"USING CASE-BASED REASONING FOR SUPPORTING DECISION PROCESSES"

by

S. DUTTA*

94/59/TM

* Associate Professor of Information Systems at INSEAD, Boulevard de Constance, 77305 Fontainebleau Cedex, France.

Printed at INSEAD, Fontainebleau, France

USING CASE-BASED REASONING FOR SUPPORTING DECISION PROCESSES*

Soumitra Dutta

INSEAD
Fontainebleau
France 77305

Tel: 33-1-60724000
Fax: 33-1-60724049
Internet: dutta@insead.fr

ABSTRACT

Conventional case-based reasoning systems act as "automated problem solvers". The use of case-based reasoning systems (CBRSs) for decision aiding has been proposed in the literature recently [2,21]. Most existing decision aiding CBRSs are designed as smart retrieval systems. They do not focus on monitoring and supporting the decision processes of decision makers. A process focus in decision aiding CBRSs calls for major changes in the design and implementation of such systems as compared to conventional CBRSs. This paper emphasizes the utility of tracking and exploiting decision processes in decision aiding CBRSs and outlines important issues in the implementation of such systems via CAL, a particular process focused and decision aiding CBRS.

Keywords: Case-based reasoning; case-based decision support systems; decision making; decision processes.

USING CASE-BASED REASONING FOR SUPPORTING DECISION PROCESSES

1 INTRODUCTION

Case based reasoning (CBR) [20,26,30] has gained in popularity over the last decade because it [21, p. 53] "provides both a methodology for building systems and a cognitive model of people (and) it is consistent with much that psychologists have observed in the natural problem solving that people do". Several case-based reasoning systems (CBRSs) have described in the literature [20,26,30] and most typically behave like automated problem solvers - producing "solutions" to a problem by retrieving and adapting a prior solution.

However, while people have been seen to frequently use case-based reasoning, they suffer [21] from an inability to consistently recall the appropriate set of prior cases, distinguish between important and unimportant features, recall prior experiences under time pressures, and deal with incomplete and uncertain information in the current problem. These limitations are especially acute for less experienced people who lack a sufficiently complete (and large) collection of prior experiences (cases). Hence the idea of using case-based reasoning for decision aiding has been proposed recently in the literature [2,3,21].

Using CBRSs for decision aiding has a certain intuitive appeal because man-machine collaboration in decision making can be mutually beneficial. A CBRS can help overcome some of the limitations of case-based reasoning by humans. For example, it can augment a decision maker's (DM's) memory by providing access to a large collection of cases and rapidly recalling the most relevant cases, and aid the decision process through appropriate critiquing. In return, the DM can assume greater responsibility for the use of cases, i.e., the adaptation of prior cases to the current decision. While humans have been observed to be fairly good at adapting cases, these adaptation procedures have proven to be complex, difficult to generalize and the Achilles heel for many CBRSs.

Few CBRSs for decision aiding have been described in the literature [3,7,8,13,16,21]. While these systems are described further in the following

sections, a characteristic common to most of them is that they behave largely like smart retrieval systems. Given a particular decision problem, they retrieve partially matching prior cases and present them to the DM. They do not focus on monitoring the decision process of the DM and judiciously intervening in the decision process to aid decision making. A process focus is important because research in decision support systems (DSSs) has shown that computer-based decision aids should be viewed as interventions [28] in the decision processes of DMs. The success or failure of computer-based decision aiding systems depends to a large degree upon the ability of the system to make the right interventions - from both timing and content perspectives - in the decision processes of the DM.

This research emphasizes a process focus in decision aiding CBRSs. Such a focus has important implications for the design of such systems because it changes both the content of cases and the use of cases to aid decision making.

Cases in conventional CBRSs typically emphasize the initial problem description and the final solution. They do not contain the decision process used to obtain the solution. Note that the decision process used to obtain the solution is distinct from the steps necessary to execute the solution. For example, a marketing manager may determine that the best response to a particular problem is to carry out a concerted sales promotion program. This solution (the decision to carry out a sales promotion program) is distinct from the decision process which the manager engaged in to come to this conclusion. A process focus in decision aiding CBRSs implies that cases in such systems need to also represent the decision process used to arrive at the solution.

If cases contain information about decision processes, then this further calls for changes in the way that cases are retrieved from the case library. Cases are retrieved in conventional CBRSs by matching the current problem or the desired solution to the current problem with problem/solution descriptions in prior cases. However, such a retrieval mechanism ignores the decision process used to obtain the solution and may render the case ineffective for decision support, i.e., for making appropriate interventions in the decision process. Thus, the case retrieval procedures in decision aiding CBRSs have

to be modified to reflect the process knowledge contained within cases and for making the right interventions in the DM's decision process.

An emphasis on decision processes also requires a change in the manner in which cases are used. Conventional CBRSs focus on adapting prior cases to provide correct solutions to the DM. In contrast, decision aiding CBRSs aim to aid the DM in the process of arriving at a solution. As noted in the literature [23] rules, algorithms and computational logic often play a limited role in human \ decision processes. More important is the ability to recognize "patterns" - harmonious and stable configurations of different problem components - till a stable pattern or "cognitive equilibrium" is reached. Thus, the information presented to the DM and suggestions provided to him/er have also to be adapted to the process focus of cases and decision support.

To summarize, this research argues for an explicit process focus in CBRSs for decision aiding. Such an emphasis is not witnessed in current decision aiding CBRSs. However, the incorporation of a process focus in decision aiding CBRSs calls for major changes in the way these systems are designed and used. These ideas are described in this research. The implementation of these ideas in a decision aiding CBRS, CAL is also described.

The structure of this paper is as follows. There are four additional sections. While Section two motivates the importance of a process focus within cases, Section three outlines the implications of the shift from "automated problem-solving" conventional CBRSs to decision aiding CBRSs. The next section describes the implementation of a decision aiding CBRS, CAL. Finally, Section five concludes the paper by highlighting the contributions and limitations of this research and providing comparisons with prior research.

2 Decision Processes as Cases

This section motivates the importance of focusing on processes within cases in CBRSs.

In a process oriented view of decision making, the emphasis is on the process by which decisions are made, and not solely on the final outcome.

The relevant question in this context is: how is the decision reached? This is in contrast to the central question in the more traditional outcome oriented view of decision making: what is the decision? The decision process used to arrive at a decision is usually not a simple and straightforward sequence of steps. More commonly it is a complex, iterative process of learning and experimentation by the DM till s/he reaches a state of cognitive equilibrium [33]. The reasons for each step of the decision process are sometimes rational, at other times are based on instinct and perhaps can even be irrational. Cumulatively, the individual steps of the decision process provide a rich insight into the DM's problem solving behavior than the final solution taken alone.

A process focus in decision making is useful for both structured and unstructured problems when there is uncertainty in the decision environment, the DMs are skilled and the organizational decision environment is flexible. With increased information about the decision process, skilled DMs can flexibly and meaningfully change critical parts of decision processes to respond to changes in the external decision environment.

The process model of decision making cited most often in the literature is Simon's [29] four phase model of decision making: intelligence (finding occasions for making a decision), design (determining possible courses of action), choice (choosing among the determined set of courses of action), and review (evaluating past choices). Despite its popularity, Simon's decision model has several limitations. For example, Gorry and Scott-Morton [14] have pointed to the fact that problem solving really consists of a series of inter-dependent, temporally separated decisions, and there is the important phenomenon of learning as a DM solves a problem over time. These temporal inter-dependencies between related decisions play an important role in supporting decision processes and are not captured adequately in Simon's model. Thus, this research takes a more broader view of a decision process as a sequence of inter-related decisions and actions which cumulatively lead to the final solution.

In a process view of decision making, computer-based decision aids can be seen [28] as interventions in the decision process which impact the decision procedures of DMs. A DSS can capture information about the different

aspects of the decision process, such as "what opportunities or problems triggered the decision process?" and "how different solution alternatives were generated and explored", and use this information to support and aid decision making. For example, knowledge about how the decision space was navigated along with reasons, if any for the particular path followed can informate DMs and provide them with insights into their own decision procedures or those of others. Information about the progressive evolution towards the solution together with comments and notes about critical steps in the process can significantly enhance decision support, specially for less experienced DMs.

A collection of such descriptions of decision processes can serve as a valuable base of knowledge to aid decision making by recognizing the strengths and limitations of prior decision processes. These insights can aid DMs to notice special features in the decision environment, explore different solution designs, test alternative hypotheses, and reflect on the obtained results. This "vertical" decision aiding and stimulation from an "horizontal" informating base of a collection of cases of prior decision processes is depicted graphically in Figure 1.

Figure 1 about here

3. THE SHIFT TOWARDS DECISION AIDING CBRSs

This section outlines the move from conventional towards decision aiding CBRSs.

3.1 Conventional CBRSs

Conceptually, the core of conventional CBRSs consists of matching the current problem to a store of cases to retrieve the most relevant similar case and then adapting the retrieved case to the current problem. This is depicted in Figure 2. Many CBRSs also store the final solution back in the case memory (to avoid duplication of effort in the future) as indicated by the dotted line in Figure 2. The DM plays a fairly passive role in either accepting or rejecting the solution produced by the CBRS.

Figure 2 about here

The objective of conventional CBRSs is similar to other AI systems: to automate problem solution. Implicit in this tendency to automate is an emphasis on the decision outcome over the decision process, on providing solutions rather than facilitating problem solution and on the system over the DM. Conventional CBRSs also tend to be fairly restrictive as their adaptation skills arise from possessing structured knowledge of a limited domain.

Most CBRSs described in the literature are conventional in nature. They typically aim to automate the generation of a solution from a prior case. For example, CHEF [15] uses a case library of prior recipes to produce a feasible recipe for a certain problem description and MEDIATOR [19] uses prior cases to suggest a solution for disputed resources.

3.2 Decision Aiding CBRSs

Decision aiding CBRSs aim to support a DM in taking a decision. Rather than automating all steps of the case-based reasoning cycle (Figure 2) as is common in conventional CBRSs, Kolodner [21, p. 65] has suggested that it may be mutually beneficial to design CBRSs for aiding decision making: "because people are good at using cases but not as good at recalling the right ones, useful systems could be built that augment human memory by providing people with cases that might help them to reason but allowing all the complex reasoning and decision making to be done by the person".

Note that the move from conventional automation to decision aiding represents an important shift in the design and use of CBRSs. CBRSs are no longer required to generate a solution, but are viewed as facilitating the process of decision making or as making "interventions" [28] in the decision process. This has major implications for the design of CBRSs as mentioned earlier in Section one and explained in more detail with CAL in the following section.

Kolodner [21] has classified the support offered by decision aiding CBRSs as either "active" or "passive". Active support is defined as the ability to retrieve cases, warn of potential problems, and help in critiquing, with the DM being responsible for adaptation procedures, choice of features and cases to consider, and the evaluation of suggestions and warnings. Passive

support is defined to be similar to a smart database interface with the ability to retrieve cases based on partial matches and answer specific questions on the retrieved case. Given the complexity of the adaptation process in conventional CBR, it has been suggested [21] that it is best to let the DM take the lead in adapting prior cases and make the final decision regarding the current problem.

A small number of CBRSs can be described as decision aiding. Most such systems act as smart retrieval systems and do not focus on capturing and exploiting decision processes. Typically, they aim to provide an interactive interface to the DM via which the DM can control some aspects of the case retrieval or case adaptation procedures.

For example, DMs using Battle Planner [13] are required to describe a battle situation and a solution plan to the system. The system uses the battle situation description and the proposed solution to retrieve similar cases. These cases along with case summaries are then used by the DM to critique his/er own solution. This leads to the DM proposing a modified solution for the battle description which is again used by the system to retrieve a new set of similar cases. This iterative procedure is carried out till the DM does not find the retrieved cases to be of additional value.

What is ignored and/or lacking in Battle Planner and similar systems is knowledge about the *process* by which the solution to the problem was reached. Presenting an analysis of *why* a solution works for a particular problem is very *different* from understanding *how* the solution was reached. For most real world problems, the process of arriving at solutions is a non-trivial procedure consisting of several temporally separated, but inter-dependent decisions. Knowledge about the process of arriving at a decision can often be richer and more useful than the final solution itself as it encodes the creative thinking and experimentation that went into the determination of the solution.

However, as the intent of Battle Planner and similar CBRSs is to aid decision making and not automate decision making, they can be classified as decision aiding CBRSs. Section 5.1 provides more details on other decision aiding CBRSs from the literature.

4 CAL: CASE-BASED ANALYSIS AND LEARNING

A CBRS, CAL has been implemented as an example of a decision aiding CBRS. The context for the development of CAL is Brandframe [6,9], a flexible knowledge-based DSS for supporting a brand manager. While Brandframe has been implemented in a large Dutch company in the area of fast moving consumer goods, CAL is a CBRSs for potential incorporation into Brandframe (see Section 5.2).

The following sub-sections describe the design and implementation of CAL. Note that decision aiding is rich concept which can be manifested in a DSS in a number of different ways. Thus CAL does not represent a unique way to build a decision aiding CBRS. Rather it presents a particular implementation of a process-focused decision aiding CBRS and in doing so it raises some questions about how the design of such CBRSs differs from those of conventional CBRSs.

4.1 Overview of Structure

CAL consists of four major components as depicted in Figure 3:

- **Case memory**: this acts as the repository of experience and stores decision making processes as cases;
- **Monitor**: this module is responsible for both storing selected aspects of decision processes as cases in the case memory and for monitoring the progress of the current decision process to determine appropriate decision-intervention and reflection-intervention conditions;
- **Analyzer**: this component analyzes requests for advise from either the DM or the monitor module and retrieves appropriate cases from the case memory; and
- **Adviser**: this module is responsible for advising the DM regarding the current decision process and problem solution. The adviser aims to both aid decision-making and to stimulate learning about the decision situation.

Figure 3 about here

It is interesting to compare the structure of CAL (Figure 3) with that of a conventional CBRS (Figure 2). An immediate observation is that CAL does not follow the "linear" solution methodology of conventional CBRSs: match - retrieve - adapt. CAL's solution methodology represents a more iterative cycle of: monitor - analyze - advise. CAL does not emphasize the generation of solutions as much as intervening (advising) in the decision process at the appropriate moments to support the DM. The DM is also given a more central interactive role in CAL. In a conventional CBRS the DM only appears at the very end to receive the solution generated by the CBRS.

Further details on the four individual components of CAL are provided in the following sections. It is useful to note that as shown in Figure 3, CAL represents the CBR component of an associated DSS which is Brandframe in this research. The following description focuses only on CAL.

4.2 Case Memory

There are two important considerations in the design of the case memory: first, to decide the nature of information contained in a case and second, to choose an appropriate indexing mechanism for organizing and retrieving cases.

4.2.1 Case Content

The information content of cases is an important issue because it significantly influences the degree of usefulness of the case for CBR. Kolodner [20] has identified the major components of a generic case as: the initial problem/situation description, the solution to the problem specified in the problem description and the outcome, i.e., the resulting state of the world after the stated solution was carried out.

In addition to initial problem and final solution descriptions, CAL emphasizes the decision process used to obtain the solution. Beside the actual steps in the decision process, CAL also stores justifications and reflective observations on the steps of the decision process. A decision making process in CAL is described by a directed graph in which nodes represent descriptions of the decision process at particular instants of time and arcs represent transitions from one decision state to the next. Transitions

between decision states can be caused either by events in the external world (not caused by the DM) or by specific actions undertaken by the DM with regard to the decision situation. The decision to distinguish between events and actions was confirmed after conducting experiments with graduate business students. In the experiments the students were asked to enter (from their memory) the process by which they got their previous jobs. The experiments revealed that the students made a strong distinction between things which happened due to external causes (events) and things which they did themselves (actions). An interesting aspect revealed by an analysis of the entered cases in CAL was that actions were the most used constructs, i.e., the best "remembered" aspects of prior decision processes were those performed by the decision-makers themselves.

Figure 4 about here

Figure 4 partially depicts a simple case depicting the decision sequence taken by a brand manager. States, events and actions in cases are displayed in their temporal order of occurrence (see Figure 4) and are color coded for ease of comprehension. Each case has an associated summary and highlights descriptions which can be accessed by the manager to obtain a quick description of the main features of the case. Each state, event and action is an object and contains both descriptive and reflective attributes. While descriptive attributes describe facts associated with the object, reflective attributes contain reflective observations stored by the DM at particular stages of the decision process. Some attributes are common across all states, events and actions. For example, every object has an attribute "Goals" which lists the specific goals (if any) associated with the object (see Figure 4). Specific attributes can also be added to particular states, events and actions in a case.

To conclude, major differences in the case descriptions of CAL as compared to conventional CBRSs are the emphasis on representing the decision processes of the DM and the inclusion of reflective observations made by the DM. Some CBRSs such as ARIES [4], JULIANA [27] and JULIA [17] record the solution process, but this refers to the steps used by an automated CBR inference engine to derive the solution. In contrast, the decision process stored in CAL refers to the steps in the reasoning processes of a DM. While somewhat similar conceptually, they are different

in structure and purpose. For example, the solution steps recorded in ARIES can be used to derive a solution for a new problem by reinstantiating and re-executing the stored solution with parameters and data from the current problem. In contrast, the decision process of the brand manager will reflect the frustrations of failed attempts and creative thinking which went into obtaining the solution to a problem. This decision process from a prior case cannot be directly used to derive the solution for a new problem.

4.2.2 Case Indexing

Indexing is an unresolved issue for CBRSs [18,24]. The challenge is to devise index structures which can help to efficiently and effectively retrieve cases needed for CBR. The complexity of indexing in CBR revolves around issues such as the exact vocabulary of features to be used as indices, the method used to choose the index labels and the construction of an appropriate indexing structure to facilitate retrieval.

While some systems use low-level or surface features which are easy to extract, others opt to obtain complex, thematic indices after deep inferencing from the decision situation. The relative advantages and disadvantages of each approach have been debated extensively in the literature [5,32] and there is no simple argument to favor one over the other. This debate has taken a more pragmatic turn of late with the growing realization that regardless of the type of indices used, they have to satisfy two important criteria: (a) they have to be functionally useful, i.e., they have to be appropriately suited to the task facing the CBRS and (b) rather than being perfect, they should aim to work most of the time [20]. CAL uses both surface level features and functionally relevant index labels.

There are two generic approaches to choosing index labels [20]: manually and by automated techniques. Manual approaches are recommended when the cases are complex and/or the knowledge required for understanding the case for indexing is not well structured. Automated techniques are more practical when it is possible to form a checklist of index features or when the knowledge for understanding and explaining the case events is well structured and complete. CAL incorporates both manual and automated checklist techniques for determining functionally relevant features and surface labels respectively.

An appropriate organizational structure for the case library is important to the degree that the case library is large and efficiency of case retrieval is required. Most CBRSs do not use a flat organization of cases for efficiency reasons; rather they typically use hierarchical structures such as shared feature networks or discrimination networks [20]. CAL utilizes multiple shared feature networks, i.e., it organizes its indexing labels in several shared feature network hierarchies and indexes cases in multiple hierarchies.

Following Owen's [24] classification of index labels, CAL utilizes both syntactic and semantic indices. Syntactic indices discriminate among cases on the basis of their surface level features. Surface level features in CAL refer to pre-defined categories of states, events and actions. For example, some actions belong to the category of sales promotion actions while other belong to the category of advertising actions. Six syntactic index hierarchies are defined in CAL. Three hierarchies, State_hierarchy, Event_hierarchy and Action_hierarchy discriminate among cases on the basis of states, events and action categories respectively. The other three hierarchies, Domain_hierarchy, User_hierarchy and Goal_hierarchy differentiate among cases based on the problem domain, DM characteristics and the major overall goals of the case respectively.

The bottom right window in Figure 5 partially depicts a simplified Action_hierarchy and User_hierarchy. Cases related to different sales promotion and advertising program designs (as indicated by the presence of the corresponding objects) are indexed autonomously by CAL under the appropriate leaf-node in the Action_hierarchy. Similarly information about the DM is used to index the case in the User_hierarchy.

Figure 5 about here

In addition, CAL allows for several DM-defined semantic index hierarchies. These semantic hierarchies aim to capture the functional relevance of the cases. Figure 6 displays two simple semantic categories, Solution_type and Implm_complexity (implementation complexity) which index cases based on the type of the solution (ranging from a minor improvement of existing

procedures to a radical rethinking of procedures) and the complexity of the solution implementation respectively.

Figure 6 about here

CAL provides a friendly interface (Figures 6 and 7) to support the DM in creating new hierarchies, modifying existing hierarchies and indexing cases in hierarchies. The same interface can also be used to manually search (as described algorithmically in Section 4.4) for cases indexed under specific hierarchies. The search patterns and short-listed cases are displayed in the top right window of Figure 5. The DM can choose specific hierarchies in the left window of Figures 6 and 7 and traverse down the chosen hierarchy via the display in the top right window of Figure 6.

While CAL autonomously indexes cases in the syntactic hierarchies by matching the state/action/event categories of objects in a case to the labels in the corresponding syntactic hierarchies, the DM is expected to shoulder the burden of indexing cases in the semantic hierarchies. This is necessary because CAL does not contain any domain knowledge to be able to understand the cases and relate the semantic hierarchy labels to the functional relevance of the cases. For example, CAL cannot determine whether the solution to a particular case represents a minor improvement or a radical rethinking.

To summarize, the major differences between the indexing mechanisms of CAL as compared to conventional CBRSs are the inclusion of both syntactic and semantic hierarchies and a higher degree of dependence on the DM for forming appropriate semantic hierarchies and indexing cases in them. While this does place an additional burden on the DM, it has some benefits. The system is simpler to implement and different DMs are able to adapt the system to their own particular needs by creating their own semantic hierarchies for the same cases. The semantic hierarchies also can be quite simple and still retain their usefulness.

4.3 Monitor

The monitor module is responsible for two tasks: to store the evolving decision process as a case in the case memory and to decide upon conditions calling for interventions in the current decision process.

4.3.1 Storing Decision Processes

A DM's interaction with a decision aiding CBRSs is an iterative and interactive process which does not typically represent a linear step-by-step approach to the final solution. Rather, it would be more like navigating through a maze where the DM goes through several alleys, turns and dead-ends before reaching the solution. The decision about which aspects of such a decision process to capture in a case is not easy due to the difficulty of determining exactly which part of the decision process would be most useful for aiding decision making and stimulating learning in a DM in the future.

Capturing complete information about decision states resulting from all actions and events would yield more information about the decision process but it would impose significant computational overheads. Note that an action implies each and every direct act (such as clicking on a specific button) committed by the DM in his interaction with the CBRS. To simplify matters and reduce overhead costs, selected actions/events can be used to trigger the storage of decision states in cases.

The Monitor module of CAL relies on the formation of a list of marked objects, i.e., selected objects which trigger the storage of decision states under certain conditions. By default, CAL uses the leaf-node labels of the indexing hierarchies (both syntactic and semantic) to "mark" objects in the knowledge-base of the associated DSS (Brandframe). The heuristic employed here is that the indexing labels signify functionally relevant aspects of the DSS's knowledge-base. The DM can add to or delete from the list of marked objects.

The marking algorithm marks an object if the index leaf-node label matches the object name. If a marked object has children objects, then all the children objects are also marked. Consider Figure 7 which represents a small part of the knowledge base of Brandframe. Given CAL's Action_hierarchy of Figure

5, it marks the all children objects of the node "Advertising" and selected children objects of the node "Sales_promotions" in Brandframe's knowledge base.

Figure 7 about here

If there are any changes in the attribute values of one or more marked objects during the decision process, the decision state is recorded by the Monitor module. Rather than store a dump of the entire knowledge base in a decision state, CAL compromises by only storing the attributes of the marked objects as attributes of the object representing the state in the case. Information about the action/event causing the change (such as the arrival of new data from on-line sources or a specific action initiated by the DM) in the marked object(s) is also recorded. CAL stores a complete dump of the knowledge base of the associated DSS at the start and at the end of the problem solution process. The DM can opt to disable case acquisition by CAL.

The DM is given the option to enter specific insights about and/or reflective observations on the decision process whenever the Monitor module stores a particular decision state. These comments serve the important purpose of aiding decision making and stimulating learning during future use of the case. The DM is encouraged to fill in comments and observations also because CAL does not contain domain knowledge to interpret the decision process autonomously. In addition, the DM is requested to provide information about himself and the major goals of the decision situation at the start and a short summary of the key lessons from the decision situation at the end.

4.3.2 Interventions in Decision Processes

Two types of interventions in decision processes are possible: active (at the initiative of CAL) and passive (at the request of the DM). Passive interventions are the responsibility of the Analyzer module and are explained in more detail in a later section. The Monitor module is responsible for the detection of conditions calling for active interventions.

The role of CAL in aiding and stimulating decision processes is similar to that of educational critiquing systems [11] in that it aims to provide decisional guidance and support learning in the DM. The design of intervention strategies in critiquing systems is complex and non-trivial. Several different factors [11] such as emotional impact on DMs, degree of intrusiveness in work process and the ability to accommodate different DM preferences and skills affect the design of appropriate critiquing strategies. Further, several different types of knowledge, such as knowledge of the domain, of the progress of the decision process thus far and of DM models can be brought to bear during interventions in the decision process. Although a topic of ongoing research, few normative guidelines exist for specifying intervention strategies.

The Monitor module uses heuristics to detect two types of intervention conditions:

(a) Decision intervention conditions: signaling stages in the decision process at which the DM would benefit from guidance regarding the decision process; and
(b) Reflection intervention conditions: identifying points in the decision process at which the DM can potentially benefit from reflecting on the decision process thus far and/or by comparing the current decision process with prior decision processes.

While decision intervention conditions aim to help the DM progress in the decision process, reflection intervention conditions have the more difficult objective of stimulating learning in the DM. The Monitor module keeps track of changes in the marked objects during the current decision process to detect the above intervention conditions.

The following types of decision intervention conditions are detected by the Monitor module:

- Immobility: The elapsed time since the last action taken by the DM is greater than a certain threshold. This may indicate that the DM does not know what to do next.

- Cycles: A DM can be stuck at a certain point in the decision process or be going around in circles as indicated by repeated cyclical changes to the same objects.
- Irrelevance: A DM may be working on the unimportant aspects of the decision situation. This is indicated by the elapsed time since the last case snapshot storage (of the current decision process) being greater than a certain threshold. The implicit assumption here is that the marked objects reflect important aspects of the decision situation and if the DM has not "touched" any of the marked objects for a long period of time then this implies that his/er attention is focused on other (not so important) aspects of the decision situation.
- Vacillations: a DM may be thrashing about in the decision process as indicated by overly frequent changes to the same marked object(s).

Upon detection of any of the above conditions, the Monitor module presents a message to the DM providing the option to extract a case from the case library (via the Analyzer module - see following sub-section) for decisional guidance.

The following types of reflection intervention conditions are detected by the Monitor module:

- Frequent decision interventions: if the Monitor module is detecting decision intervention conditions at a rate higher than a certain threshold, then the DM may wish to reflect on the current decision process and determine why this is the case.
- Termination: the DM is requested to reflect on the decision situation prior to terminating his/er interaction with the system.
- Prior case view: if CAL has retrieved and displayed a prior case to the DM, s/he is requested to enter his/er reflective observations comparing the current decision situation with the prior case.

Upon detection of any of the above conditions, the Monitor module presents a message to the DM requesting him/er to reflect on the current decision situation and to summarize his/er observations in a textual comment to be stored with the case of the current decision situation.

The intervention strategies of CAL are based on relatively simple heuristics and do not provide a comprehensive and accurate detection of all intervention conditions. For example, the Monitor module cannot determine whether the DM has a coherent solution strategy while making repeated changes in the same object. With more extensive domain and DM modeling knowledge, it is conceivable to introduce more sophisticated intervention strategies. However, this is beyond the current scope of CAL.

In summary, the monitor module of CAL performs a function which is not seen in conventional CBRSs, namely that of monitoring the current decision process, detecting conditions calling for active interventions by the CBRS in the decision process and storing the current decision process as a case in the case memory. While the precise mechanisms and degree of sophistication for monitoring and intervention may vary in other decision aiding CBRSs, the core functionalities of the Monitor module are not observed in conventional CBRSs.

4.4 Analyzer

The Analyzer module retrieves cases from the case memory at the request of either the DM or the Monitor module (Figure 3). The output of the Analyzer module is presented to the DM by the Adviser module.

A DM can request to retrieve a case under two conditions:

- Context-independent: a DM can wish to retrieve a case independent of the current decision process or the currently active case;
- Context-dependent: a DM may want to retrieve a case related to the current decision process or the currently active case.

Context-independent retrieval offers two possibilities:

- Retrieval by name: the DM can ask for a list of all cases currently contained in the case memory and select a particular case from the list.
- Retrieval by index hierarchy traversal: the DM can traverse down one or more index hierarchies, H_i and select a case according to the procedure summarized in Figure 8. The CAL interface shown in Figures 6 and 7 allow the DM to perform the retrieval.

Figure 8 about here

To activate context-dependent retrieval, the DM has to indicate the context to be used to guide the retrieval process. The context is provided by the currently active case, C_{cur}. Note that C_{cur} can either be a previously stored case which is being viewed by the DM or the partially stored case of the current decision process. The retrieval procedure consists of two parts: (a) obtaining the set of relevant cases and (b) choosing the most relevant case.

CAL uses a preference heuristic to make two lists, L_{union} and L_{inter} of relevant cases. The heuristic used is that cases indexed under the same leaf nodes in the index hierarchies are similar. While all cases in L_{inter} share the same index labels as C_{cur}, cases in L_{union} share at least one common index label with C_{cur}. The two lists, L_{union} and L_{inter} are defined in Figure 9. Note that L_{inter} is a subset of L_{union}. CAL first uses L_{inter} to retrieve the most relevant case. If L_{inter} is empty then CAL uses L_{union}.

Figure 9 about here

If L_{union} is empty then this indicates that there are no other cases sharing one or more index labels with C_{cur}. There are three further options in such a situation: (a) search through all cases in the case memory; or (b) request the DM to form the short-list of relevant cases as described in Figure 8; or (c) use another case indexed under a sibling node of the node under which C_{cur} is indexed to form L_{union} and L_{inter}. By default, CAL recommends the second choice to the DM as it represents the easiest and most accurate option. If for any reason, C_{cur} is not indexed in the index hierarchies, then the DM is again asked to form the short-list of relevant cases as described in Figure 8.

The choice of the most relevant case is determined by a similarity metric which includes either structural or semantic aspects. While structural similarity matches only object-attribute names, semantic similarity matches both object-attribute names and object-attribute values. A case, C_{ret}, retrieved from the case library on the basis of structural similarity would imply that both C_{ret} and C_{cur} have (partially) the same attribute labels though the attributes may not necessarily have the same values in C_{ret} and C_{cur}. The use of semantic similarity would mean that both C_{ret} and C_{cur} have

(partially) the same attributes with the same values. Definitions of the computation of simple structural and semantic similarities are provided in Figure 9.

CAL provides for the following additional features in the retrieval process:

- It distinguishes between generic attributes (attributes common to a certain class of objects such as states or events or actions) and specific attributes (attributes defined locally in a particular object).
- It allows the DM to indicate attributes to be given special importance (or to be ignored) during the retrieval process.
- It provides the option to build and access a "similarity table" which lists the degree of similarity between syntactically different but semantically related terms (such as "width" and "breadth").

The above distinctions between different types of attributes are accounted for by the weights $W_i^{cur,p}$ in the definitions of Figure 9. These weights change according to the attribute $F_i^{cur,p}$ and are different, for instance, for special DM-indicated attributes and generic/specific attributes.

When the Monitor module detects conditions calling for interventions on the part of CAL, it requests the Analyzer module to retrieve an appropriate case from the case memory. The retrieval procedure followed then is similar to the context-dependent retrieval described above.

Case retrieval is important because it directly determines the degree of decision support possible with a CBRS. Different approaches to case retrieval have been debated extensively in the literature [20] without a conclusive agreement on any normative approach. CAL's retrieval algorithm utilizes several concepts from other CBRSs such as structural and semantic similarities and the use of preference heuristics. A distinction between CAL and conventional CBRSs with respect to case retrieval is that the DM can potentially play an important role in the retrieval procedures of CAL. This is related to the close interactive relation between CAL and the DM, an aspect which is not significant in conventional CBRSs.

4.5 Adviser

Conventional CBRSs adapt the retrieved case to present a "solution" to the DM. While several adaptation techniques such as reinstantiation, parameter adjustment, commonsense transformation and model-guided repair have been proposed in the literature, adaptation procedures are complex, difficult to generalize, and remain the Achilles heal for conventional CBRSs. Thus, Kolodner and other researchers [21] has recommended that decision aiding CBRSs should leave the responsibility for the adaptation procedures upto the DM. In accordance with this trend, the Adviser module leaves the actual adaptation to arrive at a solution to the DM but stimulates and facilitates the DM's decision process on the basis of the case retrieved by the Analyzer module.

The use of artificial intelligence techniques for facilitation and learning ranges from open-ended exploratory learning environments such as Logo [25] which give DMs near unlimited control to system-guided tutoring environments such as Lisp TUTOR [1] which step users through a pre-designed "learning" curriculum. Finding the right balance between exploratory learning and system-directed learning is a delicate issue which is dependent upon several things including the specific characteristics of both the DM and the problem context. In the context of CAL, the Monitor module provides system-directed intervention and the Adviser module provide the complementary open-ended exploratory environment to facilitate decision making and enhance learning in the DM.

Given a DM's focus on a particular object (state or event or action) in the currently active case, the Adviser module allows the DM to browse through the case with the following types of links/associations:

- Temporal: the DM can choose to look at the temporal order of states, events and actions in the retrieved case.
- Comparative: the DM can opt to look at other comparable states or events or actions relative to the currently "active" object. Similarities across objects are computed using either structural or semantic approaches as outlined in Figure 9.
- Linked: the DM may wish to browse through the case along certain DM-defined ordering sequences imposed on the case objects (as

explained in more detail below). These DM-defined order sequences may or may not follow the temporal ordering of objects.

- DM specified: the DM can "mark" one or more specific attributes in a state, event or action in the retrieved case and search for other states, events, or actions respectively with the same attribute(s).
- Reflective observations: the DM can choose to observe stages in the retrieved case where prior decision makers may have entered some reflective observations on the case or on its usefulness for decision making in different situations. The content of these observations may provide specific insights to the DM for the current problem situation.
- Relevant cases: if the retrieved case was used previously to retrieve another similar case, then a list of such similar cases is stored with the retrieved case. The DM can access this list and decide to view other similar cases.

Let us illustrate some of the above possibilities with the example of the retrieved case displayed in Figure 4. In Figure 4 the objects of the case are displayed in a temporal order and the DM can directly observe the temporal sequence of the states, events and actions. If the DM clicks on the "Suggest" button in Figure 4, then the Adviser module suggests other linked states, events or actions (relative to the currently active object). The case of Figure 4 is reproduced in Figure 10 along with a few additional objects. Assume that the currently active state is "market_share_slightly_down", then the following links are available for browsing as shown in Figure 10:

- Before/after: they provides access to the object temporally before and after the current object;
- Next_Similar: this points to the (temporally) next similar state in the case;
- Decreasing market share: this points to the state with the next lower market share relative to the market share associated with the current state. This is an example of a DM-defined link which does not follow the temporal order of objects in the case.

Figure 10 about here

Note that the Monitor and Adviser modules complement each other. The Monitor module tracks the progress of the DM as s/he goes through the retrieved case to detect appropriate system-directed intervention points.

When the Monitor module flags a decision-intervention condition, the Adviser module tries to facilitate decision making by indicating the above range of exploration options to the DM. The Adviser module keeps track of the browsing pattern of the DM. The DM can flexibly choose any browsing pattern and look at specific objects in the case.

To summarize, the Adviser module provides a functionality which is not present in conventional CBRSs. Lacking detailed domain knowledge and appropriate user models, the adviser module attempts to facilitate decision aiding and learning by suggesting different linkages for browsing through cases and suggesting other cases for retrieval. It does not attempt to adapt prior cases or even suggest a "solution" to the DM. These actions are the responsibility of the DM. The hope is that browsing retrieved cases through associated links will aid lateral thinking and stimulate the DM into creative problem solution. The Adviser module has commonalties with educational critiquing systems [11]. More details on this are provided in the next section.

5 Conclusion

This section highlights the contributions of this research relative to prior research, outlines the limitations of this research along with directions for future work.

5.1 Comparisons with Prior Research

This research has first emphasized the importance of a process orientation in decision aiding. Next, it has described CAL, an example of a decision aiding CBRS which illustrates the radical change in the design of a CBRS called for by an emphasis on decision processes. The highlights of these two contributions and their comparisons with prior research are summarized below.

CBRSs have been largely designed and described in the literature as "automated" problem solvers. The CBR procedure of conventional CBRSs as described in Figure 2 can be seen to consist of three major parts: the case memory, the case retrieval procedure and the case adaptation procedure. The case memory in conventional CBRSs emphasizes the problem and solution descriptions in cases. The retrieval procedure matches

current problem descriptions with case problem descriptions and the adaptation procedures adapt the retrieved case solution to the current problem.

An important change in CAL is the shift from the linear "match-retrieve-adapt" solution methodology of conventional CBRSs (Figure 2) to an iterative "monitor-analyze-advise" cycle with a central emphasis on the DM (Figure 3). Cases in decision aiding CBRSs such as CAL give more importance to the decision process used to obtain the solution. The retrieval procedure focuses on the identification of appropriate decision-intervention and reflection-intervention features. The DM is largely responsible for the adaptation procedures. Depending upon the level of decision support offered, the CBRS may offer some critiquing with respect to the decision process being followed by the DM.

Because conventional CBRSs produce the solution for a "passive" user, they must contain all the knowledge needed to solve the problem. Such knowledge includes matching heuristics, commonsense and causal knowledge needed for retrieval and adaptation, and the sequence of steps needed to derive the solution. In contrast, CAL relies on the DM playing an active role in the decision process. This reduces the need to contain detailed domain and commonsense knowledge. CAL aims to augment rather than replace the decision making capabilities of the DM. The major differences between conventional and decision aiding CBRSs are summarized in Figure 11.

Figure 11 about here

Some decision aiding CBRSs have been described in the literature. Battle Planner [13], one of the first decision aiding CBRSs reported in the literature was described in Section 3.2. CLAVIER [16] is another widely quoted decision aiding CBRS used to generate autoclave load schedules within Lockheed. A case in CLAVIER is a graphical representation of autoclave loading layouts. It allows the shop floor worker to modify or choose between recommended layouts by comparing it with other layouts in the case library. ARCHIE-2 [7] is a recent example of a decision aiding CBRS in the domain of architectural design. ARCHIE-2 uses an annotated graphical schematic of a building and allows the DM to designate a particular part of the design or a

particular design issue to concentrate upon. The system then recounts the story annotated to the design issue or design sub-part chosen by the DM. By partially anticipating the requests of the DM, ARCHIE-2 attempts to recreate a virtual conversation between colleagues working under similar circumstances and trading stories with each other.

Most decision aiding CBRSs in the literature act as smart retrieval systems. They largely ignore the decision processes of DMs. As mentioned in Sections one and two, knowledge about decision processes can play an important role in constructively aiding decision making. CAL explicitly captures and exploits decision process related information as explained earlier. A move towards such an (process) emphasis can be observed in recent decision aiding CBRSs. For example, ARCHIE-2 monitors the user's decision process in order to anticipate user requests. Another CBRS co-designed by the author [2,3] in the domain of multi-criteria decision making incorporates a CBR agent which monitors the decision process of the DM. The CBR agent can adopt two roles: those of a storyteller or a adviser. In the story teller role, the agent simply replays prior cases to the DM as a "story". In the adviser mode, it uses its knowledge about the structure of multi-criteria decision problems to suggest different alternatives, criteria and weights to the DM. CAL represents a more expanded and detailed implementation of these ideas.

CAL can also be compared to critiquing agents and intelligent tutoring systems [11,31] which have a more participatory involvement in the DM's decision process. An example is ACTIVIST [12], an active help system for a text editor. It volunteers information to a user at the appropriate moments and adjusts its intervention strategy depending upon the responses of the user. For example, ACTIVIST ceases to critique actions when the user ignores its suggestions. LISP CRITIC [10] is an example of a passive critic (i.e., it has to be invoked by the user) to support programmers. It incorporates a user modeling component to maintain knowledge about the domain and goals of each individual user. Another critiquing system FRAMER [22] utilizes an extensive knowledge-base of design rules to evaluate the design of window-based user interfaces. FRAMER continuously displays a checklist of relevant issues for consideration by the DM in a window entitled "Things to be taken care of".

From the above limited descriptions it is evident that critiquing and intelligent tutoring systems share a common purpose with CAL - to facilitate decision making and possibly enhance learning in the DM. However, the methods and approaches used are different. Critiquing systems usually rely extensively upon detailed domain knowledge and user models for effective critiquing strategies. In contrast decision aiding CBRSs depend predominantly upon cases in the case memory. While some critiquing systems do refine their domain knowledge and user models with time, decision aiding CBRSs tend to learn primarily via the acquisition of new cases, i.e., decision processes of problem solving experiences. For example, CLAVIER originally started with 20 cases but had acquired through use over 300 cases by 1993. Despite these differences, there are useful synergies between critiquing systems and decision aiding CBRSs which need to researched further.

5.2 Limitations of Research and Future Challenges

"Case-based reasoning is still relatively young. Though it has much promise, there have been few industrial-strength systems built to date. Thus, the engineering of case-based systems has not yet received as much attention as has the engineering of, say, rule-based expert systems". This recent quote by Janet Kolodner [20, p. 531] illustrates some of the challenges in building and validating CBRSs. A major challenge for this research lies in enhancing and validating CAL as explained below.

CAL is a decision aiding CBRS which has to be used in conjunction with an appropriate knowledge-based DSS. Thus any implementation and validation of CAL is dependent upon the associated DSS. CAL has currently been developed as a CBR component for Brandframe, a DSS for brand managers. Brandframe has been implemented over the past two years in a large Dutch company in the domain of fast moving consumer goods. Implementing the real-life version of Brandframe has been a relatively slow process because it had to be customized in detail for the product groups of particular brand managers. Currently Brandframe has been implemented for two product groups and an experimental validation of the utility of Brandframe has been performed (see [6,9] for more details).

The implemented version of Brandframe does not include CAL because it has been necessary to keep the system easy to learn, to ensure that it works efficiently and to give the brand managers adequate time to adjust to working with a knowledge-based DSS. Thus, while an expanded implementation of Brandframe is continuing within the Dutch company, it has not been possible to incorporate CAL as yet.

However, one should be aware of the several challenges in validating CAL. First, a reasonably good set of decision processes would need to be stored as cases in the case memory. Note that it would not be possible to test the "response" of CAL to some previously encountered "problem-solution" pairs (as is commonly done in some evaluations of conventional CBRSs) because the aim of CAL is not to produce the "right" solution, but to facilitate decision making and hopefully enhance learning in the DM. This brings us to the next challenge: to devise an appropriate set of metrics to capture the value added by CAL in facilitating decision making and enhancing learning in the decision maker. Finally, the specific characteristics of the DM and problem would make it difficult to draw general conclusions on the effectiveness of decision aiding CBRSs such as CAL for decision aiding and stimulating learning.

From a technical perspective, CAL could benefit from several enhancements. For example, the monitoring and decision-intervention strategies of CAL can be made more sophisticated. Several ideas exist in the literature of critiquing and intelligent tutoring systems which can be adapted to CAL to provide more intelligent and creative support for facilitating decision making and enhancing learning in the DM. Currently, CAL does not incorporate any domain knowledge or user models. These extensions present multiple opportunities for enhancing the decision support capabilities of CAL. All these possibilities can in principle make CAL more responsive to the DM by enabling the dynamic modification of the degree of intrusiveness and mode of interventions. The currently implemented version of CAL has also benefited from several simplifications. For example, the heuristic of only storing marked objects in states makes the case storage methods simpler but reduces the amount of information available to the DM. In an ideal world, a dump of the entire knowledge base should be stored at each state.

It is possible to think of several additional suggestions, but they would all increase the complexity of building and validating the system. The critical decision in this context is not the ability to incorporate as many features as possible, but in understanding and including only those decision support capabilities which are most functionally relevant to the DM. To conclude, it would be appropriate to reiterate that the design of process-focused decision aiding CBRSs is very different from conventional CBRSs. Though CAL represents a simplified implementation, it helps to highlight these differences.

References

1. J.R. Anderson, R.G. Farrell and R. Sauers, Learning to Program in LISP, Cognitive Science, 8(2), pp. 87-129, Apr.-June 1984.
2. A. Angehrn and S. Dutta, Case-based Decision Support, INSEAD Working Paper 93/16/TM, 1993. Currently under 2nd review with the Communications of the ACM.
3. A. Angehrn and S. Dutta, Integrating Case-Based Reasoning in Multi-Criteria Decision Support Systems, in Decision Support Systems: Experiences and Expectations, IFIP Transactions A-9, T. Jelassi, M.R. Klein, and W.M. Mayon-White (Eds.), pp. 133-150, North Holland, 1992.
4. J.G. Carbonell, Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition, in Machine Learning, 2, pp. 371-392, 1986.
5. G. Collins and L. Birnbaum, Problem-Solver State Descriptions as Abstract Indices for Case Retrieval, in Working Notes of the 1990 AAAI Spring Symposium on Case-Based Reasoning, pp. 32-35, Stanford, CA., 1990.
6. A. Dalebout, Management Support for the Product Manager, Unpublished Masters Thesis, Rotterdam School of Management, Erasmus University, Holland, 1993.
7. E. Domeshek and J. Kolodner, Finding the Points of Large Cases, in Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM), 7(2), pp. 87-96, 1993.
8. N.M. Duncan, Case-Based Reasoning Applied to Decision Support Systems, Master's Thesis, Queen's University, Kingston, Canada, 1989.
9. S. Dutta, B. Wierenga and A. Dalebout, An Integrated Perspective on Designing Management Support Systems, INSEAD Working Paper no. 93/15/TM, 1993. Currently under 2nd review with the ACM Transactions on Information Systems.
10. G. Fischer and T. Mastaglio, Computer-based Critics, in Proceedings of the 22nd Annual Hawaii Conference on System Sciences, Vol. III: Decision Support and Knowledge Based Systems Track, IEEE Computer Society, pp. 427-436, Jan. 1989.

11. G. Fischer, A.C. Lemke, T. Mastaglio and A.I. Morch, The Role of Critiquing in Cooperative Problem Solving, ACM Transactions on Information Systems, 9(3), pp. 123-151, April 1991.
12. G. Fischer, A.C. Lemke and T. Schwab, Knowledge-based Help Systems, in Human Factors in Computing Systems, CHI'85 Conference Proceedings, pp. 161-167, ACM, New York, April 1985.
13. M. Goodman, CBR in Battle Planning, in Proceedings of the DARPA Workshop on Case-Based Reasoning, Vol. II, K. Hammond (Ed.), Morgan Kaufmann, San Mateo, CA., pp. 246-269, 1989.
14. G.A. Gorry and M.S. Scott-Morton, A Framework for Management Information Systems, SMR Classic Reprint, Sloan Management Review, pp. 49-61, Spring 1989.
15. K.J. Hammond, Case-Based Planning: A Framework for Planning from Experience, Cognitive Science, 14, pp. 385-443, 1990.
16. D.H. Hennessy and D. Hinkle, Applying Case-based Reasoning to Autoclave Loading, IEEE Expert, 7(5), pp. 21-26, 1992.
17. T.R. Hinrichs, Problem Solving in Open Worlds: A Case Study in Design, Erlbaum, Northvale, NJ, 1992.
18. J.L. Kolodner and M.Y. Jona, Case-Based Reasoning: An Overview, Technical Report #15, Northwestern University, The Institute for the Learning Sciences, June 1991.
19. J.L. Kolodner and R.L. Simpson, The MEDIATOR: Analysis of an Early Case-Based Problem Solver, Cognitive Science, 13(4), pp. 507-549, 1989.
20. J.L. Kolodner, Case-based Reasoning, Morgan Kaufmann, San Mateo, CA, 1993.
21. J.L. Kolodner, Improving Human Decision Making through Case-Based Decision Aiding, AI Magazine, pp. 52-68, Summer 1991.
22. A.C. Lemke, Design Environments for High-Functionality Computer Systems, Ph.D. Thesis, Dept. of Computer Science, University of Colorado, Boulder, July 1989.
23. H. Margolis, Patterns, Thinking and Cognition, University of Chicago Press, Chicago, 1988.
24. C. Owens, Integrating Feature Extraction and Memory Search, Machine Learning, 10, pp. 311-339, 1993.
25. S. Papert, Mindstorms: Children, Computers and Powerful Ideas, Basic Books, New York, 1980.

26. C.K. Riesbeck and R.C. Shank, Inside Case-Based Reasoning, Lawrence Erlbaum Associates Inc., NJ, 1989.
27. H.S. Shinn, Abstractional Analogy: A Model of Analogical Reasoning, in Proceedings of DARPA Workshop on Case-based Reasoning, Clearwater, Florida, Morgan Kaufmann, San Mateo, CA, 1988.
28. M.S. Silver, Systems that Support Decision Makers: Description and Analysis, John Wiley, 1991.
29. H.A. Simon, The New Science of Management Decisions, Harper & Row, New York, 1960.
30. S. Slade, Case-Based Reasoning: A Research Paradigm, AI Magazine, pp. 42-55, Spring 1991.
31. D.H. Sleeman and J.S. Brown, Eds., Intelligent Tutoring Systems, Academic Press, London/New York, 1982.
32. D. Waltz, Is Indexing used for Retrieval?, Proceedings of the Case-Based Reasoning Workshop (DARPA), II, pp. 41-44, Morgan Kaufmann, San Mateo, CA., 1989.
33. M. Zeleny, Cognitive Equilibrium: A New Paradigm of Decision Making?, Human Systems Management, 8, pp. 185-188, 1989.

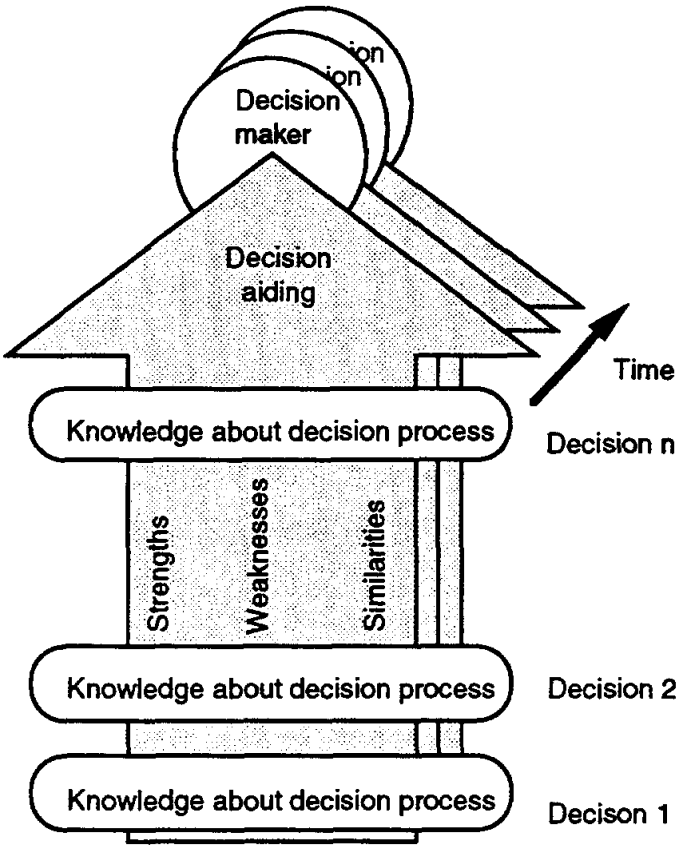


Figure 1: Using knowledge of prior decision processes

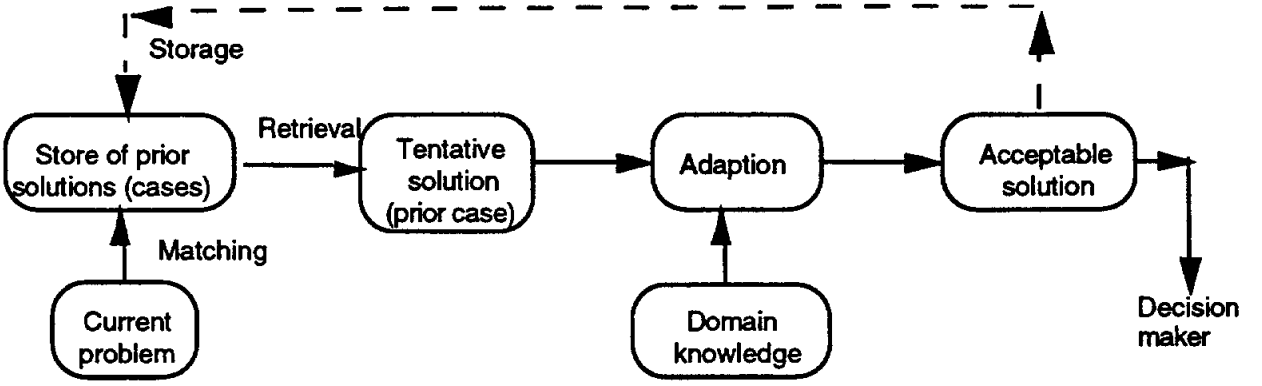


Figure 2: Case-based reasoning process

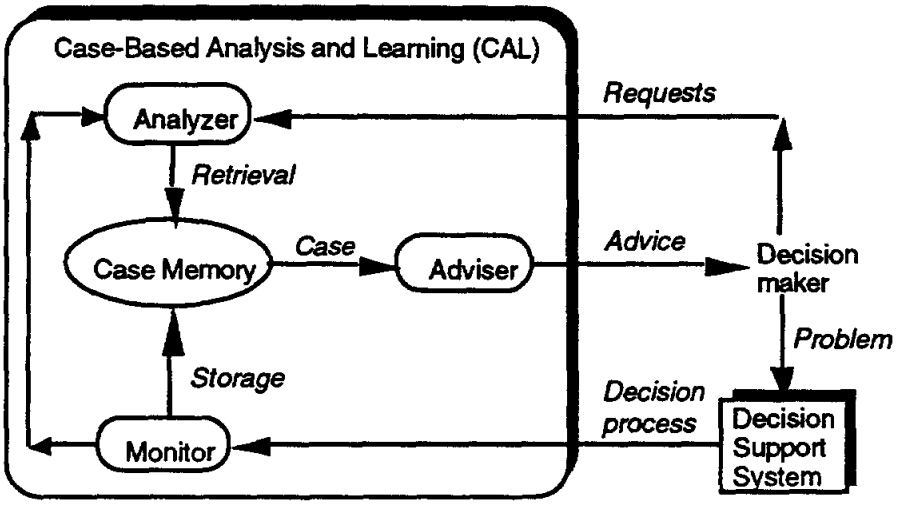


Figure 3: Structure of CAL

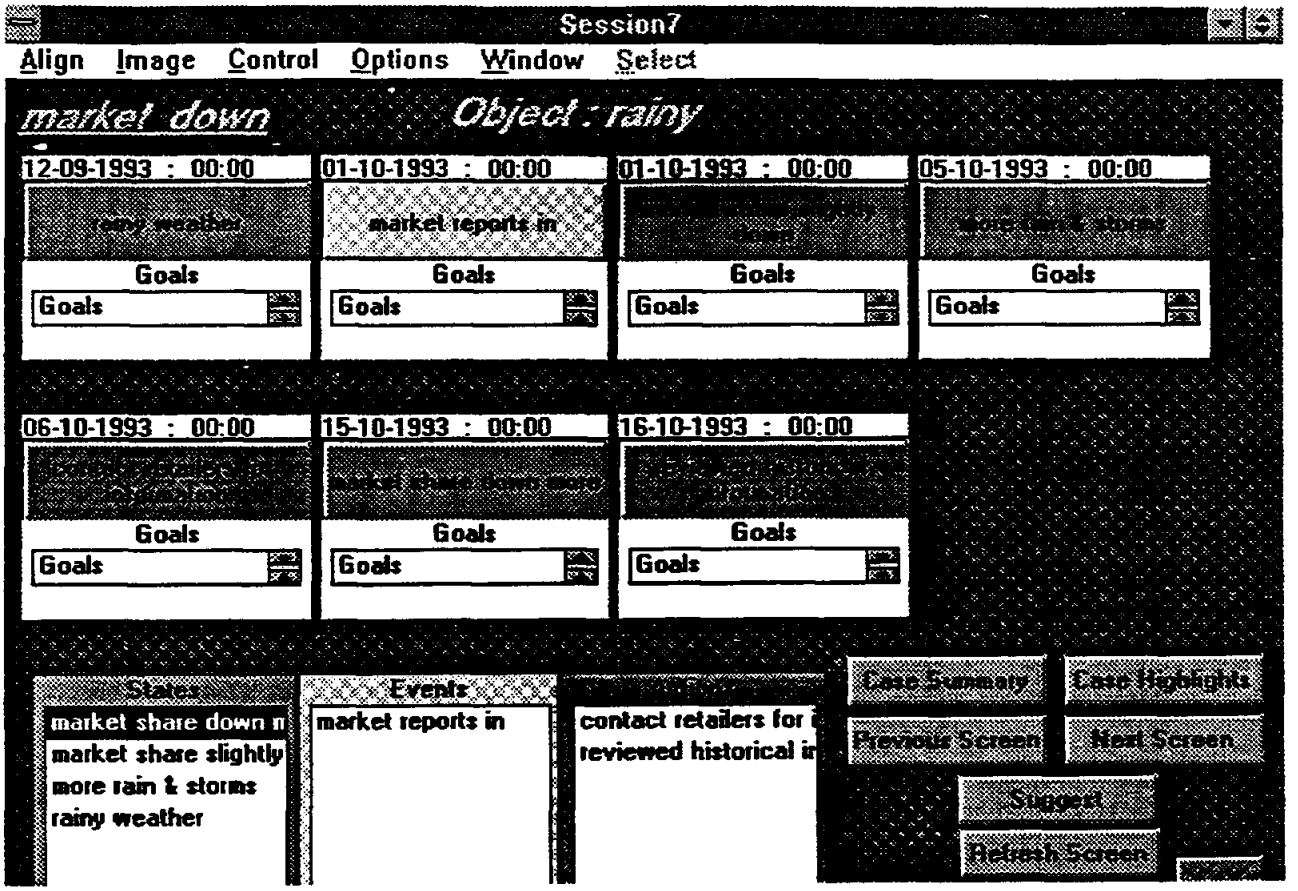


Figure 4: Display of a case in CAL

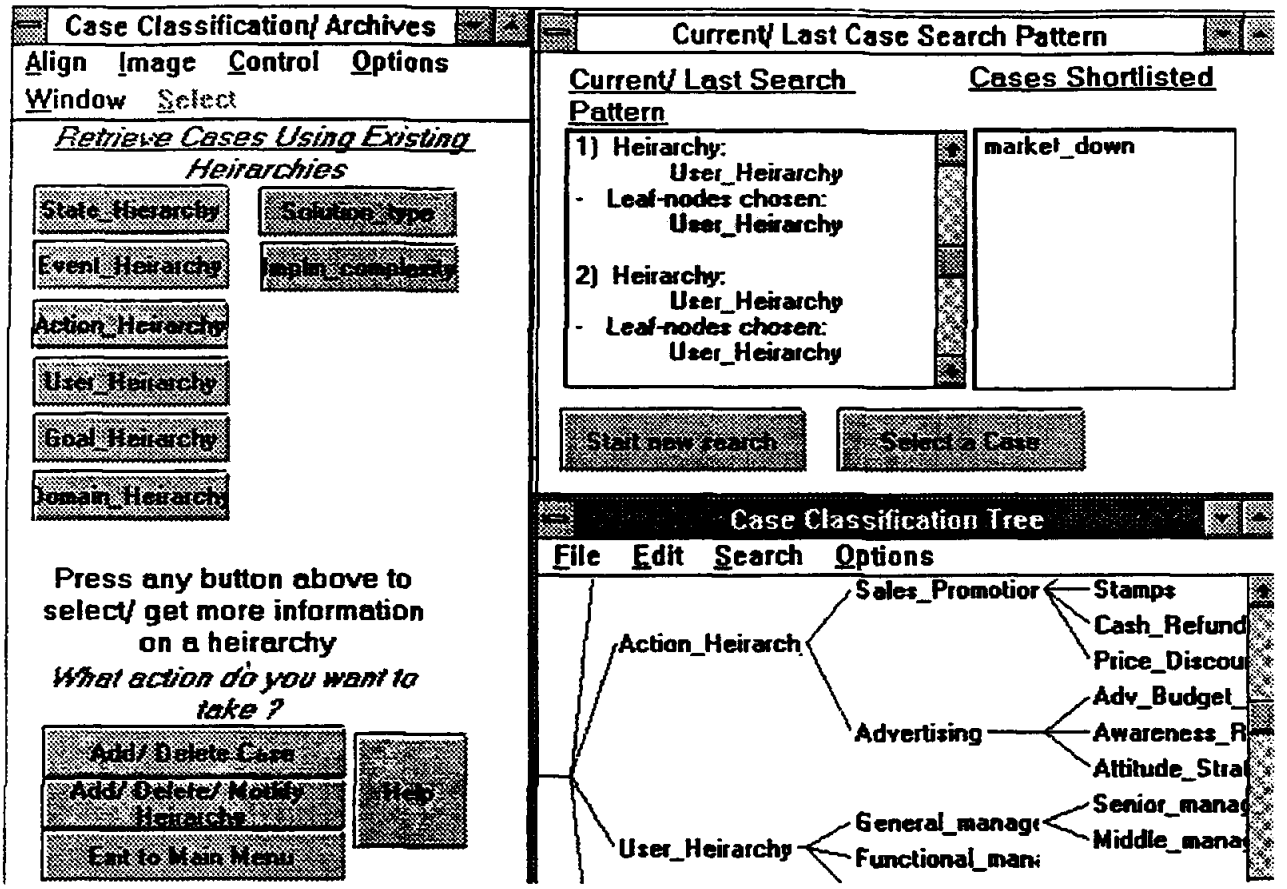


Figure 5: Syntactic hierarchies in CAL

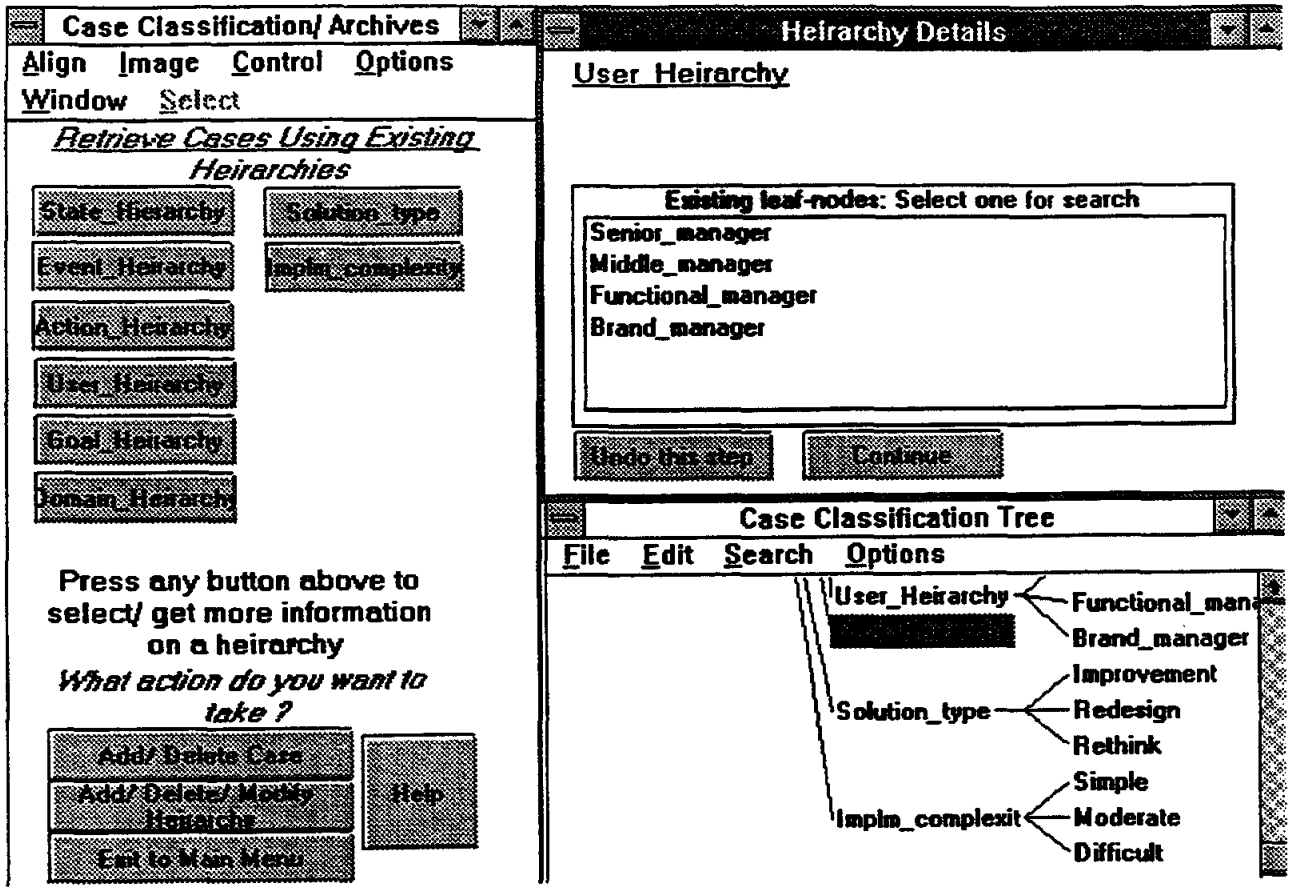


Figure 6: Semantic hierarchies in CAL

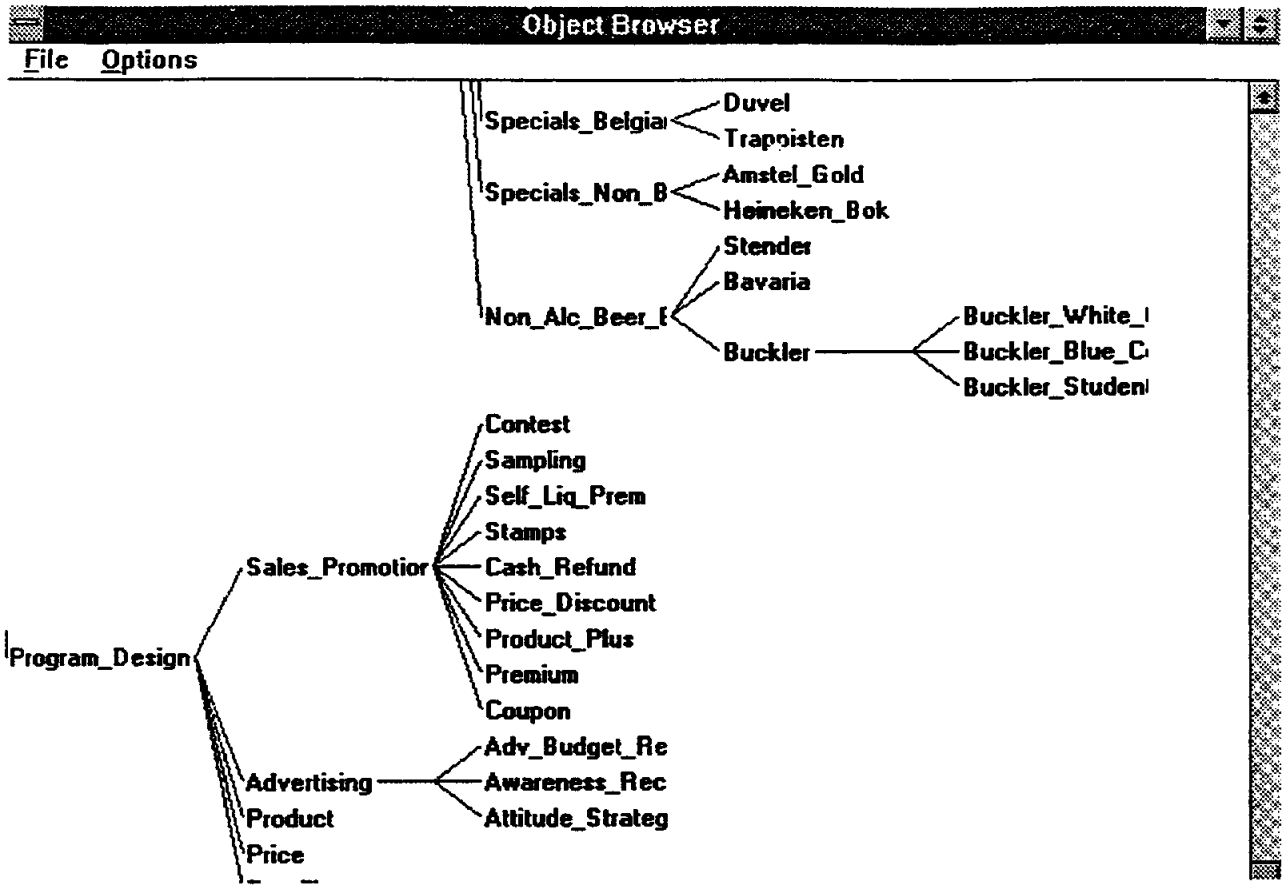


Figure 7: Partial knowledge-base of Brandframe

! Comment lines begin with an !
! H_i is the ith index hierarchy out of a total of n hierarchies
! N^i_j is the jth leaf node in H_i
! C^{ij}_k is the kth case indexed under N^i_j
! L is the list of possible cases available to the DM

$L \leftarrow \Phi$
Let DM choose H_i, $i \in \{1,2,\ldots,n\}$
Let DM traverse down H_i till a leaf node N^i_j is reached
$L \leftarrow L \cup \{\cup_k C^{ij}_k\}$
If DM selects a case from L then exit else
    Do until DM makes a selection from L or chooses "exit"
        Let DM choose H_m, $m \in \{1,2,\ldots,n\}$
        Let DM traverse down H_m till a leaf node N^m_j is reached
        $L \leftarrow L \cap \{\cup_k C^{mj}_k\}$
        If L is empty then exit
    Loop

Figure 8: Retrieval by index hierarchy traversal

! H_i is the ith index hierarchy out of a total of n hierarchies
! N^i_j is the jth leaf node in H_i; Child (N^i_j) is a case indexed under node N^i_j
! C^{ij}_k is the kth case indexed under N^i_j
! C_{cur} is the currently active case
! L_{union} is the maximal list of possible cases to be searched
! L_{inter} is the minimal list of possible cases to be searched
! The first step is to form L ,the list of possible cases to be searched:

If C_{cur} is indexed in the index hierarchies, then $L \leftarrow L_{inter}$ where:

$$L_{inter} = \cap_i \{\cap_j \{\cup_k \{C^{ij}_k\}\}\}$$

$$\text{if } C^{ij}_k = \text{Child } (N^i_j) \wedge C_{cur} = \text{Child } (N^i_j) \wedge C^{ij}_k \neq C_{cur}$$

If L is empty then $L \leftarrow L_{union}$ where

$$L_{union} = \cup_i \{\cup_j \{\cup_k \{C^{ij}_k\}\}\}$$

$$\text{if } C^{ij}_k = \text{Child } (N^i_j) \wedge C_{cur} = \text{Child } (N^i_j) \wedge C^{ij}_k \neq C_{cur}$$

! If C_{cur} is not indexed in the index hierarchies, then form L by asking DM (see Figure 8)

! The next step is to choose the most relevant case C_{ret} (for C_{cur}) from L
! $F_k^{i,j}$ is the kth attribute label in the jth object, O_j^i (which could be either a state or an event or
! an action) in the ith case, C_i
! $V(F_k^{i,j})$ is the value of attribute $F_k^{i,j}$; $W_k^{i,j}$ is a DM-defined weight for attribute $F_k^{i,j}$.
! St_score and Sm_score are the structural and semantic similarity scores respectively

Loop for all cases C_i in L
  $St\_score \leftarrow 0$
  $Sm\_score \leftarrow 0$
  Loop for all objects O_j^i in C_i
  Loop for all objects O_p^{cur} in C_{cur}
   If O_j^i and O_p^{cur} are of the same type (i.e. action/event/state) then:
    Loop for all attributes $F_k^{i,j}$ in O_j^i
     Loop for all attributes $F_l^{cur,p}$ in O_p^{cur}

$$St\_score \leftarrow St\_score + W_l^{cur,p} * f_{Struct}(F_l^{cur,p}, F_k^{i,j})$$

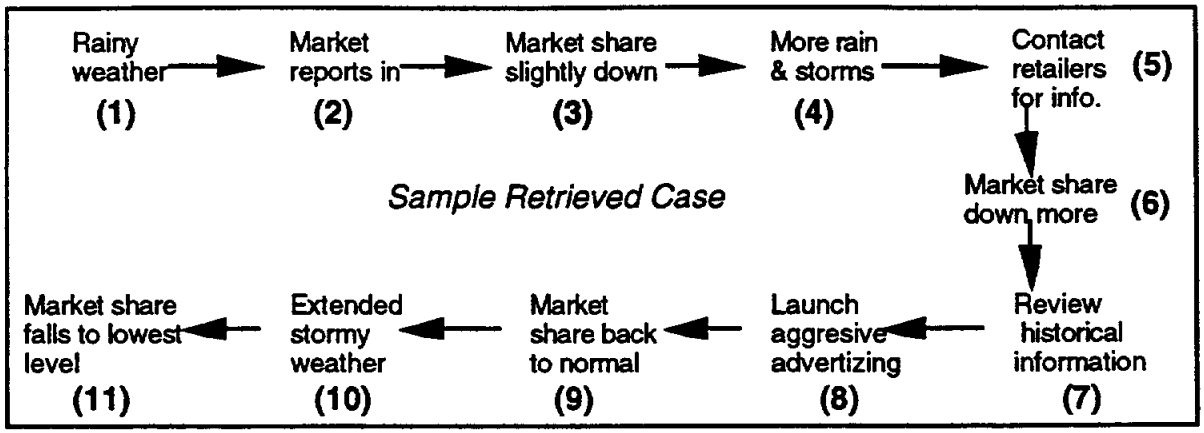
$$\text{where } f_{Struct}(F_l^{cur,p}, F_k^{i,j}) = 1 \text{ if } (F_l^{cur,p} = F_k^{i,j})$$
$$= 0 \text{ else}$$

$$Sm\_score \leftarrow Sm\_score + W_l^{cur,p} * f_{Sem}(F_l^{cur,p}, F_k^{i,j})$$

$$\text{where } f_{Sem}(F_l^{cur,p}, F_k^{i,j}) = 1 \text{ if } (F_l^{cur,p} = F_k^{i,j}) \wedge V(F_l^{cur,p}) = V(F_k^{i,j})$$
$$= 0 \text{ else}$$

End-all-loops
Assign C_{ret} as case in L with highest structural or semantic similarity score (as desired by DM)

Figure 9: Context-dependent retrieval algorithm



Sample Linkages:

Temporal order: 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8- 9 - 10 - 11

Similarity orders: 1 - 4 - 10 (rainy weather states)
3 - 6 - 9 - 11 (market share change states)

Lower_mkt_share order: 9 - 6 - 3 - 11 (order of decreasing market share)

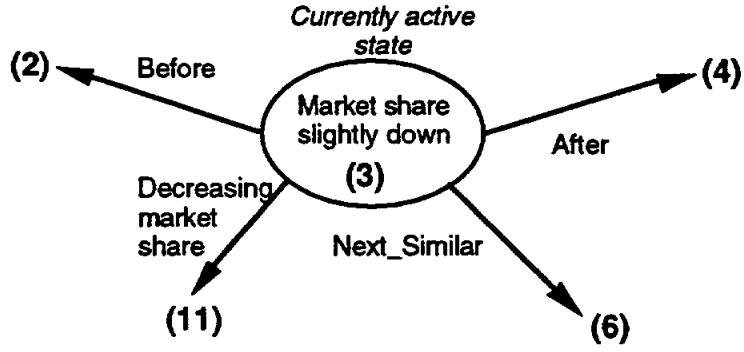


Figure 10: Link sequences used by Adviser module

	Conventional CBRSs	Decision aiding CBRSs
Case Memory	Problem description and solution to problem	Decision process
Retrieval procedure	Match current problem description with case problem description	Identify decision intervention features in current decision process
Adaptation procedure	Adapt case solution to current problem	Decision maker largely responsible for adaptation; System may provide some critiquing on decision process

Figure 11: Characteristics of conventional and decision aiding CBRSs

List of Figure Captions