An application of Case Based Reasoning to Object Oriented Database Retrieval.[1]

Jeremy Ellman
MARI COMPUTER SYSTEMS LIMITED
MARI House, Old Town Hall
Gateshead, Tyne & Wear
NE8 1HE
Jeremy.Ellman@mari.co.uk

ABSTRACT

In the near future, a potentially huge number of telecommunication services will be available to the public. Information about these services will be stored by suppliers in large object oriented databases. Users will be able to locate services that suit their requirements by retrieving objects from these databases. However the user's decision concerning the most appropriate service will not necessarily be simple: It will involve value judgements that seek to balance the desired service characteristics with other factors, such as availability, preferred supplier, and cost.

One approach to this problem is to treat potential services as cases. This permits the application of CBR techniques, where the user's service requirement acts as the match target.

The ASCOT project followed this approach. CBR has the key advantage that the user is given a choice of matched services to select among. He may then refine his requirements appropriately. The implementation highlighted the need for "explanation" capabilities that allow the user to understand why one match is preferred over others.

KEYWORDS: Case Based Reasoning, Object Oriented Databases, User Requirements Capture.

Introduction

ASCOT is a project in the CEC's RACE program running from 1992-1995. Its principal goal was specifying and demonstrating a set of software tools for end-user configuration of advanced telecommunications services. The primary purpose of these tools was to maximise service usability and hence promote use of new telecomms services. The toolset specification was influenced one the one hand by work on the architecture of telecommunication services and on the other by human factors considerations. These aim to improve usability by decreasing the number of "enabling tasks", or complex mental actions, (Whitefield et al 1992) that a user has to perform in order to use a service.

Implementation commenced in 1993, and a public demonstration and user evaluation of the toolset were held in 1994. The project is currently disseminating and generalising the toolset results.

ASCOT's key principles

[1] Published in 1st UK Case Based Reasoning Workshop 1995, Watson, Marir, and Perera (eds) University of Salford Research and Graduate College

An early principle that ASCOT decided was that user's service requirements should not be restricted to those offered by telecomms providers. There were two reasons for this: Firstly, it permits users to be flexible: Thus, if they must compromise over what services they accept on one occasion, they retain the flexibility to immediately exploit new services as they become available. The second reason concerns mobility. Since the services offered in any particular country or location will vary depending on its infrastructure, so a user may very well find his service requirements satisfied in different ways as he travels. Consequently the separation of user requirements is a key project element.

Another key element of ASCOT was the simplification of the user interface. We therefore created a graphical tool to simplify the requirements capture process -- particularly so that it was acceptable to non-technical service consumers. These diagrams called *Reference for User Services* (RUS, for details see Byerley and Bruins, 1992) describe the communicating entities and the channels that link them iconically (Beardon et al 1993): That is, the tool represents advanced services using different kinds of graphical links between iconic people and terminal devices. The tool produces user level service descriptions that are approximated into service objects using a simple rule based program.

The service objects obtained from the user may now be matched against those available. This module (known as the Service Unifier) uses Case Based Reasoning (CBR) techniques to offer the user a choice of several possible services. If the user does not find any of the services appropriate, he may refine the RUS diagram he used to specify them until a suitable service is located. Once selected, the service is started, and all the appropriate connections are made. The service may then be configured and controlled as the user uses it. (Figure 1 shows the toolset interface with a RUS diagram, and matched service)

The ASCOT toolset also includes a full set of facilities for the control of services in-call. Whilst these are technically interesting (in that they allow the change of Quality of Service, and the making and breaking of connections), there is no doubt that this would not have been possible without the Service Unifier. We shall now go on to look at its implementation in more detail, highlighting the more CBR specific elements.

The Implementation

The Service Unifier is a RETRIEVAL tool (Aamodt & Plaza 1994), and what it does is quite clear in principle: It creates a case base of possible services to be matched against a user's specification. Implementation was non trivial however due to the size and complexity of the Service Provider's object oriented database (OODB). This database was designed to support telecomm's planning and consequently includes far too much detail for the average service consumer. [#]

[#] In fact, the database links to a model of an ATM telecomms network model that allows change in network bandwidth over particular links to be predicted as user's plan to use services.

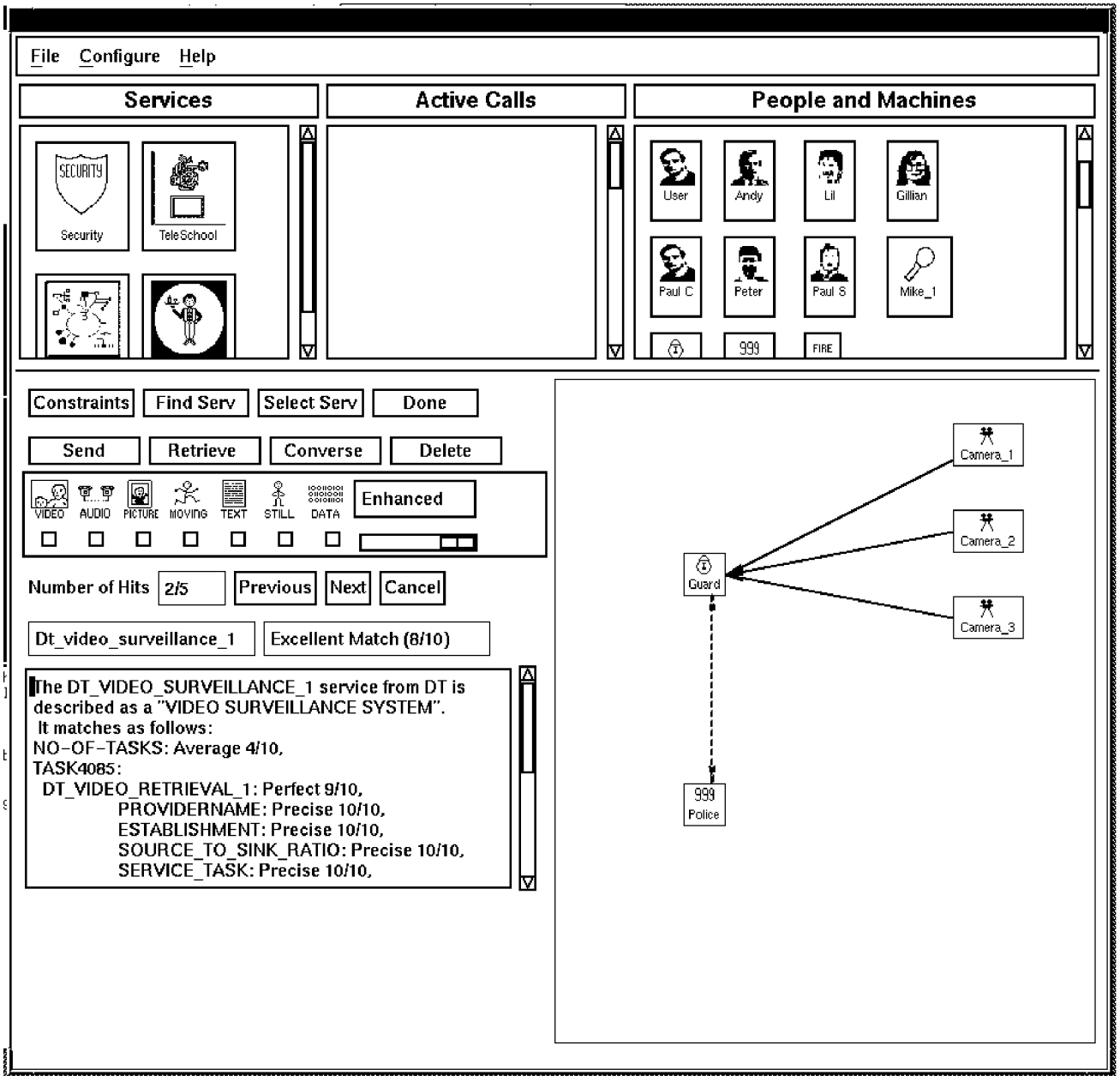

Fig 1. The ASCOT User Interface

Even if when we ignore the complexity of the OODB, future telecomms services are complex objects. Matching a user specification to one of this service involves taking into account factors at several levels of the object hierarchy of differing importance. Since the user requirements may change with each query, creating a static case base is inappropriate. The alternative, dynamically indexing the whole OODB for each query was equally unrealistic on performance grounds. Therefore, a mixed two stage approach was devised. Since this is domain dependent (though the approach will have wider applicability), I shall first discuss the nature of Broadband Service Objects.

Each service object is made up of service tasks, that in turn include service components. These correspond to what is usually understood be a telecomms service. For example an ordinary phone call would be an audio service component. Other service component types are video, data, still graphics, etc. These service components are modified by their quality of service, which for audio could be standard, hi-fi, or CD quality.

A video phone service would be a single service task made up of both an audio and video service component of particular qualities. A complex service may be made up of several service tasks: For example, a video conference where notes are made via a shared white board and an on-line database is available for consultation.

Component types are not the only factor in matching a user's requirements. The availability of the service is a factor, as is cost, and most importantly who provides the service. Indeed the service components in a complex service may be provided by different companies, and packaged together by a value added service provider.

Typically, this means that matching the user's task requirements is as important as matching his overall service preferences. Indeed, the same task may be included in several possible complex services that we must select among. The implication here is that the overall best service match may be a compromise and not include any best task matches. Alternatively the user may greatly prefer a good match on one task in an overall inferior service. In the interests of usability, the user must be made aware of both options and choose himself.

The principal here is that matching tasks is fairly straightforward, but matching services is not. This is because services can contain different number of tasks, and there is often no definitive correct solution to find. Therefore we match the tasks separately, and use these to index candidate service matches. Since there are fewer of these than in the whole OODB we avoid search time growing exponentially with regard to the number of tasks.

The steps in the process are as follows:-

¬ INITIALISE Task Case Base from the OODB
- CAPTURE user's service requirement
® RETRIEVE matching tasks from Task Case Base
¯ LOOK-UP services that contain these tasks from the OODB
° CREATE Case Base of these services
± MODIFY user's service requirement to be a query
² RETRIEVE matching services
³ CONFIRM user selection, or GOTO -

Looking at this in more detail, the first step in matching tasks is to create a case base of available service tasks. Here we face an indexing problem, since service tasks are complex objects. Since the CBR system used (ART-IM) matches strings, words, or number we need to process the service task objects into a suitable form. This takes the service components and their qualities and combines them into complex strings for each task. For example, "STANDARD#AUDIO HIGH#VIDEO ". Additional processing also ensures that service type plays a larger role than quality in matching. Since building the task case base requires considerable processing it is only done once when the system is initialised.

Steps 5, 6 and 7 above are the next relevant ones for CBR. In step 5 we collected the services that best matched users tasks, and also kept numeric scores indicating the quality of these matches. Therefore in matching services, we can reduce the users requirement for task matches to a numeric match. That is, the user would like a service whose tasks matched his specification perfectly (1.0)

Essentially this solution reduces the problem of complex object matching to a two step process: Flatten the complex task structure, and find matches of these against the user's required

tasks; Copy and simplify the services that contain these tasks dynamically into a new, small, case base that can be used to select a best overall service match.

This breakdown was needed since the CBR system used did not deal with complex objects in both source and target cases. It also has the additional advantage of "Strength Reduction", in that services with multiple tasks increase work linearly, not exponentially. Our approach did not however solve problems that we call over-matching, under-matching and multiple task matches.

Multiple task matching occurs when the user specifies several similar tasks. Here one service task may match each user task very well, however it can not match them both simultaneously. The solution here is mechanical, and simply involves preventing the creation of multiple case base entries where a task slot is multiply assigned.

Over matching is a more interesting problem: Since some services may contain more tasks than the user requires, they are clearly less preferable (not least on the grounds of cost). However, if there are no better choices he should be told about this service.

Under matching is a closely related problem. Here one (or more) tasks match the user's specification very well. However the service does not contain enough tasks to satisfy all his requirements. If no other service matches as well, the user would like to be offered this as one of his options. Clearly, some kind of compromise is required.

The solution adapted in the Service Unifier is to explicitly count the number of tasks in a service and to include this in the generated case base. Over matching (and under matching) are consequently penalised, but not excluded. This is a satisfactory approach for users, who accept it if it is appropriately explained.

Explanation

Early on in the system's development both potential users and developers were often puzzled by the services (or cases) retrieved. This was due to an interaction between the complexity of the database and the matching process. Simply, some cases would be preferentially retrieved based on slight numeric differences. It was therefore important to reduce the effect of these since a user would be recommended some case over another due to an artifact of the matching process.

It was therefore decided to implement a minimal "explanation facility". This simply shows the user the quality of a (service and task) match using one of the terms "NO MATCH, VERY POOR, POOR AVERAGE, FAIR, GOOD, VERY GOOD, EXCELLENT, PERFECT & PRECISE.

An essential first step was to understand the calculations carried out by ART-IM. We found that the weight assigned to any one parameter is influenced by the total weights assigned to the match. Consequently we needed to convert this match value into a simple fraction. This then allowed a straightforward index the set of match terms.

This explanation of match quality was a key aspect of the system's success. Users had no trouble differentiating between say good and poor matches, and were not inhibited in selecting other than the "best" match if they could see there was minimal difference between that and the second best.

Discussion

Frequently Case Based Reasoning is seen as a research paradigm (Slade 1991) into the structure of memory (ie following Schank and his group). Alternatively CBR is often used for applications such as building a help desk (Kriegsman and Barletta 1993). In both of these areas implementations follow what Aamodt & Plaza1994 calls a CBR cycle.

This CBR cycle is made up of four steps:-

¬ RETRIEVE the most similar cases
- REUSE the knowledge in those cases to solve the problem
® REVISE the proposed solution
¯ RETAIN the parts of the experience likely to be useful for future work

This cyclic description applies equally to ASCOT. The majority of the work reported here (ie the Service Unifier) has dealt with RETRIEVAL, and how to implement this for a large OODB. Uses also have the services found automatically stored for later use. Indeed the RUS diagrams used can also be REVISED later if a modified service is needed. Work to date does not however RETAIN appropriate elements of the experience. This would in essence require the tuning the parameters used in matching, so that peoples preferences were better reflected.

In this implementation, the match weights were decided in advance, and were not modifiable by the system user. This is a simplification of the true situation. Firstly, the user may wish to prioritise some particular aspect of the service, or may have to respond to external constraints. These may be simple technological issues, such as restrictions in his terminal equipment, or policy issues. For example, his employer may have some preferred supplier relationship with a particular service provider.

In a more complete implementation, the system user should be able to tune these parameters, and should also have the capability of prioritising some particular communications channel.

Whilst the work described here has been fairly domain specific, the problems encountered have often been quite general. Essentially the ASCOT Service Unifier applied CBR to the problem of retrieval from OODBs. The approach chosen (flattening the query structure, and doing retrieval in stages) has the advantage reducing the problem from an exponential one to a series of linear steps. The creation of a smaller intermediate case base was also effective in that the most complex matching was only needed on a subset of the OODB.

Related Work

Aamodt and Plaza 1994 give a considerable number of references to all areas of CBR. However, with regard to database retrieval, Shimazu et al 1993 describe a wholly more ambitious scheme that uses the corporate RDBMS as their case base. This approach allows CBR to be incorporated into enterprise wide systems, and brings with it the advantages of data security, integrity and so on. The essential aspect of their work is the application of SQL to achieve nearest neighbour matching. A major difference is the relatively simple nature of their queries, and the fact that queries and cases are essentially expressed in the same way.

Conclusion

Case Based Reasoning was an essential part of the ASCOT project. It increased the toolset's potential usability by separating user's requirements from the engineering requirements of the telecommunication service providers.

Case Based Reasoning allowed this to be achieved cleanly and naturally. No lengthy on-line user interrogation was needed. Additionally the user's involvement in the process was not compromised. This was because he always has the final choice over which service to select, or could choose to further refine his requirements.

References

Aamodt A and Plaza E 1994 "Case Based Reasoning: Foundational Issues, Methodological Variations and System Approaches" AI Communication Vol 7, 1 March 1994

Beardon P Dormann C Mealing S, and Yazdani M 1993 "Talking with Pictures: Exploring the possibilities of Iconic Communication" AISB Quarterly Issue 83/84

Byerley P., & Bruins, R,. Conceptual Framework for Usage of Telecommunication Services. In: P. Byerley and S. Connell. (Eds).

Byerley P. and Connell. S. (Eds) 1992. *Integrated Broadband Communications: Views from RACE - Usage Aspects* North-Holland Studies in Telecommunication Volume 18. Elsevier

Kriegsman M. Barletta R 1993 "Building a Case Based Help Desk Application" IEEE Expert Vol 8 no 6 December 1993

Shimazu H. Kitano H, and Shibata G "Retrieving Cases from Relational Databases: Another Stride to Corporate Wide Case Based Systems Proc IJCAI 1993

Slade S 1991 "Case Based Reasoning: A Research Paradigm" AI Magazine Vol. 12 no 1. Spring 1991

Whitefield A.D., Byerley, P., Denley, I., Esgate, A., and May, J., 1992, Integration of services for human end-users 1: Design principles, enabling states analysis and a design method. In: P. Byerley and S. Connell (Eds):