://best4x4.landrover.com/features/outfit/index.html

[10] *The Home Account Network* at http://java.sun.com:81/javareel/isv/HomeAccount/index.html

[11] Norman,Cynthia; Muir Software, Inc. *White Paper on The Open Financial Exchange* at http://www.muirsoftware.com/ofx.html#OS

Detailed Information on the OFX specification can be found at http://www.ofx.net/

discussed in a later section of this paper, the objectives behind the development OFX standard are similar to those behind the XML project, namely to develop an open standard that allows the transfer of particular structured data types over a network. [12]

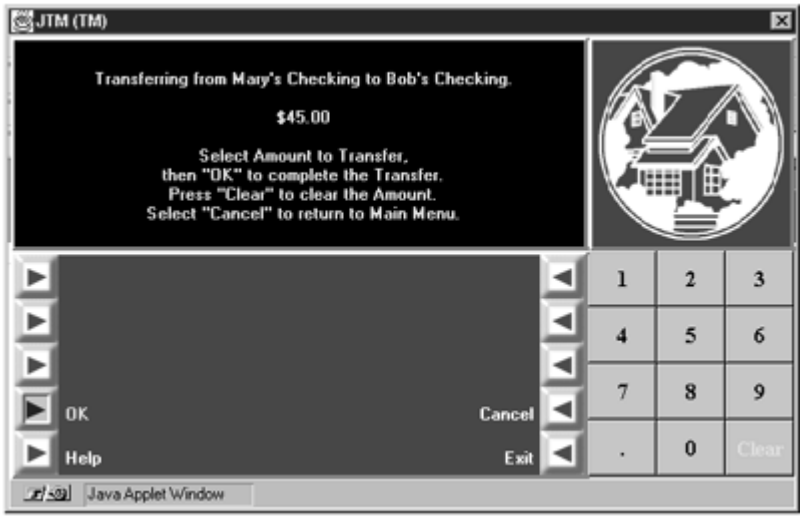


figure 3. The Java Teller Machine Applet developed by The Home Account Network

The thin client systems outlined above offers a schema for implementing a client server system in Java, where the bulk of computing is performed server side. With this in mind the object of current research is to distribute some of the work performed by the server to the client end. The thinking behind this is that the reasoning mechanism behind CBR very often involves the user working in an advisory role, particularly in the Incremental Case Based Reasoning model where the user is offered refining questions, the answers to which allow the retrieval engine to reduce the case base to a small number of well matched cases (Cunningham & Smyth 1994). The target case description, therefore is built up during the case retrieval process and involves constant interaction with the CBR retrieval engine on behalf the user.

Over a network like the Internet where data must pass through a number of servers before reaching a client machine and vice versa, the trend seems to be for thin clients that are growing fatter or have the ability to put on weight under certain circumstances (Knapik & Johnson 1998) This trend is being facilitated by technologies such as Java and, as a later section will discuss, XML. The degree of weight exhibited by a client depends upon the network environment and the work being undertaken. In an Internet situation, bandwidth considerations mean that applet size and the incremental data being passed to and from it must be commensurate with the average client's Internet connection and the job being undertaken (Wilcox 1997). The user may be willing to

[12] The syntax of OFX differs from that of XML in that OFX omits end-tags. However, XML parsers should be able to correctly interpret documents without a Document Type Definition (DTD).

wait 10 minutes for an initial applet/data download if the application is useful throughout much of the day or if the information received is valuable. As with the applets discussed earlier, downloads may be relatively small once the initial application is loaded. With an intranet scenario, issues of network latency are much less pronounced.

In the next section we propose an online CBR system, which distributes some of the processing to the client applet. Since a networked CBR system could potentially have many clients connecting simultaneously and as case retrieval can be a computationally intense process, distributing as much work as possible to the client side could possibly speed up the reply time for all users on the network.

4. Proposed CBR for distributed systems

In this section, we firstly explain why we feel that performing some of the processing on the client side would be necessary and outline our proposal for a distributed CBR architecture. We also propose a model that makes use of case-bases distributed across client nodes. Both implementations will be in Java, so there will be no cross-platform issues to deal with.

As a beginning to our discussion, 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[13]. 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

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

discriminating feature to provide next. For instance, the system might ask the user to select a Holiday Type from several types offered.

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[14] and the Configuration Agent from Cisco[15].

	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	?

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

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.

4.1 Architecture for Distributed CBR

Our architecture for Distributed CBR is shown in figure 4. This is proposed as an alternative to the thin client architecture described in section 3, where all the case-base processing is performed at the back-end. In the distributed architecture the CBR

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

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

engine is downloaded to the client side to allow for the later stages of processing to be performed there.

The details of the operation of the distributed system are 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.

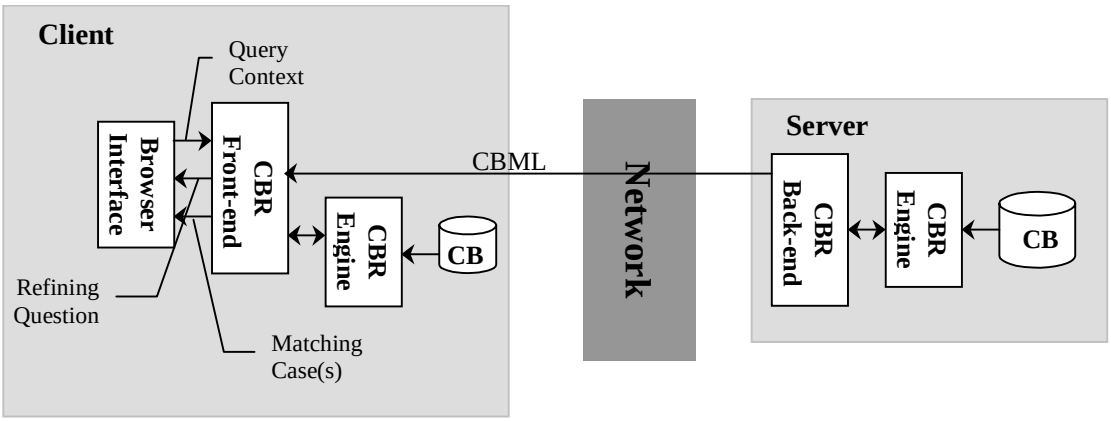


figure 4. Architecture for Distributed CBR

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.

This process is based on the incremental model of retrieval described in (Smyth & Cunningham, 1995; Cunningham, Smyth & Bonzano, 1998). Incremental CBR (I-CBR) is suited to diagnostic and classification situations where a number of free case features are initially given, but for which more expensive predictive features must be submitted in order to retrieve a usefully accurate subset of the case base. The process is incremental in that the user is offered a series of refining questions based on the predictive features of a case. As described above, we are using this model of retrieval to reduce the set of matching cases to a "small enough" number so that it can be sent

to the client side for further processing, thereby reducing server load and ending interaction 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).

4.3 Distributing the case-base across client nodes

Parallel to our investigation of using an incremental retrieval mechanism to aid distributed processing in a client-server situation, we are also investigating both the feasibility and the potential benefits of distributing the case-base across client nodes, as well as maintaining it on a central server. Having a local copy of a portion of the case-base on the client side would hopefully improve system performance and robustness. It would also reduce substantially the load on the server and the network. This however raises the question of which portion of the case-base to store. A naive solution could be to store cases on the client machine that are similar to the kind of problems the user has submitted in the past. However this would only guarantee optimal solutions for a small portion of the queries - i.e. those that are like the previously submitted queries. We need to choose the cases for the client-side case-base such that it is competence-rich. To achieve this, we are intending to exploit work undertaken in the area of Case-Base Maintenance (Smyth 1998).

The two factors that affect the competence of a case, according to Smyth, are its *coverage* and *reachability*. The coverage of a case is in essence the set of cases in the problem space which can be "solved" by that case. The reachability of a case is the set of cases which can "solve" it. Together these sets can be used to deduce whether a case in the case-base is redundant or essential - redundant cases can be deleted to improve storage and retrieval efficiency without reducing case-base competence.

Smyth (1998) categorises cases as either pivotal, support or auxiliary. The categories of interest to our work are pivotal cases and support cases. A pivotal case is one that covers a region of the problem space otherwise uncovered. This implies that its reachability set contains itself alone- in other words, no other case can solve it. Since this means it is not in the coverage set of any other case, the coverage set of no other case in the case-base subsumes its coverage set. Support cases are a group of cases with the same coverage sets. Therefore they all subsume each other so all of them but one can be deleted without reducing the competence of the case-base.

We are not concerned with competence-preserving deletion issues here, however. We are concerned with the implications these categorisations have for distributing a case-base across client nodes. These implications are also described in (Smyth 1998) and in essence the idea is to include all the pivotal cases and one case from every support group in the client-side case-base. This case-base would then have the same coverage as the entire server-side case-base. The original idea - of including cases that were similar to cases previously input by the user - could also be used; a few cases considered "typical" for this user could be included in order to speed up retrieval time.

Our current intention is to perform retrieval on the client-side case-base initially, and if the nearest retrieved case is below a certain threshold of similarity[16], to initiate server-side retrieval. This means that the server is only contacted when necessary. It may not be necessary to perform retrieval on the entire server-side case-base as suggested in (Smyth 1998) - it might be enough to search for a better match within the coverage and reachability sets of the case retrieved on the client side. If this assumption is true, then the amount of processing on the server side would be minimal, which would be in keeping with our purpose. However, since the client-side case-base would in effect offer the same coverage as the server-side case-base, it might be safe to expect that acceptable solutions would be found on the client-side.

The ideas discussed in this section are still theoretical, but they offer an interesting direction for our research. One possible implementation problem which already suggests itself is that of how cases could be stored on the client machines, as applet security does not allow writing to files on client machines without the system manager's permission[17]. At this stage it would seem that the model might only work with certain clients known to us who have already given us write permission, and not with any casual client that decides to try out our system via a browser. Thus it would seem that this idea would work best in an Intranet situation, where the predetermined finite number of clients would make it feasible for each client machine to specify special file permissions for our applet. This will be discussed in more detail in section 6, where we describe our proposed Intranet application. Another possible issue is that of updating the various client-side case-bases after new cases are included in the case-base. This would most likely involve computing the competence of the new case and dispatching it to each of the clients (perhaps using a type of "push" technology) if it is considered pivotal. Again, it must be stressed that none of these ideas are at implementation stage yet and therefore might change slightly.

---

[16] This threshold to be decided experimentally.

[17] This would require JDK 1.2, which allows a user to grant an applet from a named server read and write access to certain files.

4.4 Summary

In this section we discussed a possible distributed architecture for CBR that makes use of an I-CBR model, and also another possible distributed scenario where various clients have local case-bases that can be consulted before consulting the main server-side case-base. Our intention is to implement both models and then run some performance tests to investigate both the efficiency of each model and how each of them affects server and network load. We expect that distributing cases across client nodes will be more feasible in an Intranet rather than an Internet context.

5. 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.

5.1 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 5.2. 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[18] - 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. This one file contains all the information that is contained in the `X.types`, `X.slots`, and `X.objects` files in CASUEL (`X` being the domain name by convention). The type definitions are not separated from the slot definitions (unlike in CASUEL), as type definitions always refer to a particular kind of slot, so it seemed less verbose (and more efficient for the parser) to include the type definition implicitly within the type declaration for a particular slot. As in CASUEL, a type can be simple (like integer or string), an enumeration or a range. Three different types are shown below (figure 5).

```
<?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>
```

figure 5. Partial contents of `CaseStruct.xml` file

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

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 some of the tags defined in CBML, including `<slotdef>`, `<type>`, `<range>` and `<enumeration>`. 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 in figure 6 below (taken from `CaseStruct.dtd`).

```
:
<!ELEMENT slotdef (type, weight?, constraint?)>
<!ATTLIST slotdef name ID #REQUIRED>
<!ELEMENT type (range?)>
<!ATTLIST type a_kind_of (integer|symbol|ordered_symbol|string|real)
"symbol">
:
<!ELEMENT enumeration (#PCDATA)>
:
```

figure 6. Partial contents of `CaseStruct.dtd` file

The definition of the `<slotdef>` tag indicates that it consists of exactly *one* `<type>` tag, and optional `<weight>` and `<constraint>` tags. Its attribute list definition (`<!ATTLIST…>`) specifies that it has one attribute, called `name`, that this attribute is a unique identity for a `<slotdef>` element, and that each `<slotdef>` tag **must** specify a value for this attribute. The definition of the `<type>` tag specifies that can optionally contain a `<range>` tag, otherwise it is empty (the syntax for an empty tag is `<tagname/>`, and no closing tag, for example, the type definition for the `JourneyCode` slot on the previous page). The attribute list definition for `<type>` indicates that it has one attribute called `a_kind_of`, that this can only have certain values, and that the default value (if unspecified) is `symbol`. The `<enumeration>` tag is defined as containing *any* parseable character data (`#PCDATA`), instead of nested tags as in the other examples shown.

In figure 7 overleaf, we see partial contents of the second file, `CaseBase.xml`. Because the DTD is so short, it is included 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. The DTD does not deserve a detailed explanation as it is quite

straightforward, but one or two points should be noted. The DTD specifies that the `<casedef>` element has an attribute called `casename`, which is a unique identity for the `<casedef>` element - i.e. all cases have a unique name, a necessary requirement. However the `<feature>` element has an attribute called `name`, which is specified as consisting of character data, but which is **not** a unique identity, as the same value for the `name` attribute will appear throughout the case base.

```
<?XML VERSION="1.0"?>
<!DOCTYPE cases [<!ELEMENT cases (casedef+)>
                <!ELEMENT casedef (attributes, solution)>
                <!ATTLIST casedef casename ID #REQUIRED>
                <!ELEMENT features (attribute+)>
                <!ELEMENT feature (#PCDATA)>
                <!ATTLIST feature name CDATA #REQUIRED>
                <!ELEMENT solution (#PCDATA)>
]>
<cases>
<casedef casename="n1">
<features>
      <feature name="JourneyCode">1</feature>
      <feature name="HolidayType">Bathing</feature>
      <feature name="Price">2498</feature>
      <feature name="NumberOfPersons">2</feature>
      <feature name="Region">Egypt</feature>
      <feature name="Transportation">Plane</feature>
      <feature name="Duration">14</feature>
      <feature name="Season">April</feature>
      <feature name="Accommodation">TwoStars</feature>
</features>
<solution>Hotel White House, Egypt</solution>
</casedef>
...
</cases>
```

figure 7. Partial contents of `CaseBase.xml` file

5.2 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.[19] 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

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

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

6. Proposed Applications

6.1 Web Model: Ulysses, A Proposal For A Possible CBR on-line Application

This section contains a proposal for a possible on-line application that could benefit from the Distributed CBR architecture. The area of application is relatively new and differs from the most common on-line CBR applications (Help Desks or on line searches).

Concepts: ULYSSES is an on line planner in the domain of after-work activities in the Dublin area. The city of Dublin has been chosen both for the variety of entertainment available and the tourist turnover throughout the year.
However, the main target, at least for the short-term purpose of the project, are local users. The number of Internet connections in Dublin area and the exponential growth of the level of computer literacy, are some of the reasons why the Irish capital is ideal for such an application.
It seems that the use of the Internet as a medium for fast communication and money transaction increases the mobility of people. Since the use of the Web cuts down on the time wasted on telephone calls, faxes and traffic jams, people could have more time for going out.
Although there is a great number of on line resources for entertainment, and event guides specific to Dublin City, it seems that the amount of information available can make the decision more difficult.
The idea behind ULYSSES is to provide the users with a simple but fast way of retrieving customised information, thus minimising the time spent searching. The CBR system behind ULYSSES could be an ideal solution in this scenario where an efficient knowledge management system is required.
At the front end, particular attention will be given to the design of the site itself. Internet technologies will be fully exploited in order to implement an intuitive, easy-to-use interface, focusing chiefly on "drag and drop" features together with simple but effective graphics.
It is possible to envisage further applications of ULYSSES to other planning domains. The distributed nature of the system will allow fast and simultaneous connections and will provide the users not only with an easy way of choosing a suitable night out, but also, perhaps, the possibility of booking the selected places from home, the office or wherever.
Intranet versions, using lists of selected pubs, restaurants and others places of interest, could be implemented ad hoc and made available to travel agencies or tourist offices.

How it works: the interaction with ULYSSES roughly comprises up to six stages:

1. Submit description
2. Propose solution (with possible adaptation)
3. Ask user feed back
4. Process user feedback
5. Re-adapt solution
6. Update Case-Base

The user submits a description (1) of what he would like to do for the evening. For instance: he could ask ULYSSES where to find a good place for (a) a cheap Chinese meal, near a cinema showing an action film (b). The film should preferably end on time for the last pint in the pub across the road (c).

The CBR engine on the server retrieves from the case-base containing "pre-cooked" plans (groups of activities for a night out; *Table 6.1.1*), and passes the set of best matching cases to the front-end. If required, once the best matching case is found, it could be adapted (first level of adaptation) to the new situation. The solution is then proposed (4) to the user.

In order to evaluate the proposed solution, ULYSSES will ask for the user feedback (3) which it then processes (4). If the solution is a good one, ULYSSES will update its Case-Base (6) accordingly.

If the user is somehow not satisfied with the solution proposed (maybe in the proposed plan there was a feature that didn't match with the request i.e. a *Tha*i restaurant instead of a *Chinese* one), the system starts a second level of adaptation (5)**.**

It initiates retrieval from a second case-base (the Activities Case-Base) which contains five different kinds of activities (places or events; *Table 6.1.2*): (a) restaurants, (b) pubs, (c) theatres, (d) cinema (e) concerts. The process is passed again to the second step (propose solutions) and it will only terminate once the user is satisfied and/or he decides to quit

Table 6.1.1 **Example case (from the Pre- cooked plans Case-Base)**

```
Activities_to_plan:

 Activity_1
        (
        Event_type= Restaurant;
        Name = nd;
        Type = Chinese;
        Price = Cheap;
        Advanced_booking: Yes;
        Party_facilities: Yes;
        )

 Activity_2:

        (
        Event_type = Movie;
        Genre = Action/Thriller;
        Cinema_is_in = nd;
        )

 Activity_3:

        (
        Event_Type = Pub;
        Name = nd;
        Type = Traditional;
        Price = Cheap;
        Location = D2;
        )

        Day_of_the_week = Monday

 Time_Constraints:

        (
        Time_Start = 21.00;
        Time_Activity_2 = nd;
        Time_Activity_3 = nd;
        Time_End = 00.00;
        )

Solution:

 Activity_1:

        (
        case_ref  = #R_121;
        Matching_Rate =10;
        )

 Activity_2:

        (
        case_ref = #M_302;
        Matching_Rate =10;
        )


 Activity_3:

        (
        case_ref  = #P_109;
        Matching_Rate =9;
       )

OverAll_Feedback_Rate =  9;
```

Table 6.1.2 **Example case (from the Activities Case-Base)**

```
Event_Type = Restaurant;
Case_ref = #R173;
Name = GoodWorld;
Location = D2;
Type = Chinese;
Style = Casual;
Price = Cheap;
Closing day = none;
Advanced booking = Yes;
Party-facilities = Yes;
Takeaway = No;

Address:18 South George Street;
Tel = 6775373;
Seats = 58;
Hours= 12.00/22.30 ;
E-mail address = GoodWorld@res.ie ;
URL = …/restaurants/GoodWorld.htm;
Picture = ../images/GoodWorld.jpeg;
Transports =  Bus n° 18,46,44;
```

Since the client-server interaction is *long lived* we think that the performance will be improved if the system is supported by a distributed architecture. The resulting improvement in the system performance will obviously translate into an increased customer satisfaction. In addition, since the application will be Java implemented and CBML will be used for Cases Representation, it will be easy to readapt the system for a variety of other on-line commercial applications.

6.2 The Intranet model

An Intranet is a network that provides similar services *within* an organisation to those provided by the Internet. For example, a company could use one or more WWW servers on an *internal* TCP/IP network to distribute information within the company. Web server software is used to display the hypertext documents that are published on the organisation's Intranet. This is an efficient and cost-effective way to spread information throughout an organisation.

Intranets have become a major growth area in corporate computing due to the availability of cheap or free commercial browser and web server software. Therefore the application of our ideas to an Intranet would be commercially viable, as well as being practically viable due to the faster network. The area we have chosen to base our demonstration Intranet application in is that of machine translation, specifically, *Example-Based* Machine Translation (EBMT). This application is appealing because

many companies perform localisation tasks these days and machine translation is particularly successful in restricted, technical domains.

Bró na Collins (Collins & Cunningham, 1996) has researched and implemented a very good Example-Based Machine Translation system, and we intend to use her research as a basis for our demonstration application. Collins describes EBMT as "the marriage of MT and Case Based Reasoning (CBR) techniques". It makes use of a "case-base" of previously translated sentences to assist in translating an input sentence. The case-base consists of sets of English and German sentences[20] which are linked together in three ways - at the sentence/clause level, at the syntactic function level (Subjects, objects, verbs etc.), and at the lexeme level. When a sentence is input to the system, it is compared to the same-language sentence in each case on both a lexeme level (same words) and on a syntactic level, and assigned a similarity score accordingly. However, the similarity score calculated here is not the final score for the case under consideration, as it says nothing about the ease of adapting the retrieved case to fit the input sentence. The adaptability of a case is a necessary consideration for case retrieval in this system. If the mapping from the retrieved source language sentence to its target language equivalent were a complex one, adaptation would be quite difficult, as changing the words that are different in the input sentence would probably affect other constituents. Therefore cases with complex source language to target language mappings are not easily adaptable, and cases with higher adaptability ratings but lower "semantic" similarity scores are preferred. The "best" retrieved case is adapted ("reused") automatically by the system, unless a word in the retrieved source language sentence has no discernible link to the target language sentence. The user may be consulted for advice in this situation.

We envisage the distributed case-base model as being particularly suited to an Intranet situation, and that model would hopefully work well with the EBMT system. There should be no problems with storing case-bases on client machines, because there would be a finite set of clients who could all grant our applet read and write permissions to certain files. In addition, it should be interesting to apply Smyth's theories of case competence to the EBMT case-base to see how well they apply in this complex domain. The system will be implemented in Java (the existing EBMT system is implemented in LISP) and the cases may be represented in CBML[21].

---

[20] Collins' system (ReVerb) uses an English-German corpus, but the system is language independent - it makes use of the case-base alone without any domain (language) knowledge

[21] The current version of CBML would be too simple to store the complex cases which exist in this domain, however by the time we are implementing this application we would hope to have already

Unfortunately, the I-CBR model of retrieval does not fit the EBMT model of retrieval. The EBMT system does not ask the user to manually input the values for various features but instead asks simply for *one* input sentence from which it finds the required information automatically, via linguistic analysis. Thus the EBMT system does not have any provision for the concept of incremental user input. If we wish to experiment with the model in an Intranet situation, we will have to use another demonstration application. These issues will be resolved during the implementation phase.

7. Conclusion

In this paper we have reviewed the suitability of Case Based Reasoning to provide solutions in "weak theory" domains. As such we have suggested CBR as a solution to knowledge management problems, particularly in areas where expert information needs to be communicated to non-experts. Case Based Reasoning systems use imprecise query information to return precise solutions derived from previous experience. Hence, they will prove important in configuring online assistants for buyers, and in after sales support desks where expert assistance is required to make a sale or troubleshoot a technical problem.
We have examined current intelligent applications on the web and found them to be either essentially agent based or application based, with no interoperability between systems. CBR systems implemented on the web are of the thin client application type, using HTTP forms to submit queries to a CBR server back end. We have looked at 'fatter' thin clients implemented in Java, where the application is divided into GUI logic at the client end and business logic on the server end. The increase in use of these type of Java applications on the web, combined with a data transfer technology like XML suggests the possibility of 'fattening up' client applications. Furthermore, introducing a standard data transfer format allows for system interoperability. This type of interoperability could suggest some breaking down of the division between application systems and agent systems.

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

---

developed a higher version of CBML. However, the current system uses the frame-based language KRELL, and this might in fact be the most suitable representation for these cases.

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 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.

Finally, we have presented two potential applications of our distributed CBR architecture, one of which is suited to event planning on the Internet, Ulysses, and the other which is an Example Based Machine Translation application for intranets.
The research group concludes that CBR is primed to be a pervasive aspect of intelligent systems on the web, particularly in the area of internet commerce and after sales support, where expert assistance can be retrieved with a minimum of query detail. In the intranet domain, CBR systems are suited to industry specific applications, such as the EBMT localisation tool being implemented by the research group. Furthermore, the intranet scenario may prove more suitable for a distributed CBR system since Java security restrictions can be relaxed on client machines, allowing case bases to be saved locally.

8. References

Aamod, A., Plaza, E.; (1994)*Case-Based Reasoning: Foundational issues, Methodological Variations and System Approaches*. AI Communications 7(i) .
Althoff, K.-D, Auriol, E., Barletta R.,Manago, M.; (1995) *A Review of Industrial Case-Based Reasoning Tools*. Oxford: AI Intelligence
Bellman, R.E., (1978) An Introduction to Artificial Intelligence, Can computers think. Boyd and Fraser Publishing company San Francisco
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
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
Caglayan, Alper. Harrison, Colin (1997) Agent Sourcebook. *Wiley Computer Publishing*.
Collins, B. Cunningham, P. (1996) Adaptation Guided Retrieval in EBMT: A Case-Based Approach to Machine Translation in *Lecture Notes in Artificial Intelligence 1168*, Smith, I. Faltings B. (eds.) Springer Verlag.
Cunningham P., Smyth B., Bonzano A., (1998) An Incremental Retrieval Mechanism for Case-Based Electronic Fault Diagnosis, to appear in *Knowledge Based Systems*.
Ginsberg, M.; (1993) Essential Artificial Intelligence. *Morgan Kaufmann Publishers*, San Francisco, California.

INRECA consortium.(1994). Casuel: A Common Case Representation Language, available at http://wwwagr.informatik.uni-kl.de/~bergmann/casuel/CASUEL_toc2.04.fm.html
Knapik, Michael; Johnson, Jay; (1998) Developing Intelligent Agents for Distributed Systems. *McGraw-Hill*.
Kolodner, J.L.; (1993) An Introduction to Case-Based Reasoning. *Morgan Kaufmann Publishers*, San Francisco, California
Kriegsmann M, and Barletta R, (1993) Building A Case-Based Help Desk Application, *IEEE Expert*, 8 18-26.
Microsoft, Frequently Asked Questions About Extensible Mark-up Language (XML). The URL is http://www.microsoft.com/xml/xmlfaq.html
Minsky, M., Seymour P.; (1969) Perceptrons. *Cambridge*. MA: MIT Press
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
Riesbeck C, and Schank R., (1989) Inside Case-based Reasoning. *Lawrence Erlbaum* .
Russell, S. Norvig, P. (1995) Artificial Intelligence, a Modern Approach. *Prenctice Hall.*
Shank, R., Abelson R.; (1977) Scripts, Plans, Goals and Understanding. Hillsdaler, NJ: *Lawrence Erlbaum Associates*
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.
Smyth B. (1998) Case-Based Maintenance, to be published in **???**
Summer Institute of Linguistics web page. XML: Proposed Applications and Industry Initiatives. http://www.sil.org/sgml/xml.html#applications
The Open Trading Protocol Consortium Internet (1998) *Internet Open Trading Protocol Specification, parts 1& 2.* Available for download at http://www.otp.org:8080/
Turing, Alan; (1950) Computing machinery and intelligence. *Mind*
W3C Working Draft 09-Dec 1997. Document Object Model Specification.
URL: http://www.w3.org/TR/WD-DOM-971209/
Watson, Ian.(1997) Applying Cased Based Reasoning: Techniques for Enterprise Systems. *Morgan Kaufman Publishers*.
Wilcox, Steve, (1997) Designing Thin Java Client Applications for Network Computers. *Aviteck, LLC 1997*. http://java.sun.com:81/javareel/isv/Aviteck/ThinClientWP.html