The Knowledge Engineering Review, Vol. 9:4, 1994, 383–397

Hybrid case-based reasoning

JOHN HUNT* and ROGER MILES

Bristol Transputer Centre, University of the West of England, Filton Campus, Bristol, BS16 1QY, UK (Email: jjh,rgm@btc.uwe.ac.uk)

Abstract

This paper reviews a number of hybrid Case-Based Reasoning (CBR) systems. These systems are hybrid because the CBR components cooperate with one or more "co-reasoners" which employ a different type of reasoning strategy (e.g. qualitative simulation, constraint satisfaction, etc.). In this paper, we propose that CBR is in fact an inherently hybrid process. We review a number of systems and identify three classes of architecture which have been used for hybrid systems. We believe that to successfully exploit a co-reasoner within a CBR system it is necessary to analyse where, when, why and how the information provided by the co-reasoner will be used. We propose the KADS methodology as a suitable way of performing such an analysis and illustrate its use by example. We conclude by considering the control issues associated with the construction of hybrid CBR systems. We review the requirements of such systems and consider how well the two existing cooperation architectures match those requirements.

1 Introduction

Case-Based Reasoning (CBR) is a technology which is proving extremely successful in a wide range of application areas (Kolodner, 1993). A good introductory definition of CBR is that a case based system is one which "solves new problems by adapting solutions that were used to solve old problems" (Riesbeck and Schank, 1989). This means that in case-based reasoning essentially a store of successful past cases, part cases or solution plans are assumed to exist and the reasoning task is now to *retrieve* past solutions (cases), *adapt* these solutions to the current requirements and to *store* the successful new solutions into the case database.

Although CBR research has been applied to a variety of applications in a diverse range of domains, little attention has been paid to how inferential and problem solving tasks have been used within these systems. An analysis of these tasks highlights the diverse range of inferential processes which actually comprise many CBR systems. In many cases, CBR techniques are combined with additional reasoning techniques to solve a specific problem. Indeed, from analysis of CBR systems it appears that CBR is a technology which is well suited to augmentation by other processes.

In this paper, we briefly describe the CBR process in section 2. In section 3 we consider how CBR systems have been augmented by additional reasoning methods and the implications of such augmentations. In section 4 we present a KADS system analysis of the inferential processes which might be used within a CBR system which is to be augmented by model-based information. In section 5 we consider what requirements hybrid CBR systems impose on the architectures used to control the various elements of the system.

2 Case-based reasoning

CBR has been applied to a variety of applications from device diagnosis and design through interpretation and classification to legal argumentation (Goel, 1992; Hammond, 1986; Hua and

*Current address: Department of Computer Science, University of Wales, Aberystwyth, Dyfed SY23 3BZ, UK. (Email: jjh@aber.ac.uk)

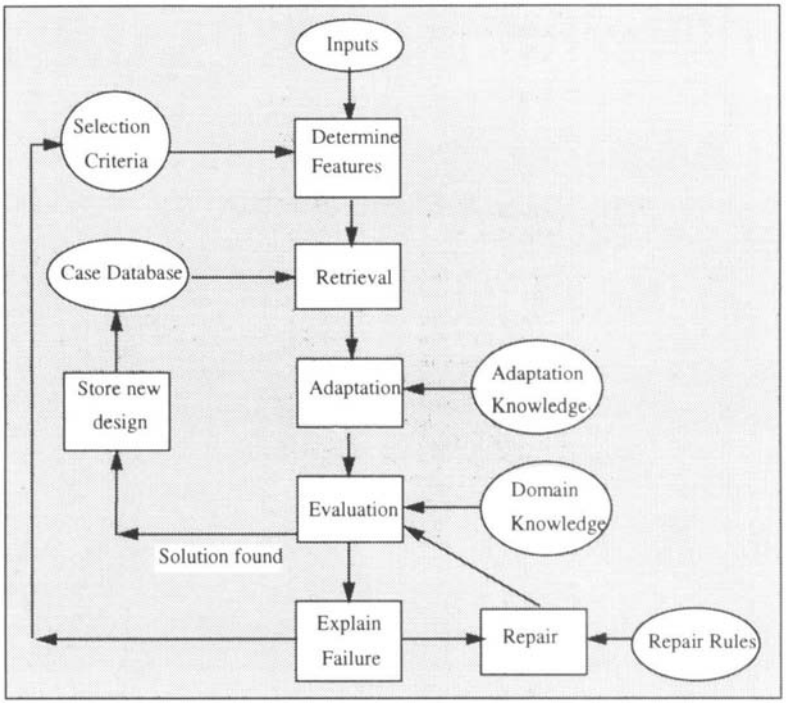

Figure 1 An example CBR system structure (based on Riesbeck and Schank, 1989)

Faltings 1993; Rissland and Skalak, 1991). Figure 1 illustrates the basic structure of many CBR systems. The most important element in a case-based system is the case database itself. This is a repository of past problem solutions, and is the basis for the whole reasoning process. As such, the way in which a case is represented is critically important. For example, a good case representation will allow the important features of the problem to be identified and reasoned about. It will also promote the effective and efficient search of the case database.

Once a case database has been obtained, the first step performed by a CBR system is to analyse the inputs to the system to determine the important features to use in selecting past cases from the case database. These features are then passed onto the retrieval step along with the initial inputs. The retrieval step then uses the information provided to it to obtain a list of past cases which match the current situation. This matching process can be performed in a number of ways including using indices to direct the search, decision trees generated using induction algorithms, nearest neighbour formulas, etc. From this list the best case can be selected.

Once the best match has been retrieved, it is often the case that it must be altered to match the current situation (this is called "adaptation"). A variety of methods have been used to perform adaptation; these can include heuristic rules, interaction with the user, first principles analysis and formulas. The actual method chosen depends upon the application and the domain.

Once the case has been adapted, it must be evaluated to determine whether it does provide a solution to the current problem. Again, a variety of techniques can be used to perform this evaluation. For example, interaction with the user may be employed, or a proposed solution can be simulated or a set of constraints may be tested. Again, the actual method chosen depends on the task and the domain. If the case is accepted by the evaluation step, then it is presented as the solution to the problem and stored in the case database for future use.

If some aspect of the current problem is not solved by the case, then the case must be repaired such that all aspects of the problem are addressed. This is done by first identifying why the case failed to solve the problem (e.g. some constraint on the problem was not met) and then using this information to guide the repair process. The repair process is most often guided by a set of heuristic repair rules. These rules can capture general knowledge about how to "repair" cases in the current domain.

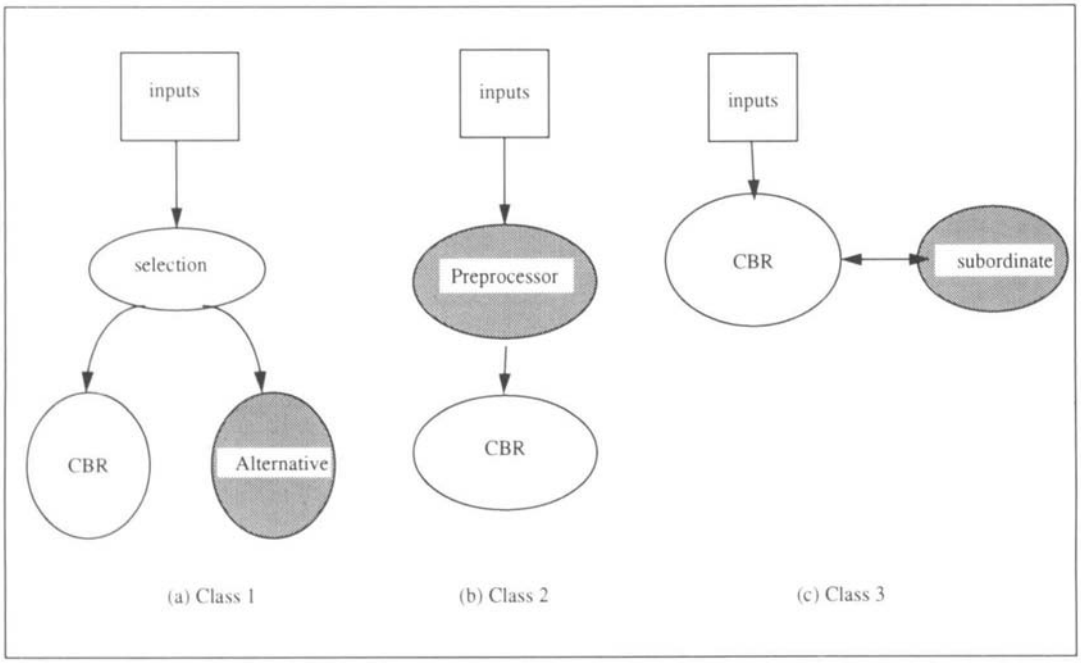

Figure 2 Approaches to cooperation in augmented CBR systems

In some situations it may be possible to abandon the current case and to select a new case from the case database. The explanations for why the previous case was inappropriate can then be crucial in helping to select a new (more suitable) case from the case database.

3 Hybrid CBR systems

CBR has been applied to a wide range of tasks and domains; some of these systems have employed a "pure" CBR approach, that is they have adopted a solution which relies solely on CBR to solve a problem. However, a large number of the CBR systems applied to complex tasks such as design use "hybrid" CBR systems. These are systems which use CBR as their foundation, but which combine some other reasoning methods with CBR. We refer to these reasoning methods/systems as "cooperative reasoners", or "co-reasoners" for short. Examples of co-reasoners which have been used with CBR systems include qualitative simulators, constraint satisfaction systems and rule-based reasoners.

For the purposes of this analysis, we have chosen to define a hybrid CBR system as one which has an identifiable separate reasoning process. Hybrid CBR systems possess an additional reasoning system; these co-reasoners are distinct from the CBR system in that they can often operate in a stand-alone manner, or are physically distinct from the CBR system. It is important to notice the difference that this definition makes between a CBR system which possesses its own set of heuristic rules which guide the adaptation process, and one which cooperates with a separate rule-based reasoning system which is used to adapt a design. In the first instance, the heuristic rules are an integral part of the CBR system; in the second, the heuristic rules in the rule-based reasoner are part of a distinct, separate reasoning system. This has significant implications for control, communication and cooperation, which are not apparent in the integrated system. In many cases it also reflects a great difference in the complexity and power of the heuristic rules.

Figure 2 illustrates some of the approaches used in Table 1 to augment CBR systems with a co-reasoner. For example, fig. 2a illustrates the approach used by the ROUTER system (Goel et al., 1991). In this approach to constructing a hybrid system, some control module selects whether to use the CBR system or the co-reasoner. In the remainder of this paper we refer to this as a class 1 system. In fig. 2b, the co-reasoner is being used as a preprocessor for the CBR system. This

Table 1 Augmented CBR systems

Name	*Application area*	*Type of augmentation*	*Use of augmentation*	*Reference*
CABARET	Legal Argumentation	Rule-based reasoning	Alternative partial solution	(Rissland and Skalak, 1991)
CADET	Design	Qualitative reasoning	Evaluation	(Navinchandra, Sycara and Narasimhan, 1991; Sycara and Navinchandra, 1989)
CADSYN	Design	Constraint satisfaction	Adaptation	(Maher and Zhang, 1991; 1993)
CADRE	Design	Numerical constraint satisfaction	Adaptation	(Faltings, 1991; Hua and Faltings, 1993)
CASEY	Diagnosis	Model-based reasoning	Justification and as fall back	(Koton, 1988)
DEJAVU	Design	Computational design environment	Drive adaptation	(Bardasz and Zeid, 1993)
GREBE	Interpretation	Rule-based reasoning	Alternative partial solution	(Branting and Porter, 1991; Branting, 1991)
JULIA	Design	Constraint satisfaction	Adaptation	(Hinrichs, 1992; Hinrichs and Kolodner, 1991)
KRITIK	Design	Model-based reasoning	Adaptation	(Goel, 1991; 1992)
PANDA	Design	Rule-based reasoning	Specification	(Roderman and Tsatsoulis, 1993)
ROUTER	Planning	(Spatial) Model-based reasoning	Alternative solution	(Goel and Callentine, 1992; Goel et al., 1991)
Wang and Howard	Design	Rule-based reasoning	Repair	(Wang and Howard, 1991)

approach is exemplified by the PANDA system (Roderman and Tsatsoulis, 1993). It is termed a class 2 system below. Finally, fig. 2c illustrates a client-server approach in which the CBR system uses the co-reasoner to augment one of its own reasoning processes. This approach represents the structure of the majority of hybrid CBR systems, and is illustrated by systems such as CADSYN, CADRE and KRITIK, etc. in Table 1. We have chosen to refer to this class of system as a class 3 system.

Such hybrid approaches are often necessary due to the complexity of the task in the chosen domain. This is not surprising as complex tasks often require different reasoning processes for different subtasks. This point is echoed by Kolodner (1993, p. 537), where she says that "in design, case based reasoning may be used to suggest solutions, to access the solution and to suggest the interactions between parts of a solution, but not for maintaining interactions between parts as the solution evolves or breaking the problem into component sub problems". In other words, for certain parts of the design task, CBR is not appropriate and another reasoning process must be used. Indeed, Maher and Zhang (1993) state that "a hybrid model, . . . becomes a common approach in many implementations of case-based designs". The situation is similar in tasks such as diagnosis, planning or repair.

We propose that CBR is an inherently hybrid reasoning process and that the designers of CBR systems should consider that for many (more complex) applications a pure CBR system is inappropriate. This point is highlighted by Bardasz and Zeid (1993), who say that "the integration of case based reasoning with other reasoning mechanisms could yield a hybrid system that surpasses the capabilities of either system alone". Table 1 illustrates a selection of CBR systems which have been augmented by a co-reasoner. We present a brief discussion of how each of the systems have been augmented in section 3.1. The table itself indicates the application area of each system, what the additional reasoning mechanism was and where in the CBR process it was used.

As can be seen from the list in Table 1, the majority of the system developers have augmented the adaptation phase of the CBR process. This could be due to the set of systems we have chosen to

present. However, from analysis of a wide range of systems, and of the CBR process in general, we believe this is actually a reflection of the complexity of the adaptation process. Indeed, if we consider the lack of an adaptation phase in many of the commercially deployed CBR systems (see Magaldi, 1994, for an example of such a system), it is clear that the adaptation process is currently not as well understood as the rest of the CBR process. To a certain extent, this is due to the complexity of the adaptation task itself.

The development of a hybrid CBR system has significant implications for the representation of cases used as the basis of the system. This is because the case representation must meet the representation requirements of the co-reasoner (as well as those of the CBR system). For a system which has been augmented by a rule-based reasoner, this is likely to be less problematic. However, for a system which includes a system such as a constraint satisfaction process, the case representation must be able to represent the constraints relating to the design, plan or interpretation, etc., as well as the design, plan or interpretation itself. This will result in larger and more complex cases, which will require careful analysis during the design of the system.

3.1 How CBR systems have been augmented

Below we briefly describe the systems in Table 1 to identify where and how they use information from their co-reasoners.

3.1.1 CABARET

CABARET is a domain independent reasoning shell which integrates CBR with rule-based reasoning using a set of control heuristics. In Rissland and Skalak (1991), the authors describe how CABARET can be instantiated with cases and rules from an area of income tax law concerning the deduction for expenses relating to an office maintained in one's home. Depending on the control heuristics CABARET can be an example of any class of system illustrated in Fig. 2. In Rissland and Skalak (1991) it possesses two "co-reasoners", one a Hypo-style CBR system and one a rule-based reasoner. Each reasoner is capable of running in a stand-alone manner. A separate controller uses the control heuristics to decide how to use the two reasoners. It does this by selecting tasks which the individual reasoners should perform. In Rissland and Skalak (1991), CABARET has been configured in such a way that the rule-based system is not subordinate to the CBR elements; instead it acts as an alternative to the CBR system. This, of course, introduces control issues and mediation between the results produced by the two systems. These issues are considered in the next section.

3.1.2 CADET

CADET (Navinchandra, Sycara and Narasimhan, 1991; Sycara and Navinchandra, 1989) is a design system specializing in hydro-mechanical systems using a case base of previous designs that realize sub-functions of the desired artifact. It is constructed around a blackboard which is managed by a control module. The control module has access to three types of synthesis methods: rule-, case- and search-based. Potential designs are evaluated by the evaluation module which has access to a qualitative simulation system and a constraint checking system. CADET is currently one of the most complex CBD systems and is extremely powerful. It mixes its reasoning strategies as and when required, and can be considered to operate at different times as both a class 1 and 3 system.

3.1.3 CADSYN

CADSYN is a structural design system for buildings which integrates design cases with generalized domain knowledge (Maher and Zhang, 1991; 1993). In CADSYN a constraint satisfaction co-reasoner is used to transform a design case to match the current situation. Design constraints are then used to verify a new design. The output of this co-reasoner is used as the basis of the repair process. This process is then repeated on the "repaired" design until a consistent design is achieved. Generalized decompositional knowledge is used to suggest fixes for the failed constraints.

CADSYN is an example of a class 3 system as illustrated in fig. 2c. That is, the co-reasoner is subordinate to the CBR system.

3.1.4 CADRE

CADRE is a very powerful building design system which focuses on case adaptation, leaving case selection to a user (see Faltings, 1991; Hua and Faltings, 1993). It is essentially an example of an architecture in which a co-reasoner is subordinate to the CBR system (hence it is a class 3 system). Once a user has selected a previous case it uses a numerical constraint satisfaction system to identify dimensional discrepancies; it then applies a number of heuristic methods to resolve the discrepancies (e.g. dimensionality reduction). If once the dimensional modifications have been made no solution has been found, then topological transformation rules can be used to relax the set of constraints.

3.1.5 CASEY

CASEY is a system aimed at diagnosing heart diseases (Koton, 1988). It does this using information provided by a causal explanation, which is used to guide the adaptation process. This is done by using evidence heuristics to reason about the differences between the current situation and the past case. It then uses repair heuristics to modify the causal explanation retrieved to match the current situation.

CASEY also possesses a model-based reasoning system which is used as a "fall back" system. This means that in situations where CASEY does not possess a past case, the model-based system can be used to generate a result from "first principles". Once the model-based system has generated a solution, the causal explanation for that solution is saved back into the case database for future use. It is therefore an example of a class 1 system as illustrated in fig. 2a.

3.1.6 DEJAVU

DEJAVU uses an object-based design environment to represent and hold the case database (Bardasz and Zeid, 1993). This system holds design plans for constructing mechanical devices. Once a case has been identified, the design environment uses a demand driven approach to adapting and repairing that design plan for the current situation. That is, a value for a particular attribute is only calculated when it is required. Thus when an existing design plan is to be used in a new situation, the system can request that the adaptation part provides appropriate modifications when missing or incorrect information is identified. The actual adaptations are performed using heuristic knowledge represented as knowledge sources in a blackboard system. Thus the design plan system drives the adaptation process. DEJAVU is a class 3 system.

3.1.7 GREBE

GREBE (Branting and Porter, 1991; Branting, 1991) possesses a rule-based co-reasoner, that is, its rule-based system is not subordinate to the CBR elements. Instead, it acts as an alternative to the CBR system; it is therefore a class 1 system.

3.1.8 JULIA

JULIA is a CBR system for planning meals (Hinrichs, 1992; Hinrichs and Kolodner, 1991). It does this by retrieving past cases and then adapting them using an adaptation engine. The adaptation engine uses a constraint propagator to identify and subsequently resolve constraint violations. It also possesses a small set of primitive adaptation transformations. The constraint propagator is a co-reasoner, which places JULIA into the class two system category.

3.1.9 KRITIK

KRITIK is a design system which uses its MBR system to guide its adaptation process (Goel, 1991; 1992). That is, KRITIK analyses the retrieved case to determine if it can provide the required functionality. The functional model used by KRITIK represents the functionality of a system, the

behaviour of the functions (essentially how the functions are achieved/implemented) and the structure elements associated with the behaviours. KRITIK uses this to determine if any part of the case fails to provide the required functionality. If it does, it uses the function's behaviour to identify the structural faults which could have caused the failure. It then uses heuristic knowledge in the form of modification plans to repair the case. KRITIK is an example of a class 3 system.

3.1.10 PANDA

PANDA is a fire engine design system (Roderman and Tsatsoulis, 1993). It is unusual (for this set of systems) in that the rule-based addition acts as a pre-processor to the rest of the system. It is therefore a class 2 system (see fig. 2b), that is, the users describe their requirements informally to PANDA. The rule-based translator then converts these into a formal design specification. This is often used as input to the rest of the CBR system.

3.1.11 ROUTER

This is a system for path planning which uses both a CBR system and a numerical model-based system (Goel and Callentine, 1992; Goel et al., 1991). When Router is presented with a problem it heuristically selects which reasoner to use. It is therefore an example of a class 1 system.

3.1.12 Wang and Howard

In this CBR system (for structural engineering design), the cases represents the previous design (which includes the problem specification, the design plan and history as well as the final design). The design plan itself is rerun to obtain a new design, instantiated with the new problem. The rule-based reasoner is used to provide solutions to subgoals (in the design plan) which cannot be met from the past design. This is an example of a class 3 system. The system is documented in Wang and Howard (1991).

3.2 Implications of augmentation

Within a hybrid CBR system, it is important to consider what co-reasoners will be required and how they will be used. A similar point is discussed by Kolodner (1993, p. 537), where she says that "in building a reasoner, it is important to assess what the component parts are to the reasoning process, to determine which of those parts will be the responsibility of the CBR system, what supports the CBR system needs to get its job done, and which kinds of reasoners will have responsibility for the remainder of (the) subtasks".

Existing systems already use additional reasoning processes for subtasks such as problem interpretation, case matching, case selection, case adaptation, case testing and case repair. It is therefore possible for the CBR system to be augmented at any stage of its reasoning process. However, for any particular task in any particular domain, only certain additions will be useful/required. It is therefore necessary to analyse the task in a particular domain to identify these additions and where to use them.

There are a number of ways in which a task analysis can be performed. For example, for applications which make use of a MBR system to build "real" applications, it is necessary to decide when and how to use the information provided by each MBR system. For both design and diagnosis, at least, there are a number of different subtasks to be performed. For each subtask, different reasoning processes might be suitable at different times and in different ways. One process may also be of use to several different subtasks. Research is still required into how, in general, a high level task such as diagnosis might be broken down into lower level, reusable subtasks. One possible approach to this is discussed in section 4.

Once the co-reasoners have been selected, it is then necessary to consider how the various components will cooperate. Below we describe two cooperation approaches; the tightly coupled approach and the loosely coupled approach:

- *Tightly coupled approach* By tightly coupled we mean systems in which the lines of communication and focus of control are prescribed algorithmically. For example, JULIA's modules are coordinated by a control strategy composed of two nested cycles of control. The *problem reduction* cycle reduces goals to subgoals, schedules them and evaluates whether they are achieved. The *constraint satisfaction* cycle controls the constraint satisfaction process as well as case retrieval and case adaptation.
- *Loosely coupled approach* By loosely coupled we mean systems in which control may be determined dynamically, and within which the various elements of the system are not closely tied together. For example, both DEJAVU and CADET employ a blackboard architecture (Englemore and Morgan, 1988) to integrate diverse problem solving components within their hybrid CBR systems. In DEJAVU, the blackboard is only used for the adaptation process, with the blackboard itself holding the developing solution and the knowledge sources handling the actual case adaptation. CADET, on the other hand, uses the blackboard to integrate a variety of knowledge sources which handle case storage and retrieval, adaptation, elaboration, user interaction and constraint checking.

When comparing the loosely coupled approach with the tightly coupled approach we note that the advantage of the loosely coupled system is flexibility; any of the cooperating processes has the potential to contribute at any time and in any way, thus enabling complex, opportunistic problem solving behaviour. The tightly coupled approach has the advantage of efficiency and the potential for greater interaction.

The majority of hybrid CBR systems use a tightly coupled approach to cooperation. This is shown by considering which systems utilize a blackboard system. Although DEJAVU and CADET do use a blackboard system, all the other systems directly connect their systems together. For example, JULIA directly links the constraint propagator to the adaptation engine (although it should be noted here that JULIA's architecture is rather more complicated than this, and includes a goal scheduler as well as a structure maintenance system built on top of a reason maintenance system). This may be due to the inherent deficiencies of the blackboard architecture for this task (discussed in section 5), or to the very application specific nature of many of the CBR systems. That is, systems such as PANDA, CADRE and CABARET have been constructed to solve a specific problem in a known domain: they have not been constructed as general CBR shells for use in a variety of domains.

4 A hybrid CBR analysis

In this section we present a KADS analysis for a hybrid CBR system (see Tansley and Hayball's book for an excellent introduction on KADS (Tansley and Hayball, 1993)). We believe that such an analysis is necessary to understand the close interaction between the various components of a hybrid CBR system. We note that the analysis we present is not necessarily the one which would have been generated if we had not been intending to use CBR and Model-Based Reasoning (MBR) systems. That is, if we had been intending to use a constraint satisfaction system for case adaptation and repair, the task structure would have been different. For example, we might not be able to use the constraint satisfaction system to identify initial components of interest, etc.

We have chosen the widely used KADS framework as the basis of our analysis. Part of the KADS framework involves the construction of a conceptual model. This is a description of the knowledge which will need to be built into a Knowledge-Based System (KBS), but at a higher level of abstraction than would normally be used when actually building the KBS. Such an analysis can help to identify how the model information available might be used within the available CBR subtasks.

To simplify the analysis involved, KADS defines interpretation models. These are abstract task and inference layers which provide a knowledge engineer with a guide to the construction of a specific type of system:

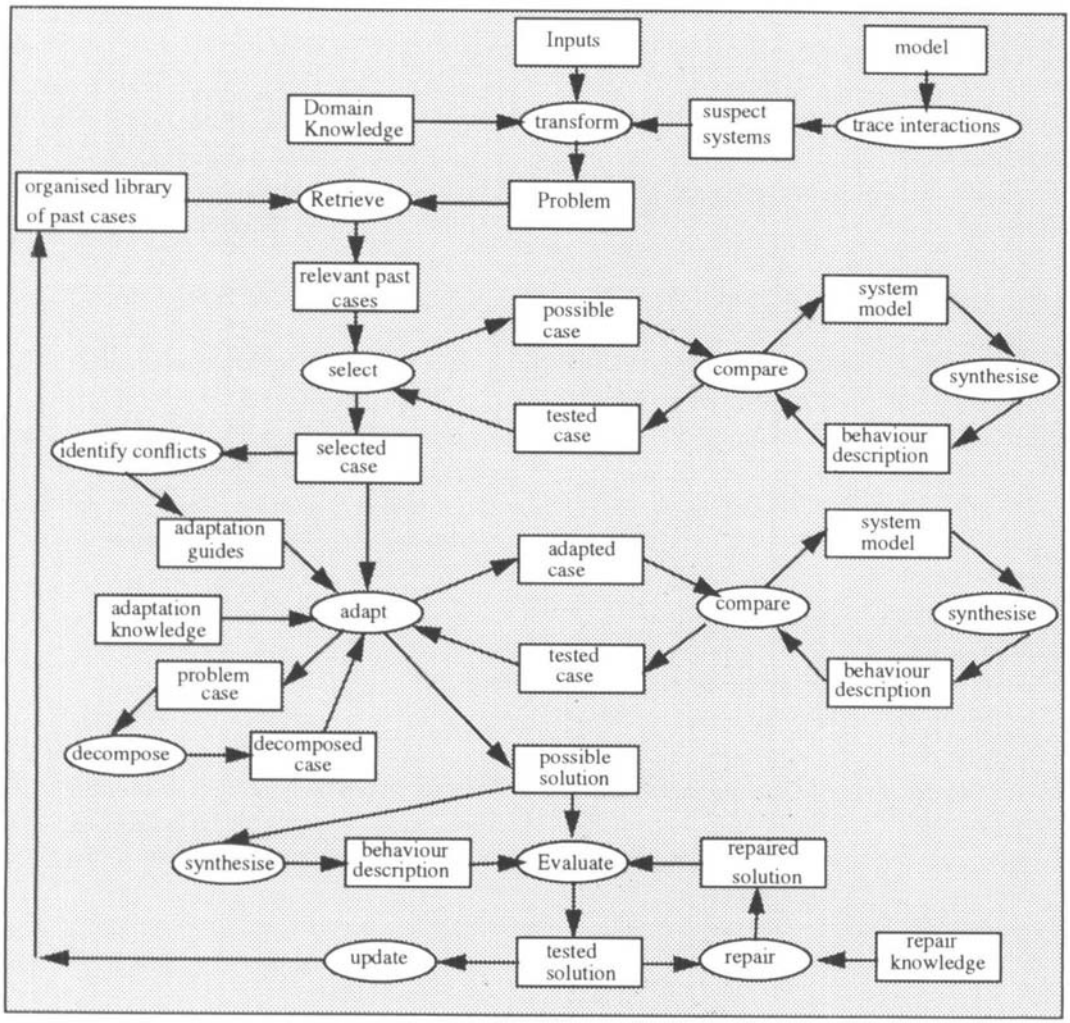

Figure 3 The inference structure of a hybrid CBR system

- *The inference layer* This layer describes the inference structure of the system. This specifies the relationship between the inferential process and the roles played by the data.
- *The task layer* This layer describes the tasks and subtasks which must be performed to achieve the overall goal of the system. The layer illustrates how the inference layer is processed.

We use the inference layer and the task layer to present the results of our analysis.

4.1 Inference layer

The inference structure is a network of inference types and domain roles. In an inference structure, domain roles are illustrated by rectangles. These domain roles are generalized in that an attempt has been made to avoid domain specific names. For example, the initial *inputs* to the system might be a faulty state description, or a failure of some function.

Inference types (represented by ellipses) are descriptions of the way domain concepts, relations or structures can be used to make inferences. That is, an inference type is a transformation of one or more domain roles into one or more domain roles. Note that the inference types are abstractions at the inference layer level (they will only be instantiated when a KBS system is constructed from the framework defined by the inference structure and the task structure). Where possible the names of the inference types are based on those suggested in Tansley and Hayball (1993).

Figure 3 illustrates the inference structure of a hybrid CBR system. It concentrates on the use of model-based information within the CBR system and *adaptation* knowledge. Through analysis of

the figure it is possible to identify where this information can be used. The remainder of this subsection describes the contents of fig. 3.

The first step in the figure illustrates that a set of initial inputs are transformed into a problem statement which is appropriate for the *retrieval* inference type. This is done using heuristic knowledge of the application domain and knowledge about the subsystems which might have been involved in the scenario being developed. This subsystem information is obtained by using a model to identify what functions would have been active given the input state. These functions are then used to identify the systems which implement them.

Next a set of past cases which "match" the current problem are *retrieved* from the case database. This can be done in a number of ways; for example, simple template matching, nearest neighbour or inductive methods can be used. Once the list of relevant past cases has been retrieved, the best of these can be selected. This *select* inference type uses information provided by the *compare* inference type. The compare inference type uses information supplied by the Synthesize inference type. This inference type takes a model containing a potential fault and synthesizes the behaviour of the system containing that fault (at the level of detail represented in the model). The compare inference type performs a comparison between the symptoms displayed by the current problem and the behaviours associated with the retrieved case. The closer the match the better the result. This information is returned to the select inference type along with an explanation of the comparison result.

Once the select inference type has chosen the best match it is passed onto the *adaptation* inference type. The adaptation inference type uses information provided by identifying any conflicts in the model of the system with the original inputs. For example in a design system, the retrieved case may describe a design which matches most of the overall behaviour required, but does not match in one or two details. These details may only be apparent from the behaviour of the model (e.g. see the way in which the KRITIK system uses a functional model to drive the adaptation phase (Goel, 1991; 1992)). It also applies (heuristic) domain specific adaptation knowledge to the case. In the domain of automotive diagnosis this knowledge could contain information such as the ways in which components or connections can fail. For design it might include information on how a design can be adapted for similar tasks.

The adapted case can then be compared against the actual problem situation to see if it meets all the requirements. In fig. 3, this comparison is again performed using a model and a synthesis inference type. If the adaptation inference type fails to adapt the case, then it can be sent to the *decompose* inference type, so that the case can be broken down into its constituent parts. It is then returned to the adaptation inference type where an attempt can be made to adapt the (now) decomposed case. This continues until a possible solution is identified.

Once a possible solution has been found, it is still necessary to *evaluate* the problem to ensure that it meets all the criteria imposed on it by the current problem. In a design application, this may mean ensuring that the adaptation inference type did not break any constraints on the required design. For diagnosis, it may involve obtaining information from confirmatory tests. These tests may result in the need to check that the new symptoms still match the behaviour of the adapted case. Again this can be done using a synthesis inference type.

In some situations, it may be necessary to *repair* the case to bring it back into line with the constraints or tests. It is likely, for diagnosis, that this might include determining whether different numeric values are significant (e.g. whether a voltage of 5 amps versus a voltage of 3 amps is significant or not). For design it might involve consideration of the properties of different substances being used.

Finally, once a successful new diagnosis, design, etc. has been generated, it must be saved back into the case database for future use. This inference type may involve the generalization of indexes, the use of induction methods or the identification of significant case features, and is performed by the synthesis inference type.

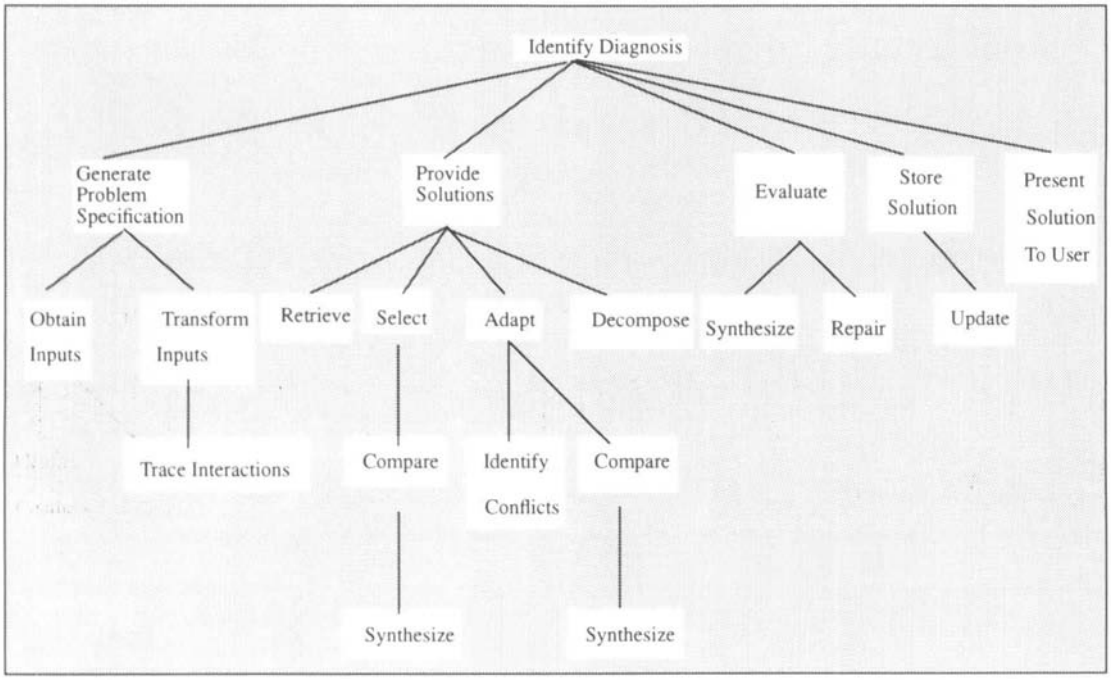

Figure 4 The task (decomposition) structure of a hybrid CBR system

4.2 Task layer

Although the inference structure shows where information from a MBR component may be used, it does not show when that information will be used. To do this we must consider the task structure. Figure 4 illustrates how the overall task of generating a solution to a problem can be broken down into the subtasks presented in the inference layer. The higher level tasks control how and when the lower level tasks are processed. For example, the "Provide Solution" task controls when the "Retrieve", "Select", "Adapt" and "Decompose" tasks are applied.

We have used this task structure as a template for a variety of CBR/MBR hybrid systems. For example, a diagnosis system was defined by instantiating the higher level tasks with diagnostic specific terms. That is, the "Generate Problem Specification" task was termed "Gather Symptoms" and the "Provide Solution" task was named "Generate Suspects", etc.

In KADS, the control of information within a particular task can be specified using pseudo code, meta rules or directed graphs. In whichever representation is chosen, the sequences of subtasks can be conditionalized and repetition can be illustrated. We again use the "Provide Solution" task to illustrate this in fig. 5[1]. This figure illustrates the task structure of the generate suspects task, using the KADS notation. As can be seen from this figure, no information is provided about how the subtasks are implemented. Thus at this level, nothing is said about how the adapt subtask will use the information provided to it.

4.3 Discussion of the analysis

Figure 3 illustrates a number of important points about how information from a model can be utilized within CBR.

Firstly, the basic CBR structure remains the same (i.e. *retrieve* a case, *select* the best case, *adapt* that case, *evaluate* it and *repair* if necessary), even though inferences outside the scope of the CBR system are being used. However almost every step has been augmented by information provided by

[1]We have chosen to use the task structure notation used by Tansley and Hayball (1993). In this notation a pseudo code language is used which supports conditional statements and repetition. It also uses the following conventions; "+" input or input/output, "−" output only.

Provide Solutions (+problem, +library of cases, -possible solution) by

retrieve (+problem, +library of cases, -relevant past cases)

select (+relevant past cases, -selected case)

UNTIL possible solution found DO

adapt (+selected case, +adaptation knowledge,

+suspect components, -possible solution)

IF not found possible solution THEN

decompose (+selected case, -decomposed selected case)

Figure 5 The task (control) structure of Provide Solutions

an inference type which relies on the use of a model. For example, the synthesis inference type uses a system model to generate a description of the behaviour of that system.

This is important, as it is such "model-based" inference types which will be performed by the model-based components (an important point to note is that the whole of these model-based inference steps could be analysed in a similar manner; for example, see Tansley and Hayball, 1993, pp. 317–321 for an inference and task model for qualitative reasoning). Also note that the information provided by these inference types differed depending upon where it is used. This indicates the different ways in which the information from a model will be used. It also implies that different model-based systems, with different functionalities, might be required by the final system. For example, from the inference structure we can see that information provided by the identify conflicts inference type is used to help guide the adaptation phase (in a similar manner to that used in KRITIK). This is the "how the information is used". A task structure could then be used to identify when during the adaptation task this information would be applied.

We note from fig. 3 that it appears that the adaptation phase of the CBR process can most benefit from augmentation by additional reasoning mechanisms. This is highlighted by the concentration of links to the *adapt* inference type. This supports the hypotheses put forward in section 3.

One feature of the task analysis, presented in figs 4 and 5, is that the higher level tasks can be easily instantiated for a variety of specific tasks such as diagnosis or design, while the lower level tasks are more generic CBR subtasks. This illustrates how task specific problem solving systems can be constructed from lower level, more generic, components.

We also note that the lowest level tasks are tasks which could be performed by a MBR system. For example, finding why a case does not exactly match the current problem can be done by finding ways in which the past situation does not match the current situation. This can be done using the models associated with the case and finding points at which they suggest a component or behaviour which does not fit the current problem. This implies that a co-reasoner is being used within the framework of the CBR process, and that it is not surplanting it. Again, this supports the theory put forward in section 3.

5 Towards hybrid CBR architectures

The cooperative use of different kinds of reasoning components is a challenge in itself. For reasons of complexity, maintainability and practicality, such cooperation needs to be flexible and dynamic.

This has led to the development of cooperative architectures. However, for a variety of reasons none of these architectures directly addresses all the issues of a hybrid CBR architecture. The development of such an architecture is likely to need the solution of several architectural issues: configuration of the architecture for particular tasks; provision of efficient communications between components; and facilities to support cooperation. A discussion of the requirements introduced by these issues is presented below. We also consider the possible architectures which could be used to integrate diverse components within a CBR system.

5.1 Requirements

To facilitate the construction of well-engineered hybrid CBR systems, it is necessary to implement an architecture which encourages the system builder to explicitly distinguish between the CBR processes and the additional components used by the system. As such, the requirements for providing a stable hybrid CBR system are exactly the same as for any other hybrid system of similar complexity. That is, it requires:

- explicit channels of interaction (otherwise *ad hoc* interactions will develop in which "grey areas" exist between the various parts of the system);
- standard translators between subsystem languages (i.e. which allow the terms in one subsystem to be converted into the terms used in another subsystem);
- flexible integration of the components. It is not likely that it will be possible to identify all possible combinations of interactions *a priori* (indeed, such a constraint is undesirable). We therefore need an architecture which will allow opportunistic cooperation between the components as and when necessary;
- meta-level control over tasks. That is, any task knowledge available should be used to direct the overall problem solving process, while at the lower level, the components should be able to interact in an opportunistic manner.

5.2 Possible architectures

Given the requirements presented above, we consider two cooperative architectures which could be used to construct hybrid CBR systems. These two systems have been chosen because the blackboard approach is (probably) the most common, and the task specific approach represents a contrasting approach to the blackboard system. We describe the strengths of each approach and the compromises they introduce. The designers of a hybrid CBR system must therefore consider the options available to them and select appropriately.

5.2.1 Task specific architectures

A task specific architecture provides structures for designing and building large grained tasks such as diagnosis or design. That is, the architecture is specific to a task such as diagnosis, and can therefore provide facilities which will explicitly support that task. The TIPS (Task Integrated Problem Solver) architecture is a framework within which to build task specific systems (Punch 1989; Punch and Chandrasekaran, 1993).

The TIPS architecture possesses a hierarchical control structure in which a top level selector determines which problem solver to invoke dependent upon information supplied by sponsors (one for each problem solver) and knowledge about the state of the problem. It provides a task specific control strategy for exploiting knowledge of how to solve problems for the current application. It also provides mechanisms for determining the appropriateness and subsequently selecting problem solvers based on the task contained in the task specific control strategy.

However, TIPS provides no support for communications between the problem solving components. Within most applications the problem is essentially engineered out. However, the lack of a uniform representation can still result in conflicting information in the different problem solving

tasks. In fact, the TIPS architecture falls back on a blackboard-like construct to provide the mechanism for communications. TIPS therefore concentrates on control and cooperation, but pays less attention to communication.

5.2.2 Blackboard architectures

In a blackboard system (Englemore and Morgan, 1988), a set of problem solving modules (typically called "knowledge sources") share a common global database (called the blackboard). The contents of the blackboard denote, and indeed are often called, hypotheses, and are often structured hierarchically. Knowledge sources respond to changes on the blackboard, and interrogate and subsequently directly modify the blackboard, by creating, modifying and solving hypotheses (in fact they are the only elements allowed to make changes to the blackboard). It is therefore through the blackboard that the knowledge sources communicate and cooperate.

A blackboard architecture meets many of the requirements outlined above. However, it does not allow the explicit representation of the task knowledge, nor does it provide appropriate communication mechanisms (the blackboard acts as the communication mechanism). In fact, the language of the blackboard becomes the communication language as well; this may or may not be appropriate. Some of these problems can be overcome through careful design of the knowledge sources. For example, the CABARET system possesses a control module which effectively encapsulates the available task knowledge.

6 Conclusions

From the analysis presented in this paper we believe that CBR is a promising technology for building cooperative reasoning systems. Not only is CBR a powerful reasoning paradigm, it also provides a natural framework for integrating multiple reasoning systems. However, the construction of such systems is not a trivial task. It requires a detailed analysis of where, when, why and how the potential co-reasoners might work with the CBR components. We believe that the KADS methodology provides an ideal way of obtaining this information. Once such an analysis has been performed, it is possible to construct the hybrid systems. We believe that these systems are (and will continue to be) far more than the sum of their parts.

Acknowledgements

We would like to thank Dr David Pugh of the University of Wales, Aberystwyth for useful comments on the KADS analysis presented in this paper. We would also like to thank an anonymous reviewer for useful comments on sections 4 and 5.

References

Bardasz, T and Zeid I, 1993, "DEJAVU: Case-based reasoning for mechanical design". *AI EDAM* **72**(2) 111–124.

Branting, LK, 1991, "Building explanations from rules and structured cases." *IJMMS* **34** 797–837.

Branting, LK and Porter, BW, 1991, "Rules and precedents as complementary warrents." In: *Proc. AAAI-91*. AAAI Press/MIT Press.

Englemore, R and Morgan T, 1988, *Blackboard Systems*. Addison-Wesley.

Faltings, B, 1991, "Case-based representation of architectural design knowledge." In: *Computational Intelligence 3* (N. Cercone, ed.). North Holland, pp 184–232.

Goel, A, 1991, "A model-based approach to case adaptation." In: *Proc. 13th Annual Conf. of the Cognitive Science Society*. Erlbaum.

Goel, A, 1992, "Representation of design functions in experience-based design." In: *Intelligent Computer Aided Design* (D. Brown, M. Waldron and H. Yoshikawa, eds). North-Holland.

Goel, A and Callentine, T, 1992, "An experience-based approach to navigational path planning." In: *Proc. IEEE/RSJ Int. Conf. on Robotics and Systems*. IEEE Press.

Goel, A, Callentine, T, Shanker, M and Chandrasekaran, B, 1991, "Representation and organization of topographic models of physical spaces for route planning." In: *Proc. IEEE Conference on Artificial Intelligence Applications*, IEEE Press.

Hammond, K, 1986, CHEF: A model of case-based planning." In: *Proc. of AAAI-86*. AAAI Press/MIT Press.

Hinrichs, TR, 1992, *Problem solving in open worlds: A case study in design*. Erlbaum.

Hinrichs, TR and Kolodner, J, 1991, "The roles of adaptation in case-based design." In: *Proc. AAAI-91*. AAAI Press/MIT Press.

Hua, K and Faltings, B, 1993, "Exploring case-based building design—CADRE." *AI EDAM* 7(2) 135–143.

Kolodner, J, 1993, *Case-Based Reasoning*. Morgan Kaufmann.

Koton, P, 1988, "Reasoning about evidence in causal explanation." In: *Proc. AAAI-88*.

Magaldi, R, 1994, "CBR for troubleshooting aircraft on the flightline." *IEE Colloquium on Case Based Reasoning: Prospects for applications* pp 6/1–6/9.

Maher, ML and Zhang, DM, 1991, "CADSYN: Using case and decomposition knowledge for design synthesis." In: *AI in Design 1991* (JS Gero ed.). Butterworth-Heinemann.

Maher, ML and Zhang, DM, 1993, "CADSYN: A case-based design process model." *AI EDAM* 7(2) 97–110.

Navinchandra, D, Sycara, KP and Narasimhan, S, 1991, "A transformational approach to case-based synthesis." *AI EDAM* 5(1) 31–45.

Punch, WF, 1989, *A Diagnosis System Using a Task Integrated Problem Solving Architectures (TIPS), Including Causal Reasoning* PhD Thesis, The Ohio State University.

Punch, WF and Chandrasekaran, B, 1993, "An investigation of the roles of problem-solving methods in diagnosis." In: *Second Generation Expert Systems* (J-M David, J-P Krivine and R Simmons eds). Springer-Verlag, pp. 673–698.

Riesbeck, CK and Schank, RC, 1989, *Inside Case-Based Reasoning*. Erlbaum.

Rissland, EL and Skalak, DB, 1991, "CABARET: rule interpretation in a hybrid architecture." *IJMMS* **34** 839–887.

Roderman, S and Tsatsoulis, C, 1993, "PANDA: A case-based system to aid novice designers." *AI EDAM* 7(2) 125–134.

Sycara, K and Navinchandra, D, 1989, "Integrating case-based reasoning and qualitative reasoning in engineering design." In: *AI in Engineering Design* (J Gero, ed.).

Tansley, DSW and Hayball, CC, 1993, *Knowledge-Based Systems Analysis and Design: A KADS Developer's Handbook*. Prentice Hall.

Wang, J and Howard, HC, 1991, "A design-dependent approach to integrating structural design." In: *Artificial Intelligence in Design '91* (JS Gero, ed.). Butterworth-Heinemann, pp. 151–170.