Applied Intelligence, ??, 1–25 (1999)

Conversational Case-Based Reasoning

DAVID W. AHA[‡], LEONARD A. BRESLOW[‡], HECTOR MUÑOZ-AVILA[†‡]
[‡]*Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Code 5510, 4555 Overlook Ave, SW, Washington, DC 20375-5337*
lastname@aic.nrl.navy.mil | (202) 404-4940 | FAX: 767-3172

[†‡] *Department of Computer Science, University of Maryland, College Park MD 20742-3255*
munoz@cs.umd.edu | (301) 405-2684 | FAX: 405-6707

Received 19 February 1999; Revised 3 September 1999

Editors: David W. Aha & Hector Muñoz-Avila

Abstract. Conversational case-based reasoning (CCBR) was the first widespread commercially successful form of case-based reasoning. Historically, commercial CCBR tools conducted constrained human-user dialogues and targeted customer support tasks. Due to their simple implementation of CBR technology, these tools were almost ignored by the research community (until recently), even though their use introduced many interesting applied research issues. We detail our progress on addressing three of these issues: simplifying case authoring, dialogue inferencing, and interactive planning. We describe evaluations of our approaches on these issues in the context of NaCoDAE and HICAP, our CCBR tools. In summary, we highlight important CCBR problems, evaluate approaches for solving them, and suggest alternatives to be considered for future research.

Keywords: Conversational case-based reasoning, case library revision, dialogue inferencing, plan authoring, ModSAF

1. Introduction

A gap exists between research and applications of case-based reasoning (CBR). In most research systems, users input a *complete* problem description. In contrast, most commercial CBR systems (e.g., Inference Corporation's k-Commerce) elicit queries via an interactive questioning process. Low research interest in *Conversational* CBR (CCBR) may reflect the implicit assumption that CCBR offers a simple usability enhancement, but poses no new research challenges relative to noninteractive CBR. The following anecdote casts doubt on this assumption.

In 1996 members of the Vertical Launching System Engineering Division (VLSED) of the Naval Surface Warfare Center at Pt. Hueneme, California were struggling with issues on how to assist maintenance personnel with fault diagnosis tasks. VLSED was trying to deliver an application to the Navy's Fleet Technical Service Centers (FTSCs) that would allow experts to remotely diagnose and solve (at least some) faults for MK 41 Vertical Launching Systems, which were used on-board some AEGIS cruisers. If this worked, remote problem solving could reduce the number of costly flights required for FTSC personnel to perform at sea, which would also save time and perhaps reducing FTSC manpower needs.

VLSED developed their application using Inference Corporation's CBR product line, then known as CBR2. This choice was well motivated; CBR2 was a highly successful commercial tool for solving interactive diagnosis tasks, arguably because its basis is CCBR. Although targeted for supporting help-desk personnel, CBR2's application to weapons systems diagnosis seemed straightforward.

Unfortunately, although several successful CCBR applications have been deployed, VLSED and some other organizations found that creating a successful CCBR application requires expertise in case library authoring. VLSED's prototype application was moderately successful, but its library was not designed using standard guidelines for creating CCBR applications, and it required substantial effort to enhance and maintain. Accordingly, when VLSED was awarded additional funds to pursue this approach, they were required to upgrade the FTSC application rather than take the next step–place it in the fleet. The FTSC application was enhanced via a long-term consulting contract, but was later abandoned because there was no funding nor Navy infrastructure to ensure its maintenance. The VLSED group responsible for creating this application dispersed and valuable Navy knowledge on creating CCBR applications was lost.

Thus, commercial CBR tools are confronted with research challenges that affect their usefulness and acceptance by corporate and government clients. Three challenges that we have focused on in our research are:

1. *Case authoring*: As illustrated in the above anecdote, the task of creating a CCBR case library is a knowledge engineering task that requires substantial expertise.
2. *Dialog inferencing*: The quality of the human-computer dialog in CCBR applications suffers from the lack intelligent methods for dynamically computing inferences from user input.
3. *Expanded applicability*: CCBR tools were limited to case retrieval, and were not applicable to more elaborate decision support tasks (i.e., of interest to the Navy).

We addressed these challenges by creating our own CCBR tool, NaCoDAE, to serve as a testbed (Breslow & Aha, 1997a).[1] First, we developed a machine learning approach for simplifying the case authoring task by enforcing some authoring guidelines (Aha & Breslow, 1997). Second, we integrated NaCoDAE with a model-based reasoning component and a query retrieval tool to automatically answer some questions during a conversation, thereby reducing interactive elicitation needs (Aha et al., 1998). Finally, we extended NaCoDAE to address knowledge-intensive planning tasks, which required an integration with a hierarchical task editor; the resulting system is named HICAP (Muñoz-Avila et al., 1999).

The following sections detail our progress. We begin by defining CCBR (Section 2), and then detail each approach and its evaluation (Sections 3 to 5). We summarize related research and outline suggestions for future research efforts in Section 6.

2. Conversational Case-Based Reasoning

This section introduces CCBR (Section 2.1) and our NaCoDAE implementation (Section 2.2).

2.1. Introduction

In most CBR research systems, the user is expected to input their entire problem description (query) at the outset; this requires the user to determine the problem-solving relevance of each feature and to have detailed domain knowledge, which users often lack in practice. In contrast, CCBR systems require the user to initially input only a brief free-text description of their problem. The system then supports interactive problem assessment to construct a query (i.e., a problem specification). During this conversation, the system progressively ranks and displays the top-matching cases' solutions (and the displayed questions). Thus, the user need only answer posed questions; a priori knowledge concerning their relevance is not needed.

CCBR was pioneered by Inference Corporation in their CBR product line, now known as K-COMMERCE. These types of tools have grabbed a large share of the customer support tool market niche. Their popularity stems, in part, from their ability to incrementally and interactively acquire queries describing customer problems while imposing few restrictions on the query's information content and internal sequencing.

System users (i.e., usually[2] call center personnel) need only guide customers through a dynamically determined set of questions, but do not need extensive domain expertise. This approach allows potential solutions, stored in cases whose problem descriptions are highly similar to the user's query, to be available at any time during a conversation.

Prior to problem solving, a *case author* creates a set of cases, called a *case library*, for the CCBR system. In its most generic form, a *case* C in a CCBR system is represented as follows:

1. *Problem* $C_p = C_d + C_{qa}$: Encodes the problem solved by C_s.
   (a) *Description* C_d: Free text that partially describes C's problem.
   (b) *Specification* C_{qa}: A set of $\langle$question,answer$\rangle$ pairs.
2. *Solution* $C_s = \{C_{a1}, C_{a2}, \ldots\}$: A sequence of actions C_{ai} for responding to C_p.

Actions can be free text, hyperlinks, or other objects. A case's problem description and specification serve as its index. Questions in case specifications can be internally disjunctive (i.e., have multiple answers). Cases are "positive" examples: applying C_s to C_p is assumed to be successful. C serves as a prototype for solving queries whose problem specifications are similar to C's; queries that closely match a case's problem are expected, with high probability, to benefit from its solution. An example case for a printer troubleshooting application might include questions referring to the printer's display panel or the status of the paper tray, while an action might be to fill the tray, clear a jam, or to phone technical support.

Users interact with CCBR systems by submitting a *query* Q in a *conversation*, which begins when a user inputs a text description Q_d of a problem. The system then computes the similarity $\mathrm{s}(Q, C)$ of Q_d to the problem description C_d of each stored case C, which yields an initial case similarity ranking. The solutions of these top-ranking cases are displayed according to decreasing similarity in a solution display D_s. Currently unanswered questions in these cases are ranked by importance (detailed in Section 2.2), and the top-ranking questions are listed in a second ranked display D_q. The user can then select a question $q \in D_q$ to answer or a solution $s \in D_s$, thereby terminating the conversation. If the user selects a question, they then input its answer a. The CCBR system adds $\langle q, a\rangle$ to the query's problem specification Q_{qa}, recomputes all case similarities, and updates the displays D_s and D_q. As more questions are answered, the similarity computations and resulting case rankings should become more accurate. Users can delete or alter their previous answers to questions at any time. Problem solving terminates successfully when the user is satisfied with a selected solution or the top-ranking case's similarity exceeds a threshold. It terminates unsuccessfully when the system cannot find a good match or further relevant questions to ask. Figure 1 visually summarizes this process.

Figure 2 summarizes the generic CCBR algorithm for conducting a conversation. This description does not define how case similarity scores are computed, how questions are ranked, or how input text is processed. These and other details vary among CCBR systems. We describe our definitions for them in the context of NaCoDAE in Section 2.2.

2.2. NaCoDAE

We originally developed NaCoDAE (Breslow & Aha, 1997a) to test our strategy for simplifying case authoring (Aha & Breslow, 1997). More recently, we evaluated its extensions for dialogue inferencing (Aha et al., 1998) and conversational case-based planning (Muñoz-Avila et al., 1999). Sections 3 through 5 detail these studies.

NaCoDAE (Navy Conversational Decision Aids Environment) embodies the generic CCBR tool description of Figure 1 and incorporates some simple additional design decisions.

Text Processing: Case problem descriptions and user input text are canonicalized. Brill's (1995) part-of-speech tagger is used to identify noun phrases and stem the cases' descriptions. Synonyms and stopwords are input by the case author. Given query text Q_d, NaCoDAE computes the similarity score for all stored cases, where $\mathrm{s}(Q, C)$ is defined as the percentage of roots and noun phrases in C_d that are also in Q_d. NaCoDAE uses

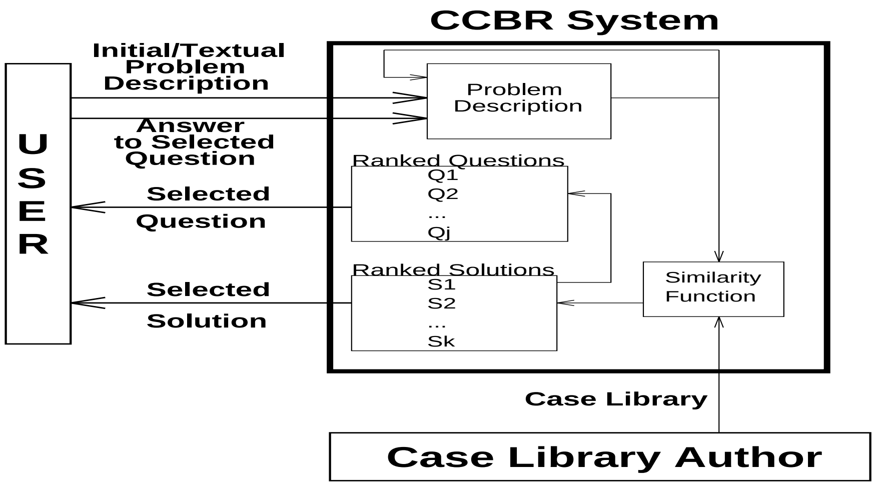


Fig. 1. The generic CCBR problem solving process

this score to identify and rank the most similar cases, whose solutions are displayed in D_s.

Question Ranking: Questions in D_q are ranked according to their frequency in specifications of the cases whose solutions are displayed in D_s. We also experimented with ranking strategies based on information gain and their ordering within a case (in addition to frequency), but found that frequency is sufficient for our needs.

```
Inputs: $L$: Case library
        $k$: Size of $D_s$
        $n$: Size of $D_q$
        $s()$: Similarity (case ranking) function
        $r()$: Question ranking function

Conversational_case_retriever($L, k, n, s(), r()$) =
  $Q_d$ = user_text();
  $Q_{qa} = Q_s = \emptyset$;
  REPEAT
    $D_s$ = rank_cases($Q, L, s(), k$);
    $D_q$ = rank_questions($D_s, L, r(), n$);
    IF (user_inputs_query_description())
    THEN $Q_d$ = user_text();
    ELSE IF ($q_i$ = user_selects_question($D_q$))
      THEN $Q_{qa} = Q_{qa} \bigcup \{q_i, \text{user\_answer}(q_i)\}$;
  UNTIL ($Q_s$ = user_selects_solution($D_s$) OR
      unsuccessful_stopping_criteria())
  RETURN $Q_s$
```

Fig. 2. Generic algorithm for a CCBR conversation

Case Ranking: After the user selects and answers some questions from D_q, yielding a partial query specification Q_{qa}, NaCoDAE scores s(Q, C) for each case C using a simple function:

$$\text{score}(Q, C) = \frac{\text{same}(Q_{qa}, C_{qa}) - \text{diff}(Q_{qa}, C_{qa})}{|C_{qa}|}, \tag{1}$$

where same(Q_{qa}, C_{qa}) (diff(Q_{qa}, C_{qa})) is their number of shared (conflicting) $\langle q, a\rangle$ pairs. An optional procedure can be used to control for a bias towards cases containing fewer questions (Aha & Breslow, 1998).

3. Simplifying Case Authoring

CCBR cases are typically *heterogeneous*: there is often little overlap in the sets of questions used to define each case's problem specification. This flexibility allows cases to be small in size and retrieved by answering few questions, but complicates the case engineering task; heterogeneity complicates the problem of deciding which questions and cases to present to the user at each point during a conversation. Poor choices for questions will prevent useful further diagnosis of the customer's problem, while poor choices for cases will prevent a good solution from being retrieved. In this section we describe a standard approach to this *case authoring* task, explain why it is problematic, in-

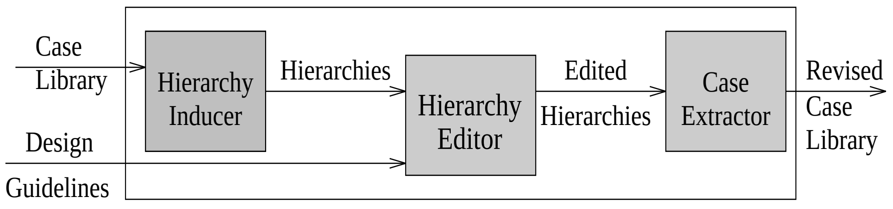


Fig. 3. The case library revision process

troduce our novel tree induction technique for revising case libraries (implemented in CLIRE), and empirically evaluate whether/why it can improve the performance (i.e., precision and efficiency) of conversational case libraries.

3.1. Design guidelines

Case authoring is the art of designing good libraries. Inference (1995) lists 46 guidelines for building CCBR libraries, including:

1. reuse questions when possible,
2. order *context* before *detail* questions,
3. eliminate questions that do not distinguish cases,
4. ask for only one thing in a question, and
5. use a similar, small number of questions in cases.

When analyzed separately, each guideline appears sensible, and contributes to good CCBR performance. For example, Guideline 1 encourages similar cases to be distinguished by a *shared* question. Guideline 2 encourages case authors to group cases into topics distinguishable by a few key *context* questions, which can quickly isolate a potentially relevant case library subset without eliminating relevant cases. Eliminating unnecessary questions from cases (Guideline 3) can increase *conversational efficiency* (i.e., reduce the number of questions that must be asked of users before a good solution can be retrieved) and, perhaps, retrieval precision (Wettschereck et al., 1997).

Although these guidelines are individually reasonable, their use is problematic for two reasons. First, they can conflict, and it is unclear to most novices how to resolve these conflicts (e.g., while reusing questions is important (Guideline 1), cases should not be described by many unnecessary questions (Guideline 5), since they tend to reduce retrieval efficiency and precision). Second, because they are large in number, mastering them requires a long learning curve. These problems are exacerbated when case libraries are large, when novice users are allowed to edit cases, or when several users jointly perform library maintenance.

Novice users can consult costly experts for help. We instead advocate using software tools to assist novices with the case authoring process.

3.2. Revising Conversational Case Libraries

Figure 3 summarizes the three phases of our approach, which we have implemented in CLIRE. The first phase creates a tree-structured representation of the library, which simplifies library assessment and revision. Revisions are performed in the second phase, according to design guidelines. The final phase extracts the revised cases from this representation. This revision process is transparent to users; they interact with the revised library only, and not with its intermediate representation.

CLIRE uses a simple top-down decision tree induction algorithm (TDIDT) (e.g., Quinlan, 1986; Breslow & Aha, 1997b) in the first phase. TDIDT algorithms recursively apply a selection criterion to choose an index question to partition a node's cases. CLIRE attempts to generate a separate leaf per case.

Traditionally, most TDIDT selection criteria assume that cases are *homogeneous* (i.e., defined by the same set of questions), although several methods exist for tolerating missing values (Quinlan, 1989). They also assume that cases have been clustered (e.g., by class). These assumptions are violated in our context, where cases are *heteroge-*

```
Key: $L$: Set of cases in the case library
     Actives: Set of cases to be distinguished from each other
     Inactives: Set of cases to be distinguished from Actives
     $Q$: Questions used in ancestors of this node (initially $\emptyset$)
     $N$: A node ($q$ is its selected question)

Top level call: induce_tree($L$,$\emptyset$,$\emptyset$)
induce_tree(Actives,Inactives,$Q$) =
1. IF stop_splitting(Actives,Inactives,$Q$)
2. THEN RETURN make_leaf(Actives $\cup$ Inactives)
3. ELSE $q$ = select_question($Q$,Actives)     // $q \notin Q$
4.      FOREACH $a \in$ answers($q$)
5.           Actives$_a$ = $\{C|C \in \texttt{Actives}, \langle q,a\rangle \in C_{qa}\}$
6.           IF Actives$_a \neq \emptyset$
7.           THEN Inactives$_a = \{C|C \in \texttt{Inactives}, \langle q,a\rangle \in C_{qa}\}$
8.                $N_a$ = induce_tree(Actives$_a$,Inactives$_a$, $Q \cup \{q\}$)
9.      Actives$_?$ = $\{C|C \in \texttt{Actives}, \{q,*\} \notin C_{qa}\}$      // ?  = ``unknown''
10.     IF Actives$_? = \emptyset$
11.     THEN Inactives$_?$ = $\{C|C \in \texttt{Inactives}, \{q,*\} \notin C_{qa}\}$
12.     ELSE Inactives$_?$ = Actives $\cup$ Inactives $-$ Actives$_?$
13.     IF (Actives$_? \neq \emptyset$ $\vee$ Inactives$_? \neq \emptyset$)
14.     THEN $N_?$ = induce_tree(Actives$_?$,Inactives$_?$,$Q \cup \{q\}$)
15.     RETURN $N$
```

Fig. 4. CLIRE's pseudocode for tree induction

neous (i.e., a CCBR library typically does not have any one question answered in all of its cases) and cases are not labeled by class (i.e., each case's solution, or action sequence, can be unique). Therefore, we used an alternative selection criterion that chooses the most frequently answered question among a node's cases. Cases that do not contain an answer for the selected question are grouped into a separate node and recursively partitioned.

Figure 4 summarizes CLIRE's tree induction algorithm. It inputs two sets of cases, *Actives* and *Inactives*, and a list Q of used questions. If none of the questions answered in Actives can distinguish them from each other or from the Inactives, then a leaf is returned with these cases. Otherwise (steps 3–8), a question $q \notin Q$ is selected for recursive partitioning, and a subtree is generated for each answer a of q that appears in at least one of the Actives. The recursive call constrains the Actives and Inactives to cases C where $\langle q,a\rangle \in C_{qa}$, adds q to Q, and notes that q was used to partition N. Steps 9–14 generate a subtree for N containing its cases C where q is not answered in C_{qa}. If some Actives have no answer for q, then Inactives$_?$ is set to the cases at N that still must be distinguished from Actives$_?$ (i.e., because the ⟨q,a⟩ pairs on the path to Actives$_?$ are a proper subset of the pairs to Inactives$_?$). Whenever possible, this algorithm recurses until the active and inactive cases are distinguishable.

After creating the indexing tree (phase 1), cases are edited (phase 2) using case-specific feature selection, which removes, from a case C, all $\langle q,a\rangle$ pairs from C_{qa} that do not appear on any path from the root to leaves containing C. We're assuming that the deleted ⟨q,a⟩ pairs are irrelevant because they do not logically distinguish their case. Finally, during case extraction (phase 3), CLIRE records, in order, the ⟨q,a⟩ pairs that appear on paths to each case C, thus reordering the pairs in C_{qa} (i.e., C's problem specification).

This implementation of CLIRE addresses the first three guidelines listed in Section 3.1: reuse questions, order context before detail questions, and eliminate non-distinguishing questions. Ex-

Table 1. Case libraries used in the experiments

			Original		After Clire Revision	
Name	#Cases	#Actions	#Questions	#Answers	#Questions	#Answers
Printing	25	28	27	70	16	55
VLS	114	227	597	710	83	395
ACDEV	3334	1670	2011	28200	1266	26827

tracting cases from a tree reuses answered questions on overlapping multiple paths, thus encouraging question reuse. Frequently answered questions are assumed to be context questions, and are identified as those on the higher nodes of paths. This question ordering is preserved in the revised cases. Non-distinguishing questions are removed during the case-editing phase.

3.3. Empirical Evaluation

3.3.1. Performance measures. We selected two measures to evaluate the performance of a CCBR library for solving a set of queries. The first is *precision*, defined as whether the retrieved case's actions solve a user's query. The second is *efficiency*, defined as the number of questions asked before retrieval occurs (i.e., lower values correspond to higher efficiency). Good CCBR libraries permit both high precision and efficiency.

3.3.2. Methodology.

The best way to run experiments with NaCoDAE is with human subjects. Unfortunately, sufficient numbers of subjects were unavailable for our evaluation. Therefore, we introduced a simple methodology that simulates a human user using a variant of leave-one-out cross validation that we call *leave-one-in* (LOICV). LOICV cycles through a library's cases, each time selecting a *target* case C, with replacement. The query is initially empty. During conversations, LOICV ignores questions in display D_q that are not answered in C_{qa}. Retrieval success occurs when the retrieved case's solution is identical to C_s. In each iteration of a conversation, LOICV either selects the top-ranking question in D_q that is also in C_{qa} (with probability p_q) or terminates the conversation by selecting the top-ranking case in D_s. (This case is also selected when no question in C_{qa} is in D_q, or if its similarity score exceeds a threshold.) Because ranking ties in these lists are randomly broken, all reported results are averages from ten runs.

3.3.3. Libraries We experimented with the three case libraries summarized in Table 1. *Printing*, a simple example library provided with Inference's products, is used to diagnose and recommend solutions for printer failures. VLS, obtained from the VLSED, provides technical assistance for maintaining a vertical missile launch system. ACDEV, from Circuit City's Answer City product, was designed to support branch store personnel. The first library is fairly well designed, while the latter two are known to be problematic. ACDEV's size prevented us from using all cases as queries. Instead, we randomly selected 100 cases for querying according to a uniform distribution with replacement.

Table 1 shows that Clire reduced the total number of questions by between 37% and 86%, and the total number of answers in cases by between 5% and 44%. It is plausible that this should increase NaCoDAE's retrieval efficiency on these libraries. However, it is not clear whether Clire simultaneously sacrifices precision.

3.3.4. Experiments Parameter k (n) is the size of the display D_s (D_q). In our initial experiment, we fixed $k = 4$ and varied n. Figure 5 summarizes the results for these three case libraries.[3] As shown, Clire slightly increased precision and efficiency for these libraries across the conditions tested. Similar relative results also occurred when we varied k.

However, modifying p_q, the probability that LOICV continues to ask unanswered questions from the target case instead of selecting a case, does affect relative performance. We observed this by setting $k = 4$, $n =$ unlimited (i.e., all unanswered questions among the the top-ranking k cases were included in D_q), and varying p_q in $\{70\%, 80\%, 90\%, 100\%\}$. For VLS and ACDEV, performance benefits with the Clire-revised libraries did not occur for smaller settings of p_q. Thus, Clire's benefits accrue primarily when conversations are not terminated prematurely.

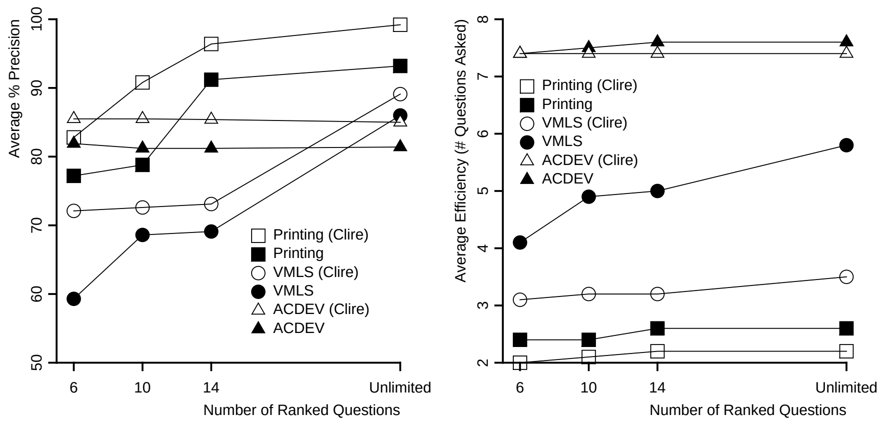


Fig. 5. CLIRE improves CCBR performance in a baseline study

Although our initial experiments establish that revision by CLIRE can improve CCBR performance for these libraries, it is not clear why. Therefore, we performed an ablation study with variants of CLIRE that (1) applied only case-specific question selection, (2) applied only question reordering (within each case), (3) did both, or (4) did neither. Thus, this study isolates the effects of CLIRE's two case-editing modifications. Question selection is eliminated by reinstating the "deleted" ⟨q,a⟩ pairs of each case C, placing them after the CLIRE-selected pairs in C_{qa}'s ordering. Question re-ordering is eliminated by retaining each revised case's original question ordering, but without deleted questions.

Table 2 summarizes some typical results we found in this ablation study. Invariably, CLIRE's power emanated primarily from its question selection capability. Using only question selection yields behavior similar to using both revision operators, while using only question reordering yields smaller gains, and sometimes even reduces performance. This occurred independently of NaCoDAE's parameter settings and the question-ranking strategy used. (However, ordering questions remains a good case authoring guideline; a consistent ordering simplifies locating similar cases and encourages question reuse.)

3.4. Discussion

Designing high-performance CCBR libraries will interest any organization who wants to deploy this technology. Although commercial vendors supply guidelines for designing cases to ensure good CCBR performance, they are difficult to implement for complex libraries. Software assistants for case authoring can potentially meet this challenge.

Three topics are of particular interest for future research. First, our evaluation should be more realistic. Question ordering in cases should affect question selection in conversations, and LOICV should be permitted to select questions not answered in the target case and to answer questions incorrectly (i.e., simulating noise). This requires domain models for noise and for answering questions not in C_{qa}. Second, CLIRE should exploit domain knowledge. For example, although it deletes questions from cases because they are not selected in the tree, these may be important questions. More generally, we assume domain experts are unavailable, although CLIRE could be modified to incorporate domain-specific information. Finally, CLIRE's approach is post-hoc; the case library must first exist before it can be edited. This approach could be modified so that case authoring guidelines are enforced as cases are created.

4. Enhancing Dialogue Inferencing

Even when case libraries are well-designed, CCBR conversations can be inefficient because the user may be prompted with questions that could be automatically answered (e.g., through inferencing). Therefore, CCBR tools should automatically infer problem description details (i.e., answers to questions) from the user's inputs whenever possible during a conversation.

Table 2. CLIRE ablation study results ($k = 4, n = 6, p_q = 100$)

Revision Operations	Libraries					
	Printing		VMLS		ACDEV	
	Precision	Efficiency	Precision	Efficiency	Precision	Efficiency
Neither	78.8%	2.4	59.3%	4.1	80.5%	7.6
Question Reordering Only	75.6%	2.4	68.7%	5.0	83.7%	7.6
Question Selection Only	82.8%	2.0	72.5%	3.2	85.8%	7.4
Both	82.8%	1.8	72.1%	3.1	85.8%	7.4

User's Initial Problem Description: I'm getting black streaks on my paper

Display of Top-Ranked Questions in Response to this Description:

1. Q21: What does the print quality look like?
2. Q24: Are you having print quality problems?
3. Q18: Are you printing on the correct side of the paper?
4. Q25: What is the display message?

Fig. 6. Example of the dialogue inferencing problem during a CCBR conversation

Figure 6 demonstrates an example of this problem for the Printing case library. Although the first two displayed questions were implicitly answered in the query's problem description Q_d, NaCoDAE cannot automatically infer these answers (i.e., "Black Streaks" and "Yes," respectively). Case retrieval efficiency can be increased by automatically deriving these answers.

Some commercial CCBR tools employ rule-based reasoning to automatically derive inferences. For example, suppose that, for the printer troubleshooting task, the user enters the description "black streaks on paper" and the system has the following rule:

IF text includes "black streaks" and "paper",
THEN assign "Yes" as the answer to Q24.

Then the second question in Figure 6 could automatically be answered "Yes." However, this solution to the *dialogue inferencing* task requires the case author to provide a complete and correct set of independent inferencing rules that (1) relate text to all possible $\langle q, a \rangle$ pairs that it implies (*text implication rules*) and (2) relate $\langle q, a \rangle$ pairs inferentially to one another (*chaining implication rules*). Also, existing tools do not guarantee rule correctness or domain completeness, and significant knowledge engineering challenges ensue:

1. *Input Size*: Rule sets are often large (e.g., Printing, which contains only 25 cases and 27 questions, yields over 100 rules). Manually eliciting them is tedious and error-prone.
2. *Comprehensibility*: Large sets of unrelated rules can be difficult to examine.
3. *Maintenance*: Maintaining large rule sets can be difficult, if not impossible, for case libraries that require updating.

Therefore, some CCBR case authors avoid using rules, either because an incomplete rule set will decrease case retrieval performance for their applications, or because the maintenance issues are daunting.

We devised a dialogue inferencing approach that uses model-based reasoning to generate implication rules. To search these rules, we integrated NaCoDAE with Parka-DB (Hendler et al., 1996), a fast query-retrieval tool. Our integration, summarized in Figure 7, automatically infers answers implicit in partially-specified queries. This section details our approach and its evaluation.

4.1. Model-Based Dialogue Inferencing

Our approach interactively elicits a case library's *object* model (relating domain objects) and *question* model (relating questions to these objects) from the case author. Because these models, represented as semantic networks, are usually more compact than the corresponding rule set, they

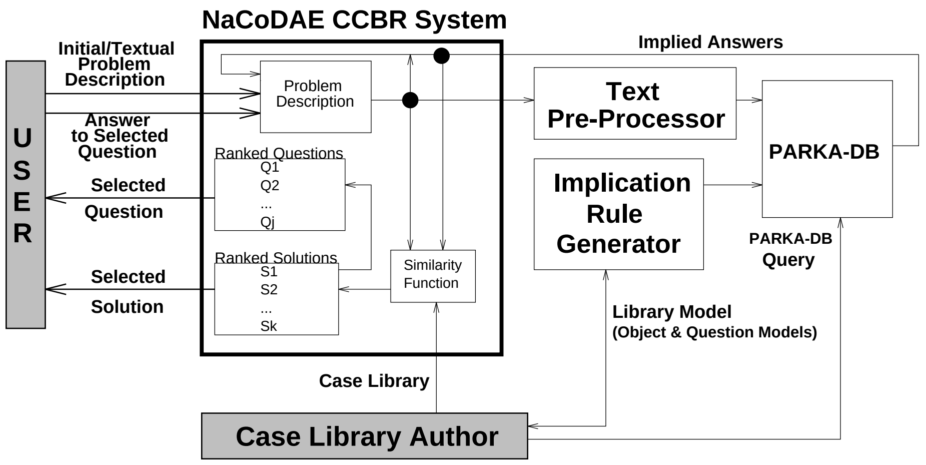


Fig. 7. Model-based support for dialogue inferencing in CCBR conversations

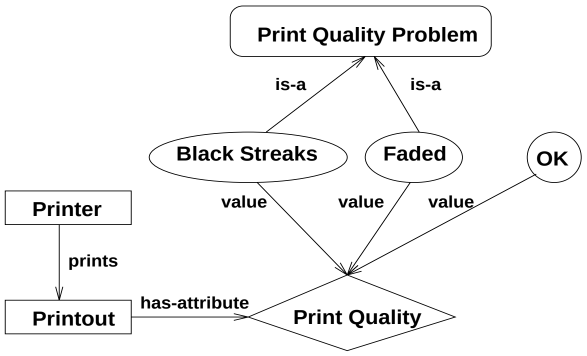


Fig. 8. Partial object model for the printer troubleshooting library

will usually increase comprehensibility and simplify maintenance. Given these models, the *implication rule generator* derives a rule set. Given this set and the query, Parka-DB will retrieve all answers implied by the user's text and/or previously answered questions, and add them to Q_{qa}.

Library models will be created by the library author using an interactive editor; this is the only module that we have not yet implemented. Parsing techniques will assist in identifying objects and their relationships. For example, when the user enters the question *Can your printer print a self test?*, "printer" and "self test" will be identified as (possibly previously identified) objects connected by the relationship "print." Users will be queried to confirm all tentative identifications.

Figure 8 shows part of the object model for Printing. It represents a printer, its printout, and possible print qualities, with boxes denoting objects, diamonds denoting domain attributes, and ellipses denoting attribute values and value categories.

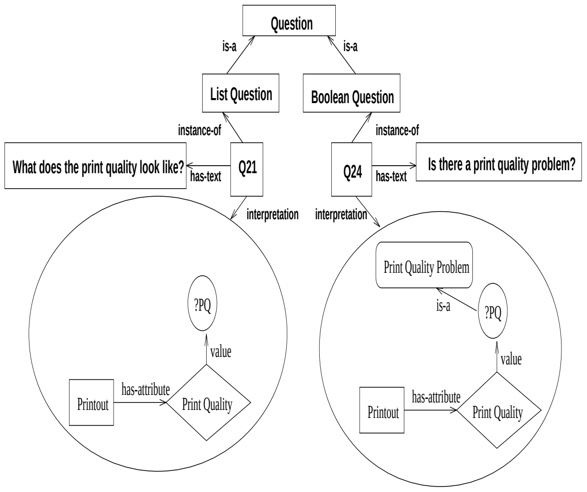


Fig. 9. Partial question model for the printer troubleshooting library

Figure 9 displays a partial question model that relates two questions to the object model. Each question's *interpretation* is used to determine how to derive an answer. Interpretations for boolean questions (e.g., Q24) test for subgraph existence using existentially quantified variables. Interpretations for list questions (e.g., Q21) instead search for specific variable bindings during subgraph matching.

Textual inferences depend on triggering phrases associated with nodes in the question model. Syn-

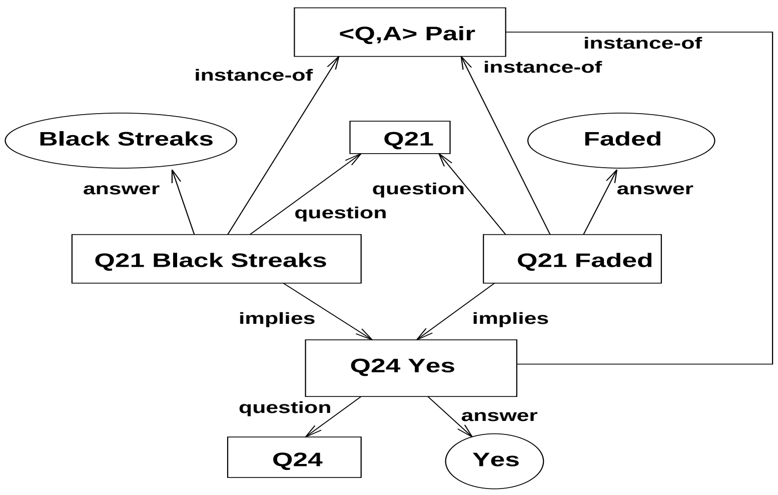


Fig. 10. Two chaining implication rules for the printer troubleshooting library

```
{ (text_implies ?User_phrase ?QA),        ;; This phrase implies ?QA if:
  (Triggering_text ?QA ?Phrase),          ;; Locate triggering phrase.
  (stringMatch ?User_phrase ?Phrase),     ;; Ensure these phrases match.
  (QA_question ?QA ?Q),                   ;; Fetch the question.
  (answer ?Q "unknown"),                  ;; If it's not yet answered,
  (QA_answer ?QA ?A)                    } ;; then retrieve implied answer.
```

Fig. 11. Parka-DB query for retrieving implied answers from text rules

onym lists maintained by the system allow less literal matches to be made.

Deriving chaining rules requires relating the interpretations of two questions in the question model. For example, given an answer to Q21 ("What does the print quality look like?") we can derive an answer for Q24 ("Is there a print quality problem?"). Relating these interpretations requires matching subgraphs, where unbound variables of the target subgraph are bound in the source graph. The two chaining rules that relate the interpretations shown in Figure 9 are shown in Figure 10. The first rule states that ⟨Q21,"Black Streaks"⟩ implies ⟨Q24,"Yes"⟩. The second rule is similar, but for "Faded" print quality.

Parka-DB infers chaining rules from the model and generates text rules from the wording of the questions in the case library, where all knowledge is represented as binary assertions. For example, the text rule "If Q_d contains 'some black streaks' and Q21 is not yet answered, then answer Q21 with 'Black Streaks'" could be represented as:

```
(Triggering_text Q21_Black_Streaks
                    "some black streaks")
(QA_question Q21_Black_Streaks Q21)
(Answer Q21 "unknown")
(QA_answer Q21_Black_Streaks "Black Streaks")
```

Suppose Q_d contains this phrase and Parka-DB is given these assertions along with the query shown in Figure 11. Then the following bindings could be found for (text_implies "some black streaks" ?QA) to derive the answer "Black Streaks" for Q21:

```
{ ?User_phrase/"some black streaks",
  ?QA/Q21_Black_Streaks,
  ?Phrase/"some black streaks",
  ?Q/Q21,?A/"Black Streaks"}.
```

The assertions corresponding to the two chaining rules shown in Figure 10 are:

```
(QA_question Q21_Black_Streaks Q21)
```

```
{   (chaining_implies ?QAx ?QAy),      ;; ?QAx implies ?QAy if:
    (QA_question ?QAx ?Q1),            ;; Find qn of first ⟨q, a⟩ pair.
    (QA_answer ?QAx ?A1),              ;; Find answer of this pair.
    (answer ?Q1 ?A1),                  ;; This is its current answer?
    (QA_question ?QAy ?Q2),            ;; Yes; find qn of second pair.
    (answer ?Q2 "unknown"),            ;; If it's not answered,
    (QA_answer ?QAy ?A2)           }   ;; then retrieve implied answer.
```

Fig. 12. Query for retrieving implied answers from chaining implication rules

(QA_answer Q21_Black_Streaks "Black Streaks")
(QA_question Q24_Yes Q24)
(QA_answer Q24_Yes "Yes")
(chaining_implies Q21_Black_Streaks Q24_Yes)
(QA_question Q21_Faded Q21)
(QA_answer Q21_Faded "Faded")
(chaining_implies Q21_Faded Q24_Yes)

If the problem description currently contains only one answered question, namely (answer Q21 "Black Streaks"), then these rules can be queried as shown in Figure 12, and Parka-DB will find the following bindings:

{ ?QAx/Q21_Black_Streaks,
?Q1/Q21, ?A1/"Black Streaks",
?QAy/Q24_Yes, ?Q2/Q24, ?A2/"Yes"}.

In sum, if the user inputs "some black streaks", the system will infer Q21_Black_Streaks by text inferencing and Q24_Yes using a chaining rule. Additional chaining inferences may then be derived from these inferences or question answers provided by the user later in the conversation.

4.2. Evaluation

We hypothesized that dialogue inferencing should increase NaCoDAE's retrieval efficiency without impacting its retrieval precision. We tested this hypothesis on Printing, after constructing a model of it from which 63 text and 43 chaining rules were extracted.

We conducted an ablation study to identify whether these two types of rules can increase conversational retrieval efficiency. In our LOICV experiments we fixed $s = 4$ and $q = 6$; other values for the display sizes gave similar results. The ablations for the set of inferencing rules used were none, text (only), chaining (only), or both. Text rules were applied to the text description, influencing the initial case ranking. Chaining rules, whenever used, were applied immediately after questions were answered and were always recursively applied until no new answers were derived.

Table 3 summarizes results for Printing, including the average number of text and chaining rule inferences. Both types of rules increase efficiency, and their effects were synergistic (i.e., their combination yields a 13.7% to 20.9% reduction in the number of questions answered by the simulated user). Retrieval precision was unaffected.

Printing is not a good library for demonstrating the efficiency gains of our dialogue inferencing approach because its problem specifications are small (i.e., only 2.84 ⟨q,a⟩ pairs on average) and few of its ⟨q,a⟩ pairs can be inferred from others. Therefore, we tested NaCoDAE on ACDEV, whose characteristics are better suited for this demonstration, and from which we could develop a reasonable case library model.

ACDEV's cases can be clustered according to troubleshooting topic (e.g., problems with TV remotes), but not their solutions. We selected three clusters, totaling 20 cases, 33 questions, and 19 actions. These cases are less sparse (26.1%) than the printer library (10.5%), the average size of their specifications is much higher (8.6 vs 2.84), and more of the ⟨q,a⟩ pairs in them are logically related (e.g., if *What is the general nature of the video problem?* has any answer, then *What is the general nature of the problem?* has answer "Video"). We extracted a model containing 121 text and 105 chaining rules.

ACDEV's results are shown in Table 4. The increase in efficiency was 30.4%, substantially higher than for Printing. For some cases, the limited

Table 3. Dialogue inferencing ablation study results (precision, efficiency, and number of inference) for the printing case library

Rule Sets Used	**Prec.**	**Eff.**	Num Inferences	
			Text	**Chaining**
None	95%	2.78	–	–
Text	96%	2.48	0.29	–
Chaining	94%	2.40	–	0.35
Both	96%	**2.20**	0.27	0.36

Table 4. Dialogue inferencing ablation study results (precision, efficiency, and number of inferences) for the subset of ACDEV

Rule Sets Used	**Prec.**	**Eff.**	Num Inferences	
			Text	**Chaining**
None	86%	7.72	–	–
Text	91%	7.46	0.57	–
Chaining	87%	5.58	–	1.94
Both	88%	**5.37**	0.68	2.05

question display size (i.e., q) reduced retrieval precision when inferencing was not used. For these cases, dialogue inferencing succeeded in retrieving a correct case because answers to questions that were not displayed were automatically inferred.

We also tested our integrated approach on a subset of a third case library, DC220, obtained from Xerox Corporation. We again selected one cluster of 20 cases (on printer copy quality problems), whose problem specifications contain an average of 5.6 $\langle q, a \rangle$ pairs and whose cases have only 12 distinct solutions. We generated 70 text and 113 chaining rules from this subset. The results were similar to our previous results: dialogue inferencing increased efficiency by 32.9%, reducing the average number of questions answered from 5.56 to 3.73 per conversation, while precision remained 100%. However, text inferencing was ineffective because the case's problem descriptions, which should mimic a user's text, had few words in common with the questions' text.

4.3. Discussion

We hypothesize that, in the context of our model-based approach for dialogue inferencing, text inferences should be beneficial when case descriptions overlap with question text. Also, chaining rules should benefit case libraries whose case specifications are long and contain logically or causally related ⟨q,a⟩ pairs. Importantly, rule completeness and accuracy should impact retrieval performance. For example, when we randomly deleted half of ACDEV's chaining rules, efficiency was reduced (i.e., 0.5 more questions were answered, on average, by the "user"), and when we added noise to half of ACDEV's chaining rules, retrieval precision dropped by 15%.

5. Conversational Case-Based Planning

Although CCBR systems have been successfully used in case retrieval applications, they have not been applied to synthesis tasks (e.g., planning, design). Yet CCBR should prove useful for decomposable synthesis tasks that require intensive user interaction. In particular, CCBR could be used to iteratively elaborate an initial solution, which would allow users to tailor a solution to their needs and, thus, enhance their confidence in the resulting solution. We developed HICAP, an integrated extension of NaCoDAE for use in planning tasks. This section describes this integration and its application to a complex military task.

5.1. Plan Authoring Task

HICAP (Hierarchical Interactive Case-based Architecture for Planning) is a general-purpose plan authoring tool that we are applying to support noncombatant evacuation operations (NEOs). NEOs are military operations, directed by the USA Department of State, for evacuating noncombatants, nonessential military personnel, and selected host-nation citizens and third country nationals whose lives are in danger to an appropriate safe haven. They usually involve a swift insertion of a force, temporary occupation of an objective (e.g., a USA Embassy), and a planned withdrawal after mission completion. NEOs are usually planned and conducted by a joint task force (JTF), and are under an Ambassador's authority. Force sizes can range into the hundreds with all branches of armed services involved, while the evacuees can number into the thousands. At least ten NEOs were conducted within the past decade

(Siegel, 1995). Unclassified publications describe NEO doctrine (DoD, 1994), case studies (Siegel, 1991; 1995), and more general analyses (Stahl, 1992; Lambert, 1992).[4]

Formulating a NEO plan is complex because it requires considering a wide range of factors (e.g., military resources, meteorological predictions), uncertainties (e.g., hostility levels and locations), and hundreds of subtasks (e.g., evacuee processing). NEOs are challenging to plan, and flawed plans could be disastrous. For example, Siegel (1991) reported that evacuees in Operation Eastern Exit were not inspected prior to transport, and one of the evacuees produced his weapon during a helicopter evacuation flight. Although it was immediately confiscated, this oversight could have been tragic.

NEO are planned with the help of published military doctrine, which provides a framework for designing strategic and operational plans (DoD, 1994). However, doctrine cannot address most tactical issues, which are operation-specific. Thus, the JTF commander (CJTF) must always adapt doctrine to the specific needs of a NEO in two ways. First, the CJTF must modify doctrine by eliminating irrelevant planning tasks and adding others (e.g., depending on resource availabilities). Second, the CJTF must employ experiences from previous NEOs, which complement doctrine by suggesting tactical refinements that are suitable for the current operation. For example, past experiences could help identify whether evacuees for a specific operation should be concentrated at an embassy or grouped at multiple evacuation sites.

After analyzing NEO doctrine, reviewing case studies, and consulting with NEO experts, we concluded that a NEO plan authoring assistant must have (at least) the following capabilities:

- *Doctrine-driven*: Use a doctrine task analysis to guide plan formulation.
- *Interactive*: Support interactive plan editing.
- *Provide Case Access*: Index plan segments from previous NEOs, and retrieve them for users if warranted by the current operational environment.
- *Perform Bookkeeping*: Record and maintain information on tasks, their completion status, and relations between task responsibilities and joint task force (JTF) elements.

Although incomplete, this list provides a useful initial specification for HICAP. Although several other systems have been proposed for NEO planning, the only deployed tool is limited because it cannot reason from previous experiences.

5.2. *HICAP: An Interactive Case-Based Planner*

HICAP integrates a task decomposition editor, HTE (Muñoz-Avila et al., 1998), with NaCoDAE/HTN, an extension of NaCoDAE suitable for working with simple hierarchical task networks (HTNs). HTE allows users to edit doctrine and select operational[5] tasks for refinement into tactical actions. NaCoDAE/HTN is used to help users select which refinements to implement. (We ignore high-level strategic planning issues because they involve political concerns that are challenging to model and simulate.) Figure 13 summarizes HICAP's integrated architecture.

HICAP's plans are represented using a variant of HTNs (Erol et al., 1994), a particularly expressive plan representation. We define a HTN as a set of tasks and their ordering relations, denoted as $N = \langle\{T_1,\ldots,T_m\},\prec\rangle$ $(m\geq 0)$. The relation $\prec$ has the form $T_i \prec T_j (i\neq j)$, and expresses temporal restrictions between tasks. Problem solving is performed by applying *methods* to decompose tasks into subtasks. Each method has the form $M = \langle l, T, N, P\rangle$, where l is a label, T is a task, N is a HTN, and $P = \langle p_1,\ldots,p_k\rangle$ is a set of preconditions for applying M. When P is satisfied, M can be applied to T, yielding N.

Three task types exist. First, *non-decomposable* tasks are tactical actions; they can occur only at leaves of the network. Second, *uniquely decomposable* tasks are specified by doctrine and are unconditional (i.e., $P = \emptyset$). Finally, *multiply decomposable* tasks can be subdivided in multiple ways according to the specific problem-solving context.

HICAP inputs a HTN describing the doctrine for an application, a second HTN for the command hierarchy, a mapping from tasks to command elements, and one set of cases for each subtask that can be decomposed in multiple ways. Under user control, HICAP outputs an edited HTN whose leaves are tactical actions.

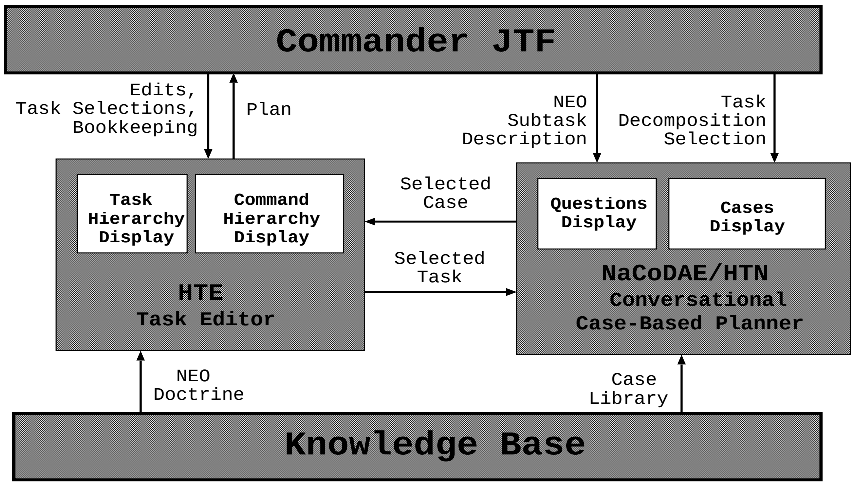


Fig. 13. The HICAP architecture

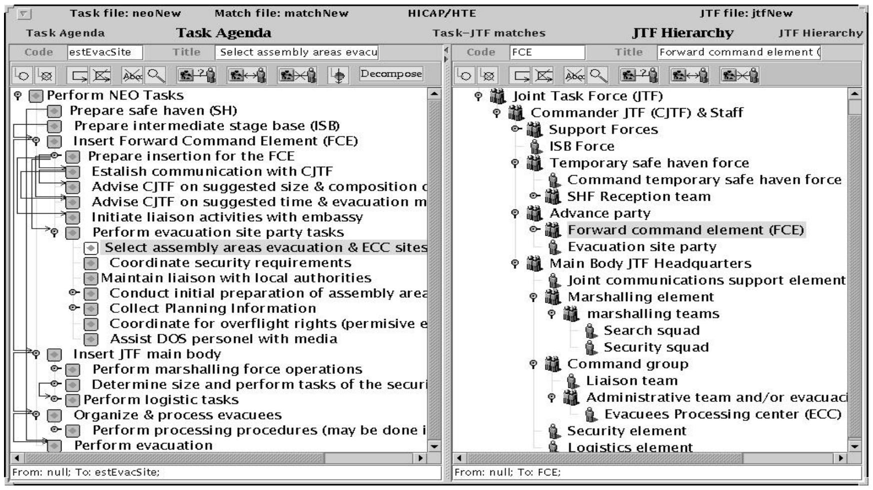
Plans and tasks in HICAP are managed by HTE (Hierarchical Task Editor), which serves as a bookkeeping tool and visualizes the task hierarchy HTN, the command hierarchy, the assignment of tasks to command elements, and task orderings (Muñoz-Avila et al., 1998). HTE can be used to:

1. browse and edit the given HTNs and links,
2. select tasks for further decomposition,
3. edit assignments of military personnel to tasks, and
4. record the completion status of tasks.

For NEO plan formulation, we elicited a HTN to capture critical planning knowledge corresponding to NEO doctrine (DoD, 1994). This substantial manual knowledge acquisition effort yielded more than 200 tasks and their ordering relations. We also elicited the JTF command hierarchy that is commonly used in NEO operations. Finally, we assigned default JTF elements responsible for each task. Figure 14 displays (left) some tasks that, according to doctrine, must be performed during a NEO and (right) the elements in the JTF responsible for them. (Also shown are some task orderings for the task agenda and a task/command mapping; the FCE is responsible for selecting evacuation assembly areas.) HTE can be used to edit the HTN (i.e., doctrine), task ordering relations, the command hierarchy, and task-command mappings. Thus, military commanders can use HTE to tailor its knowledge to the current NEO's needs.

HICAP represents decomposition methods, for multiply decomposable tasks, as case solutions. Method preconditions are represented by a case's problem specification (i.e., its $\langle q, a \rangle$ pairs). Cases denote standard operational procedures (SOP), obtainable from operational manuals, or task decompositions used in previous NEO operations. Users can direct HTE to solve/decompose one of these tasks T, at which point HICAP initiates a NaCoDAE/HTN conversation that accesses all cases for decomposing T (i.e., using T as an index for case retrieval). If all the preconditions of a SOP case are met, then it is used to decompose T. Otherwise, the cases are (incrementally) ranked by their conditions' similarities with the current planning scenario, and the user can select any of their task decompositions to apply. For example, standard procedures call for the Department of State to concentrate evacuees in the embassy prior to troop deployment. This is not always possible; escorted transports were organized after the JTF was deployed in Eastern Exit (Siegel, 1991) and the evacuees of Sharp Edge (Sachtleben, 1991) were concentrated in several places, which forced multiple separate evacuations.

NaCoDAE/HTN can be used to recursively refine selected operational-level tasks into tactical subtasks. It operates similarly to NaCoDAE except for three differences. First, all plan scenario information obtained earlier in a conversation is



```
Task file: neoNew     Match file: matchNew     HICAP/HTE     JTF file: jtfNew
Task Agenda    Task Agenda    Task–JTF matches    JTF Hierarchy    JTF Hierarchy
Code estEvacSite   Title Select assembly areas evacu      Code FCE   Title Forward command element (
Decompose

Perform NEO Tasks                                          Joint Task Force (JTF)
  Prepare safe haven (SH)                                    Commander JTF (CJTF) & Staff
  Prepare intermediate stage base (ISB)                        Support Forces
  Insert Forward Command Element (FCE)                         ISB Force
    Prepare insertion for the FCE                              Temporary safe haven force
    Estalish communication with CJTF                             Command temporary safe haven force
    Advise CJTF on suggested size & composition c                SHF Reception team
    Advise CJTF on suggested time & evacuation m               Advance party
    Initiate liaison activities with embassy                     Forward command element (FCE)
    Perform evacuation site party tasks                          Evacuation site party
      Select assembly areas evacuation & ECC sites             Main Body JTF Headquarters
      Coordinate security requirements                           Joint communications support element
      Maintain liaison with local authorities                    Marshalling element
      Conduct initial preparation of assembly area                 marshalling teams
      Collect Planning Information                                   Search squad
      Coordinate for overflight rights (permisive e                  Security squad
      Assist DOS personel with media                             Command group
  Insert JTF main body                                             Liaison team
    Perform marshalling force operations                           Administrative team and/or evacuaci
    Determine size and perform tasks of the securi                   Evacuees Processing center (ECC)
    Perform logistic tasks                                       Security element
  Organize & process evacuees                                    Logistics element
    Perform processing procedures (may be done i
  Perform evacuation
From: null; To: estEvacSite;                               From: null; To: FCE;
```

Fig. 14. HTE snapshot displaying the task (left) and command hierarchies (right); Arrows denote ordering constraints

available for computing case similarities. Second, the user-selected solution is applied to decompose the current task into subtasks (i.e., solutions are task decompositions rather than action sequences). All expansions are immediately displayed by HTE. Non-decomposable tasks corresponding to tactical actions will eventually be reached through the task expansion process. Finally, SOP cases require complete matching, and cannot be selected otherwise. In contrast, cases based on previous NEOs support partial matching: they can be selected even if some of their questions have not been answered, or if the user's answers differ from the case's.

In summary, HICAP integrates HTE with NaCoDAE/HTN to formulate plans that are in accordance with both doctrine and stored cases. In doing so, it satisfies the requirements stated in Section 5.1 for a NEO plan authoring assistant. The following section describes an example of its use, followed by an evaluation with a simulator.

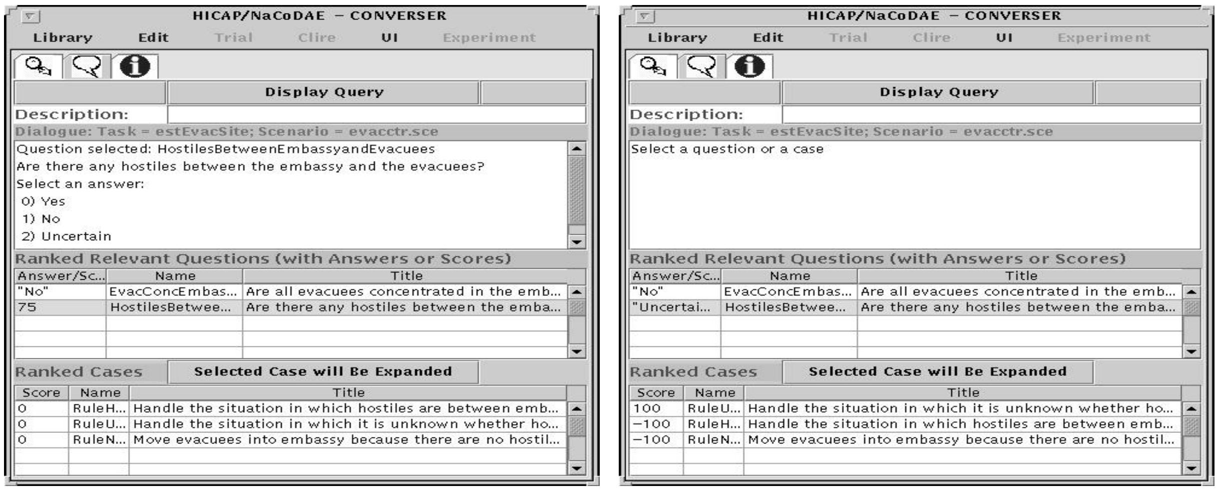
5.3. HICAP Example

During NEO planning, the user views the top level tasks first, revising them or their assignments if necessary. Any task can be selected and expanded. Figure 14 shows an intermediate stage during this process. The user has selected the task *Select assembly areas for evacuation & ECC (Evacuation Control Center) sites*, which is highlighted together with its assigned command element.

Although SOPs dictate that the embassy is the ideal assembly area for all evacuees, this is not always feasible. A military planner can select this task to initiate a NaCoDAE/HTN conversation (see Figure 15 (left)), which will them to assess and examine the alternative task decompositions listed under "Ranked Cases."

Suppose the user answers *Are there any hostiles between the embassy and the evacuees?* with "uncertain." This yields a perfect match with the second displayed case (Figure 15 (left)), resulting in a revised case ranking (Figure 15 (right)). If the user selects it to decompose this task, then Figure 16 shows the decomposition containing two new subtasks (i.e., corresponding to this case's decomposition network). The first subtask, *Send UAV (Unmanned Air Vehicle) to ...*, is non-decomposable; it corresponds to a tactical action. If the user tells HICAP to decompose the second subtask, *Determine if hostiles are present*, HICAP will initiate a new NaCoDAE/HTN dialogue (also in Figure 16).

If the user next selects *The UAV detects hostiles* method, the *Handle hostile presence* subtask will be added to the HTN (Figure 17). If the user then decides to decompose that task, a new NaCoDAE/HTN dialogue begins. Suppose the user answers *Can the hostile forces ...* with "yes." This matches the situation in Operation Eastern Exit in which the evacuees were dispersed



Description:
Dialogue: Task = estEvacSite; Scenario = evacctr.sce
Question selected: HostilesBetweenEmbassyandEvacuees
Are there any hostiles between the embassy and the evacuees?
Select an answer:
0) Yes
1) No
2) Uncertain

Ranked Relevant Questions (with Answers or Scores)

Answer/Sc...	Name	Title
"No"	EvacConcEmbas...	Are all evacuees concentrated in the emb...
75	HostilesBetwee...	Are there any hostiles between the emba...

Ranked Cases

Score	Name	Title
0	RuleH...	Handle the situation in which hostiles are between emb...
0	RuleU...	Handle the situation in which it is unknown whether ho...
0	RuleN...	Move evacuees into embassy because there are no hostil...

Description:
Dialogue: Task = estEvacSite; Scenario = evacctr.sce
Select a question or a case

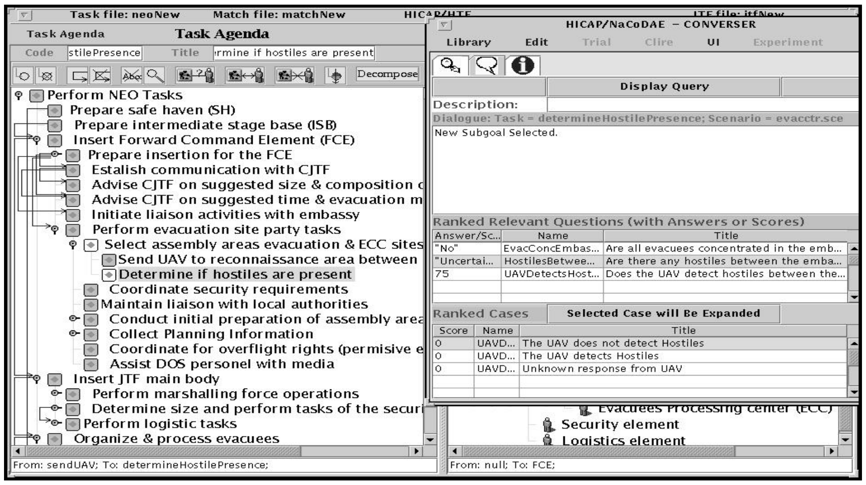
Ranked Relevant Questions (with Answers or Scores)

Answer/Sc...	Name	Title
"No"	EvacConcEmbas...	Are all evacuees concentrated in the emb...
"Uncertai...	HostilesBetwee...	Are there any hostiles between the emba...

Ranked Cases

Score	Name	Title
100	RuleU...	Handle the situation in which it is unknown whether ho...
−100	RuleH...	Handle the situation in which hostiles are between emb...
−100	RuleN...	Move evacuees into embassy because there are no hostil...

Fig. 15. NaCoDAE/HTN's interface, before (left) and after (right) answering a question. The top window directs the user and lists the possible answers. The lower windows display the ranked questions and cases.



Code: stilePresence
Title: ermine if hostiles are present

- Perform NEO Tasks
  - Prepare safe haven (SH)
  - Prepare intermediate stage base (ISB)
  - Insert Forward Command Element (FCE)
    - Prepare insertion for the FCE
    - Estalish communication with CJTF
    - Advise CJTF on suggested size & composition c
    - Advise CJTF on suggested time & evacuation m
    - Initiate liaison activities with embassy
    - Perform evacuation site party tasks
      - Select assembly areas evacuation & ECC sites
        - Send UAV to reconnaissance area between
        - Determine if hostiles are present
      - Coordinate security requirements
      - Maintain liaison with local authorities
      - Conduct initial preparation of assembly area
      - Collect Planning Information
      - Coordinate for overflight rights (permisive e
      - Assist DOS personel with media
  - Insert JTF main body
    - Perform marshalling force operations
    - Determine size and perform tasks of the securi
    - Perform logistic tasks
  - Organize & process evacuees

From: sendUAV; To: determineHostilePresence;

Description:
Dialogue: Task = determineHostilePresence; Scenario = evacctr.sce
New Subgoal Selected.

Ranked Relevant Questions (with Answers or Scores)

Answer/Sc...	Name	Title
"No"	EvacConcEmbas...	Are all evacuees concentrated in the emb...
"Uncertai...	HostilesBetwee...	Are there any hostiles between the emba...
75	UAVDetectsHost...	Does the UAV detect hostiles between the...

Ranked Cases

Score	Name	Title
0	UAVD...	The UAV does not detect Hostiles
0	UAVD...	The UAV detects Hostiles
0	UAVD...	Unknown response from UAV

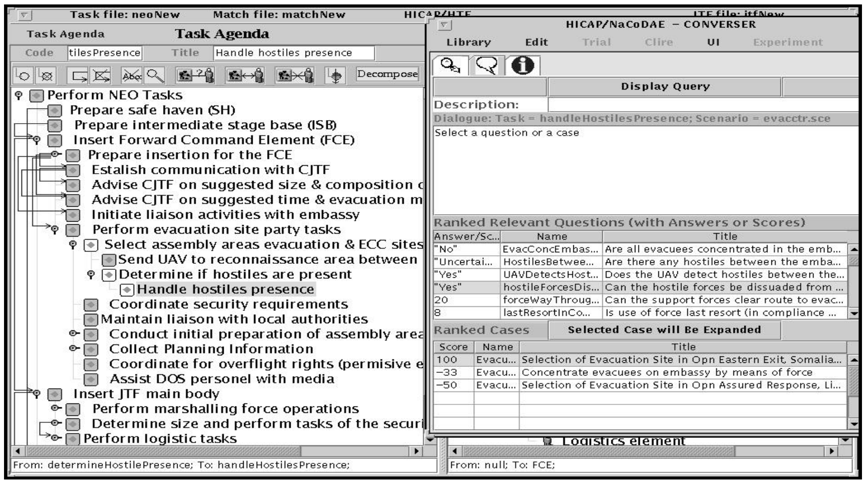
From: null; To: FCE;

Fig. 16. HICAP's interface after selecting the *Determine hostile presence* task

into multiple locations in Mogadishu and escorted transports gathered all evacuees into the embassy. If the user selects this case, then its two non-decomposable subtasks, *Assign dissuasive escort* and *Escort evacuees to embassy*, will be added to the HTN.

5.4. Evaluation

We tested HICAP's ability to choose successful plans for a specific NEO subtask. Two researchers performed this experiment: one operated a military simulator while the other operated HICAP. A strict blind was imposed to ensure that the HICAP user had no knowledge concerning the simulated hostile forces; this tests HICAP's utility for planning under realistic situations where decision makers have uncertain information about the



Task Agenda

Code: tilesPresence Title: Handle hostiles presence

```
Perform NEO Tasks
  Prepare safe haven (SH)
  Prepare intermediate stage base (ISB)
  Insert Forward Command Element (FCE)
    Prepare insertion for the FCE
    Estalish communication with CJTF
    Advise CJTF on suggested size & composition c
    Advise CJTF on suggested time & evacuation m
    Initiate liaison activities with embassy
    Perform evacuation site party tasks
      Select assembly areas evacuation & ECC sites
        Send UAV to reconnaissance area between
        Determine if hostiles are present
          Handle hostiles presence
      Coordinate security requirements
      Maintain liaison with local authorities
      Conduct initial preparation of assembly area
      Collect Planning Information
      Coordinate for overflight rights (permisive e
      Assist DOS personel with media
  Insert JTF main body
    Perform marshalling force operations
    Determine size and perform tasks of the securi
    Perform logistic tasks
```

From: determineHostilePresence; To: handleHostilesPresence;

Description:

Dialogue: Task = handleHostilesPresence; Scenario = evacctr.sce

Select a question or a case

Ranked Relevant Questions (with Answers or Scores)

Answer/Sc...	Name	Title
"No"	EvacConcEmbas...	Are all evacuees concentrated in the emb...
"Uncertai...	HostilesBetwee...	Are there any hostiles between the emba...
"Yes"	UAVDetectsHost...	Does the UAV detect hostiles between the...
"Yes"	hostileForcesDis...	Can the hostile forces be dissuaded from ...
20	forceWayThroug...	Can the support forces clear route to evac...
8	lastResortInCo...	Is use of force last resort (in compliance ...

Ranked Cases — Selected Case will Be Expanded

Score	Name	Title
100	Evacu...	Selection of Evacuation Site in Opn Eastern Exit, Somalia...
−33	Evacu...	Concentrate evacuees on embassy by means of force
−50	Evacu...	Selection of Evacuation Site in Opn Assured Response, Li...

From: null; To: FCE;

Fig. 17. Advising the user on how to handle a hostile presence

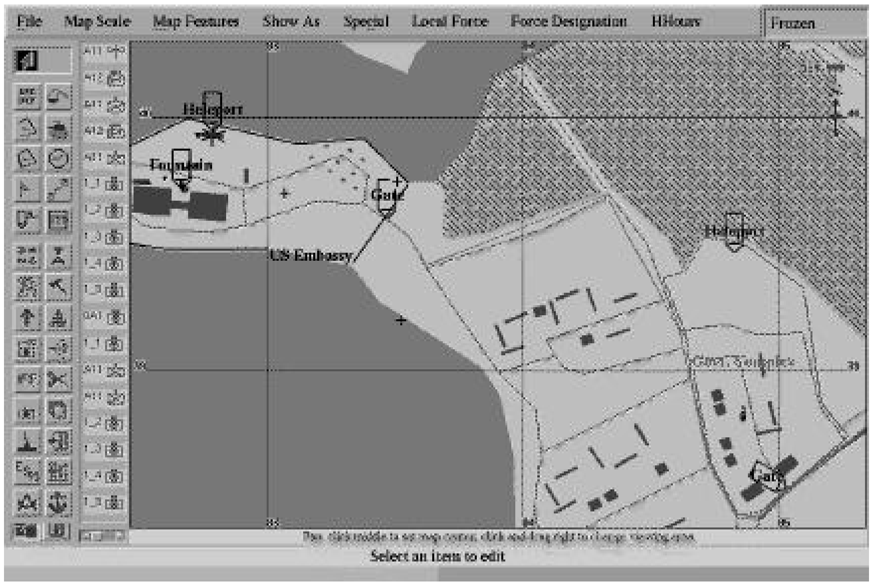


Fig. 18. A MCSF snapshot of the Camp Lejeune area

world state. We hypothesized that HICAP would allow users to choose a relatively successful plan vs. three alternative methods for selecting plans: *random choice*, *heuristic choice*, and choice by the *most frequently used plan* in past NEOs.

5.4.1. The ModSAF Simulator. We used Marine Corps SAF (MCSF), a variant of ModSAF (Modular Semi-Automated Forces), to evaluate the quality of NEO plans elicited using HICAP. ModSAF, developed by the US Army to inject simulated auxiliary forces into training exercises, has been deployed to simulate real-world military scenarios (Ceranowicz, 1994). It is a finite state simulation with modular components that represent individual entities and sub-entities. For example, a simulated tank's physical components include its turret, and its behavior components include move, attack, target, and react to fire. Certain 3D aspects are also represented (e.g., terrain elevation, tree and vegetation, rivers, oceans, atmospheric conditions), which can affect sensory and movement behavior. Figure 18's MSCF snapshot displays a simulated USA Embassy, a host country government compound, and some simulated objects (e.g., a transport helicopter is positioned at the heliport within the Embassy site).

MCSF is a nondeterministic simulator that models multiple sources of stochastic variation. Some events are determined by a random number generator; others are highly sensitive to the initial startup conditions. MCSF simulates the behavior of military units in context as they follow given tactical orders. Therefore, MCSF can simulate simplified NEO subtasks in which a single planning decision determines tactical orders.

5.4.2. Empirical Methodology. We created a NEO subtask for this evaluation concerning how to move 64 evacuees from a meeting site (i.e., a crossroads in an uninhabited area outside of a city) to a US embassy. Evacuees had to be transported (eight per vehicle) through this undeveloped area, which had heavy tree cover, and out through a city past a local government complex and to the US embassy. This NEO context requires only a single tactical plan decision with four distinct planning solutions:

1. Land evacuation using 8 armored trucks
2. Land evacuation using 8 armored trucks with an 8 tank escort
3. Air evacuation using 8 transport helicopters
4. Air evacuation using 8 transport helicopters with an 8 attack helicopter escort

The military units used in this simulation are typical of those available to Marine Expeditionary Units (MEUs) that frequently perform NEOs. A detailed terrain database of Camp Lejeune (North Carolina, USA) was chosen to simulate the environment, where some MEUs train for NEOs.

Two scenarios were defined that were identical except for the type of hostile forces. All hostiles were two-person dismounted infantry teams. Hostile teams in both scenarios were armed with two automatic rifles and a portable missile launcher. However, the scenarios were distinguished by the type of hostile missile: either anti-tank or anti-air missiles, but not both. These hostile forces are typical of the kinds of hostile forces encountered in NEOs. The positions of the hostile teams were the same for both scenarios and selected to ensure that the opposing forces will meet.

All four plan options were simulated ten times per scenario, resulting in 80 ($2 \times 4 \times 10$) total MCSF simulations. As noted earlier, MCSF is nondeterministic. For example, slight differences produced by MCSF's stochastic movement models resulted in very different formations of friendly units when they first encountered the hostile teams. These differences often lead to drastically different simulated battle outcomes.

The HICAP user had no knowledge of the scenarios being tested; scenario information was gradually extracted through the questions prompted by NaCoDAE/HTN. That is, case-based planning was done with incomplete information about the world. Furthermore, the effects of actions were uncertain; the only way to learn the effects of an action was to actually execute it. This contrasts with the assumptions of traditional planning approaches (Fikes & Nilsson, 1971).

5.4.3. Results. Table 5 summarizes the casualty results for the 80 simulations, which each required approximately 15 minutes to run. The success measures were taken from the US Navy's Measures of Effectiveness (MOEs) published in the Universal Naval Task List. Recommended MOEs are specified for evaluating each kind of military operation. There are several MOEs for the tactical aspects of NEOs; three were chosen as most important for evaluating the results of this experiment: (1) number of evacuees safely moved, (2) number of casualties to friendly forces, and (3) number of casualties to hostile forces.

HICAP did not chose the same tactical plan for both scenarios. For the first (anti-tank) scenario, it chose to move the evacuees by helicopter with an attack helicopter escort.[6] For the second (anti-air) scenario, it chose to move evacuees by armored truck with a tank escort.

HICAP's results were compared with the plans chosen by the other three plan selection methods. *Random choice* simply averages the results of all four plans. *Heuristic choice* always sent an escort, and its results were the average of the two escort plans. The *most frequently used* plan for this subtask in recent NEOs moved evacuees using escorted land vehicles. Figure 19 compares the effectiveness of these four selection methods.

Table 5. Summaries of casualties (mean & standard deviation) from 80 MCSF simulations

	Scenario 1						**Scenario 2**					
Tactical Plans	**Evacuees**		**Friends**		**Hostiles**		**Evacuees**		**Friends**		**Hostiles**	
Land	6.4	5.1	0.8	0.6	5.5	1.3	0		0		4.2	0.8
Land/Escort	3.2	10.1	7.4	1.5	6.5	1.8	0		0		7.6	0.6
Air	56.0	9.2	7.0	1.2	0		64.0	0.0	8.0	0.0	0	
Air/Escort	0		0.8	1.5	8.0	0.0	20.0	18.6	6.3	4.4	5.7	2.9

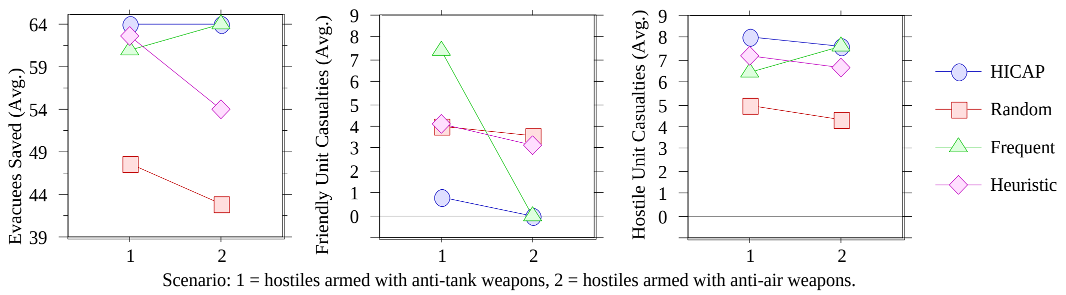


Fig. 19. Comparison of plan selection methods using Navy MOEs for a NEO subtask

Overall, HICAP selected plans of higher quality than the other methods because its plan selection decisions are tailored to the characteristics of each scenario.

5.5. *Discussion*

The HICAP plan authoring tool helps users to formulate a course of action for hierarchical tasks. It is the first tool to combine a task guideline decomposition process with CBR to support interactive plan formulation. It yields plans that benefit from previous experiences and are sound according to predefined guidelines. HICAP also supports experience sharing, thus allowing planners to exploit knowledge from other planning experts. These design characteristics enhance HICAP's acceptance by military planning personnel.

We are currently integrating HICAP with SHOP (Nau et. al., 1999), an HTN planner that can process numeric expressions. In addition, we are extending HICAP with resource tracking capabilities to provide conflict management control for user edits.

6. Related and Future Research

6.1. *Interactive CBR*

Inference Corporation (1995) introduced the first commercial CCBR system, and several other companies have since developed similar products (Watson, 1997). Although many successful CCBR applications have been published (e.g., Nguyen et al., 1993), only three other research groups have comprehensively addressed this topic. First, Shimazu (et al., 1994; 1999) examined how multiple interface modes can facilitate case retrieval and described the benefits of indexing cases using multiple hierarchies and an entropy question-ranking procedure. Supporting multiple ways to retrieve cases could be useful in a distributed military environment, in which users may differ in their viewpoints. Shimazu (1999) also showed how *cue questions* from recorded spoken dialogues can assist with indexing cases using scripts in EXPERTGUIDE. Script representations are particularly appropriate for planning tasks, and we intend to address this in our future research.

Second, FindMe systems (Burke et al., 1997; Burke, 1999) can be viewed as CCBR tools that focus users on the top-ranking solution and cases similar to it, where users direct the search process by requesting different answers to some questions. This simplifies conducting what-if analyses, which we plan to investigate in the context of planning tasks.

Finally, Yang and his colleagues have continued to develop CaseAdvisor. For example, Racine and Yang (1997) described how to maintain conversational case bases; we have incorporated some of their ideas into NaCoDAE. Zhang and Yang (1998) introduced a question-weighting learning algorithm, inspired by error backpropagation, that requires the user to provide feedback on their retrieval ranking preferences. Because their algorithm makes fewer assumptions than the one developed by Trott and Leng (1997), who structured the case authoring process using KADS, we will consider using it in our future research on case-based maintenance. Carrick et al. (1999) described how automated information gathering techniques could support planning tasks, which is of particular interest to military applications in which sensor information must be quickly fused in situation assessment tasks. Yang and Wu (1999) demonstrated how case-clustering techniques and an entropy-driven question-selection strategy can improve retrieval precision and efficiency. In our context, cases are pre-clustered, and users probably would prefer question-ranking strategies that support more comprehensible explanations (McSherry, 1999). Finally, Abi-Zeid et al.'s (1999) application to search and rescue tasks highlighted CCBR's use for situation assessment (and report generation) in a time-critical context is particularly pertinent to our future research goals.

Alternative forms of interactive CBR prompt users with questions suggested by decision trees. For example, INRECA (Manago et al., 1993) uses a (global) decision tree induced from the case library, defaulting to a traditional CBR approach when the user cannot answer a given question. In contrast, NODAL$_{CBR}$ (Smyth & Cunningham, 1994) retrieves a subset M of matching cases using only zero-cost features, and then dynamically induces a local (info-gain) decision tree from M, which is used to prompt users to supply (non-zero cost) feature values. This is reminiscent of CS-IBL (Tan & Schlimmer, 1990), which incrementally evaluates features with non-zero evaluation cost and selects features for evaluation that maximize the ratio of expected match success to cost. Unlike CCBR, these question-answering processes are not user-driven.

6.2. Case library authoring

Although few researchers have focussed on automated support for case authoring, some have investigated manual methods. For example, Heider et al. (1997) describe problems due to poorly designed cases (i.e., incomplete or noisy), and a methodology for improving their quality by imposing more structure on the authoring process. Kitano et al. (1993) also describe a general methodology for building case bases. However, these publications do not target CCBR systems.

Many CBR systems use decision trees to index and retrieve cases. For example, this includes IBPRS (Ku & Suh, 1996), which uses K-trees, and INRECA (Auriol et al., 1995), which integrates decision trees and k-d trees. Clire differs from most of these approaches in that it uses trees to revise, rather than index, case indices. One exception is Cardie's (1993) TDIDT approach for feature selection, although Clire again differs in that it performs *case-specific* feature selection. Other case-specific feature selection algorithms exist (e.g., Domingos, 1997), but they assume that cases are homogeneous (i.e, described by the same questions), which is not true for most CCBR libraries. Finally, unlike the others mentioned, Clire performs in the context of a user-driven CCBR engine.

6.3. Model-based CBR

There is a long history of model-based CBR frameworks. For example, Vilain et al. (1990) introduced a representation language that supports a model-based approach for increasing learning rates and reducing the brittleness of induced generalizations for classification tasks. CADET (Navinchandra et al., 1991) uses index transformation techniques to solve mechanical design tasks by representing causal relations between problem variables. Carma (Hastings et al., 1995) uses a model-based adaptation component to increase its predictive accuracy. However, to our knowledge, no previous effort has focused on integrating model-based components to improve CCBR performance.

6.4. Crisis response planning

Case-based planning (CBP) has been the subject of extensive research (Bergmann et al., 1998). Our work is closely related to studies on hierarchical CBP (Kambhampati, 1994; Bergmann & Wilke, 1995; Branting & Aha, 1995). HICAP differs from these other approaches in that it includes the user in its problem solving loop. This is particularly important for applications like NEO planning, where automated tools are unacceptable.

Currently, no intelligent NEO planning tool has been deployed. Kostek (1988) proposed a conceptual design for predicting the force size and type required for a NEO. Chavez and Henrion (1994) described a decision-theoretic approach for instantiating a general NEO plan with specific parameters for locations, forces, and destinations, and used it to assess alternative plans. Gil et al. (1994) presented a system for predicting manning estimates for certain NEO tasks. None of these systems formulate NEO plans, although desJardins et al. (1998) proposed a distributed hierarchical planning approach for this task.

Although DARPA and other agencies have sponsored several projects related to NEO planning (e.g., ARPI (Tate, 1994)), HICAP is the *first* system to use a *conversational* case-based approach for plan formulation. HICAP allows users to incrementally elaborate a planning scenario, provides a focus of attention that guides this elaboration, and provides access to stored plan fragments for use in new NEO plans.

Some researchers have used case-based approaches for HTN planning tasks on military domains. For example, Mitchell's (1997) case-based planner selects tasks for a Tactical Response Planner. However, NEO planning requires that each task be addressed–no choice is involved–and we use CBP to instead choose *how* to perform a task. MI-CBP (Veloso et al., 1997) uses rationale-directed CBP to suggest plan modifications, but does not perform doctrine-driven task decomposition. HICAP's interactions instead focus on retrieval rather than plan adaptation and learning. IFD4's (Bienkowski & Hoebel, 1998) plan formulator automatically generates plans as guided by an editable objectives hierarchy. In contrast, HICAP's objectives are fixed, and user interaction focuses on task formulation.

Other researchers have developed related crisis response systems. Ferguson and Allen (1998) described an interactive planner for military crisis response, but their system does not use cases during plan formulation and does not perform doctrine-driven task decomposition. Likewise, Wolverton and desJardins' (1998) distributed generative planner also does not use cases. Gervasio et al. (1998) described an interactive hierarchical case-based scheduler for crisis response that does not perform interactive plan formulation. Avesani et al. (1998) described a CBP for fighting forest fires that supports interactive plan adaptation, but does not use hierarchical guidelines to formulate plans as HICAP does. Finally, Leake et al. (1996) described a CBP applied to disaster response that focuses on learning case adaptation knowledge, but it is not doctrine-driven and focuses interaction on knowledge acquisition rather than problem elicitation.

7. Conclusions

This paper summarizes our recent research on conversational case-based reasoning (CCBR), an interactive form of case-based reasoning that supports mixed-initiative case retrieval. We described our system NaCoDAE, its module Clire for revising case libraries to simplify the case authoring task, its integration with Parka-DB to improve conversational retrieval efficiency, and its extension in HICAP for plan authoring tasks. We summarized evaluations for each extension that demonstrate some confidence in their expected performance.

CCBR research is becoming more popular, probably due both to its proven utility in commercial applications and its simplicity. Of particular interest to us is integrating complementary technologies (e.g., machine learning, generative planning) with CCBR that will extend it to synthesis and knowledge management tasks, and, in doing so, prepare it for capturing additional market niches.

Acknowledgements

Many thanks to our NRL colleagues Tucker Maney, Dan McFarlane, and Jim Ballas for their contributions, and to Dana Nau for our ongoing collaborations. This research was generously supported by grants from the Office of Naval Research and the Naval Research Laboratory, and through a CRADA with Inference Corporation.

Notes

1. NaCoDAE, written in Java, is available upon request.
2. Although in some applications, the customer interacts with the system directly (e.g., Nguyen et al., 1993).
3. Standard deviations for efficiency were very small throughout our experiments, and were always below 5% for precision.
4. See www.aic.nrl.navy.mil/∼aha/neos for more information on NEOs.
5. We use the term "operational" in the military sense, rather than the sense common to AI (e.g., explanation-based reasoning). From this perspective, tasks are ordered, from abstract to concrete, as *strategic*, *operational* and *tactical*.
6. HICAP's highest ranked case was used to select its plan.

References

1. Abi-Zeid, I., Yang, Q., & Lamontagne, L. (1999). Is CBR applicable to the coordination of search and rescue operations? A feasibility study. *Proceedings of the Third International Conference on Case-Based Reasoning* (pp. 358–371). Seeon, Germany: Springer.
2. Aha, D. W., & Breslow, L. A. (1997). Refining conversational case libraries. *Proceedings of the Second International Conference on Case-Based Reasoning* (pp. 267–278). Providence, RI: Springer.
3. Aha, D.W., & Breslow, L.A. (1998). Correcting for length biasing in conversational case scoring (Technical Report AIC-98-007). Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence.
4. Aha, D.W., Maney, T., & Breslow, L.A. (1998). Supporting dialogue inferencing in conversational case-based reasoning. *Fourth European Workshop on Case-Based Reasoning* (pp. 262–273). Dublin, Ireland: Springer.
5. Auriol, E., Wess. S., Manago, M., Althoff, K.-D., & Traphöner, R. (1995). INRECA: A seamlessly integrated system based on inductive inference and case-based reasoning. *Proceedings of the First International Conference on Case-Based Reasoning* (pp. 371–380). Sesimbra, Portugal: Springer.
6. Avesani, P., Perini, A., & Ricci, F. (1998). The twofold integration of CBR in decision support systems. In D.W. Aha & J.J. Daniels (Eds.), *Case-Based Reasoning Integrations: Papers from the 1998 Workshop* (Technical Report WS-98-15). Menlo Park, CA: AAAI Press.
7. Bergmann, R., Muñoz-Avila, H., Veloso, M., & Melis, E. (1998). Case-based reasoning applied to planning tasks. In M. Lenz, B. Bartsch-Spoerl, H.-D. Burkhard, & S. Wess (Eds.) *CBR Technology: From Foundations to Applications.* Berlin: Springer.
8. Bergmann, R. & Wilke, W. (1995). Building and refining abstract planning cases by change of representation language. *Journal of AI Research*, *3*, 53–118.
9. Bienkowski, M.A., & Hoebel, L.J. (1998). Integrating AI components for a military planning application. *Proceedings of the Fifthteenth National Conference on Artificial Intelligence* (pp. 561–566). Madison, WI: AAAI Press.
10. Branting, L. K., & Aha, D. W. (1995). Stratified case-based reasoning: Reusing hierarchical problem solving episodes. *Proceedings of the Fourteenth International Joint Conference on AI* (pp. 384–390). Montreal, Canada: Morgan Kaufmann.
11. Breslow, L., & Aha, D. W. (1997a). *NaCoDAE: Navy Conversational Decision Aids Environment* (Technical Report AIC-97-018). Washington, DC: NRL, NCARAI.
12. Breslow, L., & Aha, D. W. (1997b). Simplifying decision trees: A survey. *Knowledge Engineering Review*, *12*, 1–40.
13. Brill, E. (1995). Transformation-based tagger, V1.14. See www.cs.jhu.edu/∼brill.
14. Burke, R. (1999). The Wasabi Personal Shopper: A case-based recommender system. *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (pp. 844–849). Orlando, FL: AAAI Press.
15. Burke, R., Hammond, K., & Young, B. (1997). The FindMe approach to assisted browsing. *IEEE Expert*, *12*(4), 32–40.
16. Cardie, C. (1993). Using decision trees to improve case-based learning. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 25–32). Amherst, MA: Morgan Kaufmann.
17. Carrick, C., Yang, Q., Abi-Zeid, I., & Lamontagne, L. (1999). Activating CBR systems through autonomous information gathering. *Proceedings of the Third International Conference on Case-Based Reasoning* (pp. 74–88). Seeon, Germany: Springer.
18. Ceranowicz, A. (1994). *Modular Semi-Automated Forces. Proceedings of the Winter Simulation Conference of the ACM* (pp. 755–761). New York, NY: IEEE.
19. Chavez, T., & Henrion, M. (1994). Focusing on what matters in plan evaluation: Efficiently estimating the value of information. *Proceedings of the ARPA/Rome Laboratory Knowledge-Based Planing and Scheduling Initiative* (pp. 387–399). Tuscon, AR: Morgan Kaufmann.
20. desJardins, M., Francis, A., & Wolverton, M. (1998). Hybrid planning: An approach to integrating generative and case-based planning. In D.W. Aha &

J.J. Daniels (Eds.) *Case-Based Reasoning Integrations: Papers from the 1998 Workshop* (Technical Report WS-98-15). Menlo Park, CA: AAAI Press.

21. DoD (1994). *Joint tactics, techniques and procedures for noncombat evacuation operations* (Joint Publication Report 3-07.51, Second Draft). Washington, DC: Department of Defense.
22. Domingos, P. (1997). Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review, 11*, 227–253.
23. Erol, K., Nau, D., & Hendler, J. (1994). HTN planning: Complexity and expressivity. *Proceedings of the Twelfth National Conference on Artificial Intelligence* (pp. 1123–1128). Seattle, WA: AAAI Press.
24. Ferguson, G., & Allen, J.F. (1998). TRIPS: An integrated intelligent problem-solving assistant. *Proceedings of the Fifthteenth National Conference on Artificial Intelligence* (pp. 567–572). Madison, WI: AAAI Press.
25. Fikes & Nilsson, (1971). Strips: A new approach to the application of theorem proving in problem solving. *Artificial Intelligence*, *2*, 189–208.
26. Gervasio, M.T., Iba, W., & Langley, P. (1998). Case-based seeding for an interactive crisis response assistant. In D.W. Aha & J.J. Daniels (Eds.) *Case-Based Reasoning Integrations: Papers from the 1998 Workshop* (Technical Report WS-98-15). Menlo Park, CA: AAAI Press.
27. Gil, Y., Hoffman, M. & Tate, A. (1994). Domain-specific criteria to direct and evaluate planning systems. *Proceedings of the ARPA/Rome Laboratory Knowledge-Based Planing and Scheduling Initiative* (pp. 433–444). Tuscon, AR: Morgan Kaufmann.
28. Hastings, J., D., Branting, L. K., & Lockwood, J. A. (1995). Case adaptation using an incomplete causal model. *Proceedings of the First International Conference on Case-Based Reasoning* (pp. 181–192). Sesimbra, Portugal: Springer-Verlag.
29. Heider, R., Auriol, E., Tartarin, E., & Manago, M. (1997). Improving the quality of case bases for building better decision support systems. In R. Bergmann & W. Wilke (Eds.) *Fifth German Workshop on CBR: Foundations, Systems, and Applications* (Technical Report LSA-97-01E). University of Kaiserslautern, Department of Computer Science.
30. Hendler, J., Stoffel, K., & Taylor, M. (1996). *Advances in high performance knowledge representation* (Technical Report CS-TR-3672). College Park, MD: University of Maryland, Department of Computer Science.
31. Inference Corporation (1995). *CBR2: Designing CBR Express Case Bases.* Unpublished manuscript.
32. Kambhampati, S. (1994). Exploiting causal structure to control retrieval and refitting during plan reuse. *Computational Intelligence*, *10*, 213–244.
33. Kitano H., Shimazu H., & Shibata A. (1993). Case-method: A methodology for building large-scale case-based systems. *Proceedings of the Eleventh National Conference on Artificial Intelligence* (pp. 303–308). Washington, DC: AAAI Press.
34. Kostek, S.R. (1988). *A User's Design of a Decision Support System for Noncombatant Evacuation Operations for United States Central Command.* Master's thesis, School of Engineering, Air Force Institute of Technology, Dayton, Ohio.
35. Ku, S., & Suh, Y.-H. (1996). An investigation of the K-tree search algorithm for efficient case representation and retrieval. *Expert Systems with Applications, 11*, 571–581.
36. Lambert, K.S. (1992). *Noncombatant evacuation operations: Plan now or pay later* (Technical Report). Newport, RI: Naval War College.
37. Leake, D. B., Kinley, A., & Wilson, D. (1996). Acquiring case adaptation knowledge: A hybrid approach. *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (pp. 684–689). Portland, OR: AAAI Press.
38. Manago, M., Althoff, K.-D., Auriol, E., Traphoner, R., Wess, S., Conruyt, N., & Maurer, F. (1993). Induction and reasoning from cases. *Proceedings of the First European Workshop on Case-Based Reasoning* (pp. 313–318). Kaiserslautern, Germany: Springer-Verlag.
39. McSherry, D. (1999). Interactive case-based reasoning in sequential diagnosis. *Applied Intelligence*, this issue.
40. Mitchell, S.W. (1997). A hybrid architecture for real-time mixed-initiative planning and control. *Proceedings of the Ninth Conference on Innovative Applications of Artificial Intelligence* (pp. 1032–1037). Providence, RI: AAAI Press.
41. Muñoz-Avila, H., Breslow, L.A., Aha, D.W., & Nau, D. (1998). *Description and functionality of HTE (Version 2.90).* (Technical Report AIC-98-022). Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence.
42. Muñoz-Avila, H., McFarlane, D., Aha, D.W., Ballas, J., Breslow, L.A., & Nau, D. (1999). *Using guidelines to constrain interactive case-based HTN planning. Proceedings of the Third International Conference on Case-Based Reasoning* (pp. 288–302). Seeon, Germany: Springer.
43. Nau, D., Cao, Y., Lotem, A., & Muñoz-Avila, H. (1999). SHOP: Simple hierarchical ordered planner. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence* (pp. –). Stockholm: AAAI Press.
44. Navinchandra, D., Sycara, K., & Narasimhan, S. (1991). A transformational approach to case based synthesis. *AI in Engineering Design and Manufacturing, 5.*
45. Nguyen, T., Czerwinsksi, M., & Lee, D. (1993). COMPAQ QuickSource: Providing the consumer with the power of artificial intelligence. *Proceedings of the Fifth Conference on Innovative Applications of Artificial Intelligence* (pp. 142–150). Washington, DC: AAAI Press.
46. Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106.
47. Quinlan, J. R. (1989). Unknown attribute values in induction. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 164–168). Ithaca, NY: Morgan Kaufmann.
48. Racine, K., & Yang, Q. (1997). Maintaining unstructured case bases. *Proceedings of the Second International Conference on CBR* (pp. 553–564). Providence, RI: Springer.

49. Sachtleben, G.R. (1991). Operation Sharp Edge: The Corps MEU (SOC) Program in action. *Marine Corps Gazette*, 11, 76–86.
50. Shimazu, H. (1999). Translation of tacit knowledge into explicit knowledge: Analyses of recorded conversations between customers and human agents. In D.W. Aha, I. Becerra-Fernandez, F. Maurer, & H. Muñoz-Avila (Eds.) *Exploring Synergies of Knowledge Management and Case-Based Reasoning: Papers from the AAAI Workshop* (Technical Report WS-99-10). Menlo Park, CA: AAAI Press
51. Shimazu, H., Shibata, A., & Nihei, K. (1994). Case-based retrieval interface adapted to customer-initiated dialogues in help desk operations. *Proceedings of the Twelfth National Conference on Artificial Intelligence* (pp. 513–518). Seattle, WA: AAAI Press.
52. Shimazu, H., Shibata, A., & Nihei, K. (1999). ExpertGuide: A conversational case-based reasoning tool for developing mentors in knowledge spaces. *Applied Intelligence*, this issue.
53. Siegel, A.B. (1991). *Eastern Exit: The noncombatant evacuation operation (NEO) from Mogadishu, Somalia, in January 1991* (Technical Report CRM 91-221). Arlington, VA: Center for Naval Analyses.
54. Siegel, A.B. (1995). *Requirements for humanitarian assistance and peace operations: Insights from seven case studies* (Technical Report CRM 94-74). Arlington, VA: Center for Naval Analyses.
55. Smyth, B., & Cunningham, P. (1994). A comparison of incremental CBR and inductive learning. In M. Keane, J. P. Haton, & M. Manago (Eds.) *Working Papers of the Second European Workshop on Case-Based Reasoning*. Chantilly, France: Unpublished.
56. Stahl, David T. (1992). *Noncombatant evacuation operations in support of the National Military Strategy* (Technical Report). Fort Leavenworth, KA: United States Army Command and General Staff College, School of Advanced Military Studies.
57. Tan, M., & Schlimmer, J. C. (1990). Two case studies in cost-sensitive concept acquisition. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 854–860). Boston, MA: AAAI Press.
58. Tate, A. (1994). Mixed initiative planning in O-Plan2. *Proceedings of the ARPA/Rome Laboratory Knowledge-Based Planing and Scheduling Initiative* (pp. 512–516). Tuscon, AR: Morgan Kaufmann.
59. Trott, J. R., & Leng, B. (1997). An engineering approach for troubleshooting case bases. *Proceedings of the Second International Conference on Case-Based Reasoning* (pp. 178–189). Providence, RI: Springer.
60. Veloso, M., Mulvehill, A.M., & Cox, M.T. (1997). Rationale-supported mixed-initiative case-based planning. *Proceedings of the Ninth Conference on Innovative Applications of Artificial Intelligence* (pp. 1072–1077). Providence, RI: AAAI Press.
61. Vilain, M., Koton, P., & Chase, M. P. (1990). On analytical and similarity-based classification. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 867–874). Boston, MA: AAAI Press.
62. Watson, I. (1997). *Applying case-based reasoning: Techniques for enterprise systems*. San Francisco: Morgan Kaufmann.
63. Wettschereck, D., Aha, D. W., & Mohri, T. (1997). A review and comparative evaluation of feature weighting methods for lazy learning algorithms. *Artificial Intelligence Review*, *11*, 273-314.
64. Wolverton, M., & desJardins, M. (1998). Controlling communication in distributed planning using irrelevance reasoning. *Proceedings of the Fifthteenth National Conference on Artificial Intelligence* (pp. 868–874). Madison, WI: AAAI Press.
65. Yang, Q., & Wu, J. (1999). Enhancing the effectiveness of interactive case-based reasoning with clustering and decision forests. *Applied Intelligence*, this issue.
66. Zhang, Z., & Yang, Q. (1998). Towards lifetime maintenance of case base indexes for continual case based reasoning. *Proceedings of the International Conference on Artificial Intelligence: Methodology, Systems, Applications*. Sozopol, Bulgaria: Springer.