Applied Intelligence 14, 77–94, 2001

A Case-Based Framework for Interactive Capture and Reuse of Design Knowledge

DAVID B. LEAKE AND DAVID C. WILSON
Computer Science Department, Lindley Hall 215, Indiana University, 150 S. Woodlawn Avenue, Bloomington, IN 47405-7104, USA
leake@cs.indiana.edu
davwils@cs.indiana.edu

Abstract. Aerospace design is a complex task requiring access to large amounts of specialized information. Consequently, intelligent systems that support and amplify the abilities of human designers by capturing and presenting relevant information can profoundly affect the speed and reliability of design generation. This article describes research on supporting aerospace design by integrating a case-based design support framework with interactive tools for capturing expert design knowledge through "concept mapping." In the integrated system, interactive concept mapping tools provide crucial functions for generating and examining design cases and navigating their hierarchical structure, while CBR techniques facilitate retrieval and aid interactive adaptation of designs. Our goal is both to provide a useful design aid and to develop general interactive techniques to facilitate case acquisition and adaptation. Experiments illuminate the performance of the system's context-sensitive retrieval during interactive case adaptation and the conditions under which it provides the most benefit.

Keywords: case-based reasoning, concept mapping, interactive systems, integrations, knowledge acquisition, design

1. Overview

Aerospace design is a complex process that requires designers to address complicated issues involving numerous specialized areas of expertise. No single designer can be an expert in every relevant area, and becoming proficient may require years of experience. Consequently, intelligent systems to support and amplify the abilities of human designers have the potential to profoundly affect the speed and reliability of design generation. An appealing approach is to augment the designers' own design experiences with relevant information from prior designs: to provide support with case-based reasoning (CBR) [1–5].

Ideally, case-based design support tools will include three related capabilities to aid reuse of designs: capture of and access to specific design experiences to enable "experience sharing" [6]; support for new designers as they try to understand the lessons of those prior experiences; and support for adapting prior designs to fit new design goals. To be practical to develop, the tools must not require the encoding of extensive domain knowledge. For designer acceptance, they must leave the designer in control. This article describes principles to address these goals and techniques for their application in the interactive design support framework DRAMA (Design Retrieval and Adaptation Mechanisms for Aerospace).

The DRAMA project integrates case-based reasoning with interactive tools for capturing expert design knowledge through "concept mapping" [7], with the goal of leveraging off the strengths of both approaches. The project uses interactive concept mapping tools, developed by the Concept Mapping group at the University of West Florida, led by Dr. Alberto Cañas, to provide an interactive interface and crucial functions for generating and examining design cases, as well as for navigating their hierarchical structure [8]. Using

concept maps as a case representation, as well as using the concept mapping tools to manipulate them, provide the novel capability for users themselves to develop and revise case representations. This raises interesting research questions about reconciling the conflicting goals of flexibility, customization, and case standardization as the case library grows.

The implemented DRAMA system supports browsing of prior design knowledge and provides designers with concrete examples of designs and design adaptations from similar prior problems. At the same time, the DRAMA interface unobtrusively acquires new examples by monitoring the user's interactive design process. This monitoring is also used to dynamically adjust the relevance criteria for retrieving prior experiences, exploiting task-based information without requiring the user to provide it explicitly. We present experiments examining this behavior in detail, demonstrating how the benefits of this automatic retrieval approach vary under conditions modeling different levels of design novelty, different stages in the design process, and different levels of user expertise.

This article first presents some basic tenets that motivate the design of DRAMA's interactive framework. It then briefly summarizes case-based reasoning and concept mapping. It next describes the system itself and experimental results, followed by perspective relating its approach to other work in case-based reasoning and design. We close with a discussion of principles developed by the project and important future steps.

2. The Task Domain

A significant concern in aerospace design is "knowledge loss:" that critical design expertise for their programs is the domain of a few experts and will be lost when they retire or leave the organization. This has given rise to knowledge preservation efforts, some of which have employed CBR. For example, the RECALL tool at the NASA Goddard Space Flight Center was developed to store and access textual reports of important lessons [9]. However, even when records have been captured, they may be hard to understand and reuse. Different experts may conceptualize designs very differently, making it hard to interpret their notes of prior designs.

To make records more comprehensible, projects have investigated the use of *concept mapping* [7]. The goal of concept mapping for design is to capture not only important features of the designs themselves, but also designers' conceptualizations of those designs—the relationships and rationale for their components. This raises the question of how to organize and access the knowledge that concept maps capture, and how to facilitate its reuse.

Our framework uses interactive CBR techniques to support retrieval and reuse of designs represented as concept maps. The processes will be illustrated with simple examples concerning high-level configuration of airliners (e.g., to select appropriate engines); a long-term goal is to develope richer concept maps to explore the framework as applied to a design initiative for reusable spacecraft.

3. Tenets of the Approach

The goals of the DRAMA project are twofold: First, to develop useful tools for aerospace design, and, second, to establish a general "knowledge-light" [10] framework for interactive case-based design support systems. The tenets shaping this general framework are:

- *The systems developed should leverage a designer's knowledge, rather than attempting to replace it.* This motivates the focus on interactivity and design support rather than autonomous design. All parts of the process accept user control and interactive problem-solving.
- *Designs take many forms.* The system must be able to support multiple (potentially idiosyncratic) design representations, but should also encourage and support standardization when that does not impose a burden.
- *Support information should automatically be focused on the current task.* This requires that the system monitor the task context in order to anticipate information needs and to determine how to fulfill them.
- *Learning must play a central role, both at the design level and at the level of design manipulation.* This requires the capability to capture and reuse multiple types of cases.

4. Background

4.1. Case-Based Design Support

CBR systems learn and reason by capturing and reusing lessons from analogous prior experiences. The lessons may include a wide range of information such as useful solution strategies to follow, mistaken strategies to

avoid, or likely outcomes if a given strategy is followed. The fundamental process of CBR—retrieving relevant cases and using them as a guideline for new reasoning—naturally lends itself to "retrieve and propose" systems that support human reasoners by presenting the guidance of relevant prior cases [11]. Many such systems have been developed and a growing number fielded [5,12].

Case-based design has been a particularly active CBR research area. It has been investigated for tasks including designing aircraft subsystems [13], architectural design [14–18], circuit design [19], and mechanical design [20]. Design cases may record information such as the designs themselves, traces of design successes and failures, the designer's design actions, and the corresponding rationale. The case library serves as a memory of suggestions and warnings to augment the current designer's expertise. Because each suggestion is directly grounded in the experience of a prior episode, it can be explained to the current designer in terms of the results in that prior situation, and the designer can evaluate the advice by considering the relevance of the prior case to the new situation.

Many existing case-based design systems display impressive capabilities, but at the expense of considerable development effort to tailor them to domain-specific needs. The aim of DRAMA is instead to unobtrusively build up case knowledge from interactions with users as they generate designs, and to use its monitoring of the design process to extract contextual information that can be used to proactively focus retrieval on useful information as the user adapts prior cases to fit new situations. The goal is not to provide autonomous design generation or adaptation capabilities, but instead to provide useful capabilities that can be realized with limited knowledge acquisition effort.

4.2. Concept Mapping

Concept mapping is a process designed to reveal internal cognitive structures by externalizing them in terms of networks of concepts and propositions. A concept map (CMap) is a two-dimensional representation of a set of concepts and their relationships. Individual concepts are linked to related concepts through one or two-way links, each link associated with a label/proposition describing the relationship. The vertical axis generally expresses a hierarchical framework for the concepts; for example, a concept map of design problems might represent a hierarchy of abstract and more specific problems. However, we stress that there is no requirement that they represent particular relationships; they are compatible with *any* structured representation. Semantic networks are a form of concept map, but concept maps are a more general notion: concept maps are not constrained by syntactic rules and have no associated semantics; they are normally seen as a medium for informally "sketching out" conceptual structures.

Different reasoners are likely to conceptualize a space differently, with important variations reflected in the differences in the maps they generate. For example, a designer specializing in airflow might include features such as wing or surface shapes and operational constraints that require them (e.g., the need for short-field landings), while an avionics designer would focus on very different features. The concept mapping process is intended to help individuals to clarify their own conceptualizations and to capture them in a form that is accessible for examination by others (e.g., members of a design team seeking to understand the expert's design to evaluate or modify it, or novices seeking to increase their own understanding). One recent effort integrated concept mapping into a set of knowledge construction and sharing tools used to link over a thousand schools in Latin America [21].

The visual presentation of information in concept maps provides a natural starting point for organizing and accessing information. For example, Fig. 1 shows a sample CMap describing the basic structure of the Boeing 777 aircraft. Three nodes of this CMap are associated with images: An exterior view is accessible from the top-level node, an engine picture is associated with the engine node, and a schematic diagram of the seating layout is accessible from the nodes for any of the seating classes. The CMap is displayed by the CMap software tools to be described in the following section. Note that the maps shown in our example CMaps are high-level maps to demonstrate system capabilities; more specific maps used by expert designers would include more abstruse features or finer-grained technical details.

4.3. Building Concept Maps

Procedures to aid the initial generation of CMaps have been described in a number of sources, primarily for use in instructional contexts to illuminate the structure of domains of study, further the synthesis of useful ideas, and encourage their analysis (e.g., [7, 22]). Computerized tools have also been developed to interactively

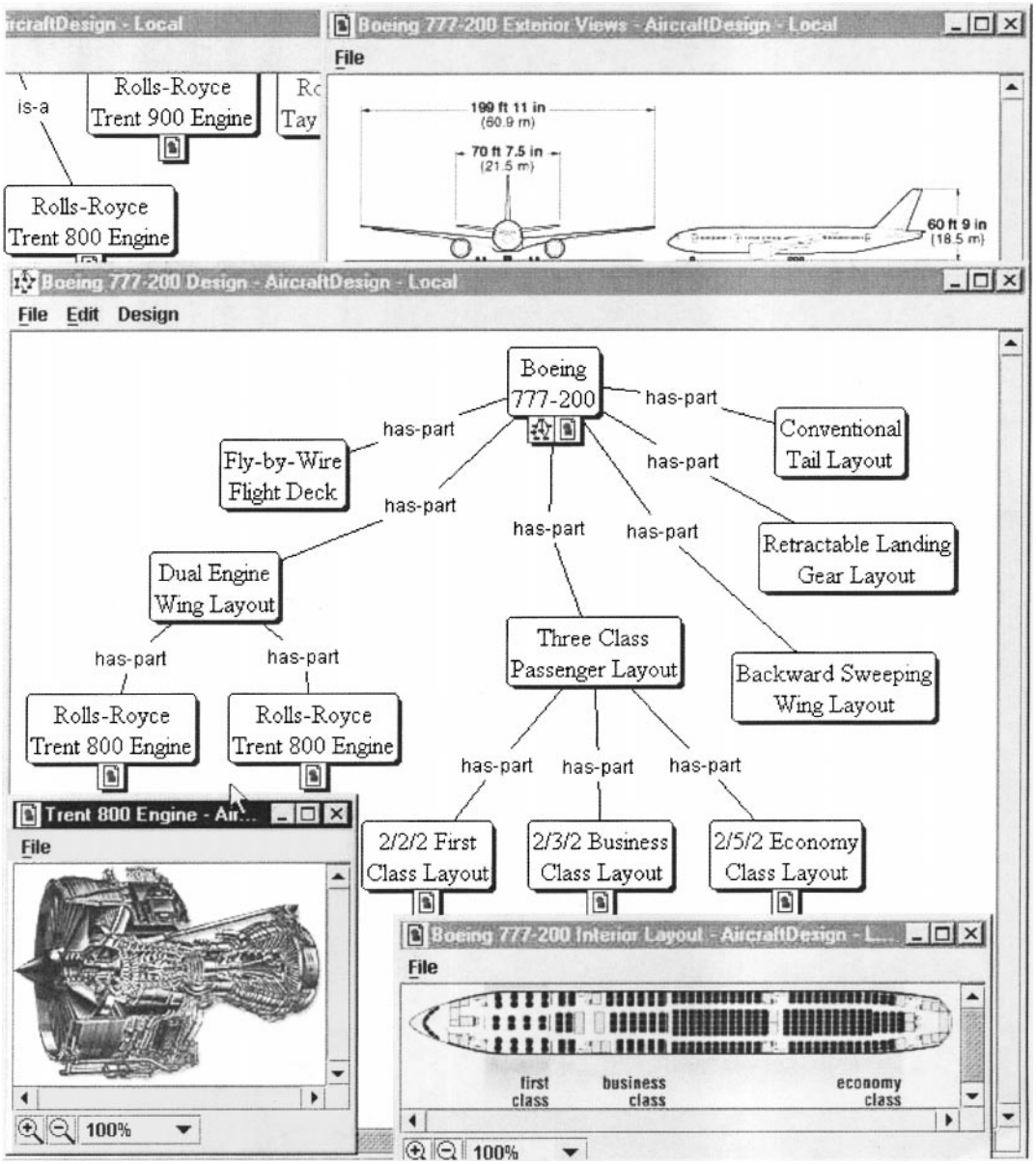

Figure 1. Sample screen images produced by the CMap tools. Exterior and interior aircraft views copyright The Boeing Company and reproduced with permission. Engine image copyright Rolls-Royce plc and reproduced with permission.

support the concept mapping process. The DRAMA project uses a set of Java-based tools developed at the Institute for Human and Machine Cognition of the University of West Florida. These tools support the interactive definition and arrangement of initial maps (including multimedia components), the hierarchical examination of those maps, and transparent sharing and navigation of maps on remote servers anywhere on the Internet. The CMap tools are publicly available on the world-wide web at http://cmap.coginst.uwf.edu/.

5. Benefits of Integrating CMaps and CBR

The integration of CBR with interactive CMap tools provides leverage for both the CBR and CMap systems. Existing CMap tools provide an interactive medium

for representing and browsing designs, but their framework does not provide facilities for automated searching for relevant stored CMaps. Likewise, although the tools provide capabilities for interactively defining new CMaps and manipulating their structure by adding, deleting, or substituting components, the tools provide no support for decision-making required by that adaptation process. Consequently, their usefulness can be extended by the addition of automatic facilities for retrieving relevant CMaps, automated aids to navigating CMaps and finding relevant information, and by aids to the reuse of prior CMaps.

Conversely, case-based reasoning can leverage off the interactive case definition and revision capabilities of the CMap tools. The CMap tools provide a convenient method for entering case information in an intermediate form between textual descriptions (which are easy to input but hard to reason about) and rich structured representations (which are hard to input but support complex reasoning). In our domain, the push to use concept mapping to understand the design process is expected to make such cases available at low cost as "seed cases" for the CBR system. In addition, the tools already provide crucial functions for interactively generating and examining these cases and navigating their hierarchical structure.

6. The DRAMA System

The DRAMA system uses concept mapping tools as a method for initial capture, manual browsing, and manual modification of design cases represented as concept maps. It uses interactive CBR techniques to retrieve relevant prior cases and to retrieve alternatives to support adaptation. In addition, it uses CBR to manage and present cases recording the rationale for particular decisions and cases suggesting adaptations of designs. The following sections discuss the main features of the system.

6.1. Using CMaps to Organize and Represent Design Information

In DRAMA, CMaps are used to represent two types of information. First, they represent hierarchies of aircraft and part types. This information is used to organize specific design cases and to guide similarity assessment during case retrieval, providing the designer with browsable hierarchies of aircraft (e.g., commercial aircraft), aircraft components (e.g., specific wings, engines, fuel tanks), and component configurations (e.g., fuel tanks inside or outside the aircraft) for reference during the design process. Our work makes no commitment to a particular set of taxonomies for a given domain. Instead it provides the tools to help particular design groups to interactively generate and refine their own sets.

Second, CMaps represent specific information about particular designs such as their components and component relationships. Each component may be represented as another CMap, enabling viewing and treating hierarchical designs at different levels of granularity through an interactive navigation process.

6.2. How the System Supports Design

To illustrate the design process, the following sections present a simple example involving the coarse-grained configuration of an airliner after an initial set of "seed case" designs has been provided to the system, along with hierarchies of aircraft types organizing those designs. The steps described include retrieval of a similar prior design as a starting point, retrieval support for adaptation and refinement of system suggestions, and the capture of a new adaptation for future use. Figure 2 summarizes these steps.

6.2.1. Retrieving a Relevant Prior Design. The design process begins by selecting a similar example as a starting point. The user may choose either of two

1. Retrieve prior design
   - CMap browsing
   - Manual feature-based case retrieval
2. Select working design to adapt
   - Null design (design from scratch)
   - The prior design
   - A copy of the prior design
   - A schema abstracted from the prior design
3. While working design $\neq$ target,
   (a) Select design component to adapt
   (b) Adapt component
      - Edit working component manually
      - Substitute prior component and adapt
        i. Retrieve substitute component
           - CMap browsing
           - Automatic context-based retrieval
        ii. Revise sub-parts of retrieved component recursively, starting at step 3b with the substitute component.

Figure 2. Steps in DRAMA's case-based design generation. Enumerated points are sequential; bullets are mutually exclusive choices within steps.

interfaces for the initial search process, one non-interactive and the other interactive. The first (non-interactive) option, the "Design Finder," is a simple and traditional CBR retrieval interface. The interface presents menus for selecting the desired features of a design from a pre-defined set of standard attribute types (e.g., aircraft type, manufacturer, model number). Currently the system uses a standard pre-defined feature set, but features could also be derived automatically from the set of designs.

Given the list of features, the designer selects any features of interest and the system performs nearest-neighbor retrieval, according to a predefined feature weighting scheme, to retrieve references to potentially-relevant CMaps. These are presented to the designer with a match score; the designer can browse and select from the alternatives.

The second interface allows the designer to navigate the set of concept maps providing alternative "views" of aircraft and aircraft component types. This is an interactive process. As the designer proceeds through these hierarchical maps, being presented with increasingly-specific alternatives, the designer interactively determines how to proceed at each level.

In our sample scenario, the designer is considering alternatives for developing a highly fuel efficient airliner. The first step is to establish a context for the design by locating the CMap for an aircraft similar to the one envisioned. The designer then selects the engine on the CMap as the part to adapt. If no CMap is already present for the component in the starting design (e.g., the designer wishes to fill in a sketchy design by specifying its engine), the designer can use the interactive CMap tools to create a new CMap from scratch, or can browse the CMaps for designs, import a design, and then adapt as desired.

When the previous case has been retrieved, the designer has four choices, as shown in Fig. 3: to *adapt* it (changing the representation in memory, e.g., when continuing work on a design begun in a previous session); to *derive* a new design, by having the system make a copy to adapt; to ask the system to use its hierarchy of aircraft parts to form an abstraction of the current design's structure as a template to fill in; or to ignore the design and begin a new design from scratch.

6.2.2. Adapting Designs.

Retrieving Designs to Understand or Suggest New Components. Once the designer has navigated, for example, to the engine of a particular aircraft, the system supports three ways of examining why the engine

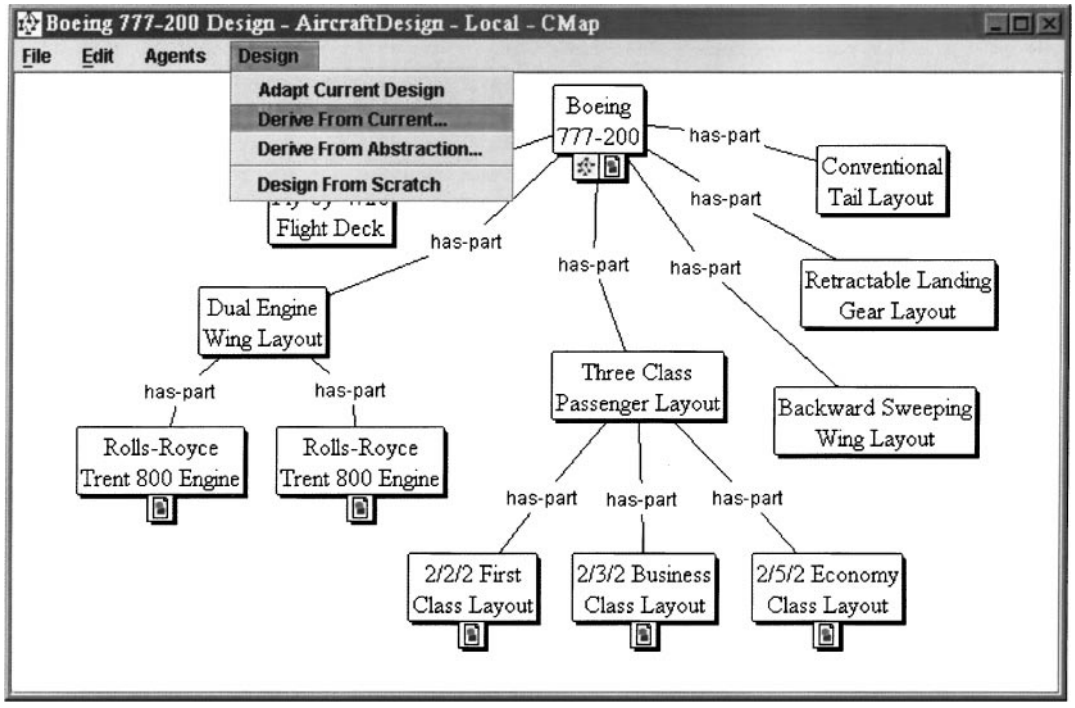

Figure 3. Beginning derivation of a new design from a prior case.

was used and the alternatives that may exist. First, the designer may simply interactively browse stored information, following links in the CMap to examine associated information such as finer-grained concept maps, video clips of explanations from previous designers, or photographs or specifications of the engine.

Second, the designer may request information about similar designs. The designer may request to have this retrieval targeted to either:

- Focus on designs with components similar to the one that is currently of interest (e.g., CMaps that show aircraft using similar engines)
- Focus on designs that provide similar contexts for the current type of component (e.g., CMaps that show the engines of similar aircraft)

The algorithms underlying this retrieval are described in Section 7. The interface for presenting this information is shown in Fig. 4.

Retrieved alternatives are listed in order of goodness of match according to the chosen focus. The designer may also enter additional criteria to be matched against any textual annotations of rationale recorded by previous designers. For example, the designer may request that fuel-efficient engines be weighted more heavily, as shown in the bottom portion of Fig. 4. This revision uses simple text matching techniques from information retrieval (e.g., [23]) to decide which prior rationale to consider most relevant.

Suggesting Prior Adaptations. When the designer selects a component of an aircraft to adapt, the system has access to three pieces of information: the component affected, any designer input of additional retrieval criteria, and the design itself. This information is used to index into stored records of prior adaptations, in order to suggest proven adaptations. If adaptations have been previously performed in similar contexts to address similar issues, those adaptations are highlighted in the list of alternatives. Note that this adaptation process does not assume knowledge of complex constraints. DRAMA's method reduces the amount of knowledge needed, but at the cost of requiring the designer to evaluate the possibilities suggested (cf. [18]).

Performing Adaptations. When replacing an engine, the designer may select any of the suggested engines to browse further, or to substitute for the engine in the design. The designer may also simply delete or add a component to the representation using the CMap tools.

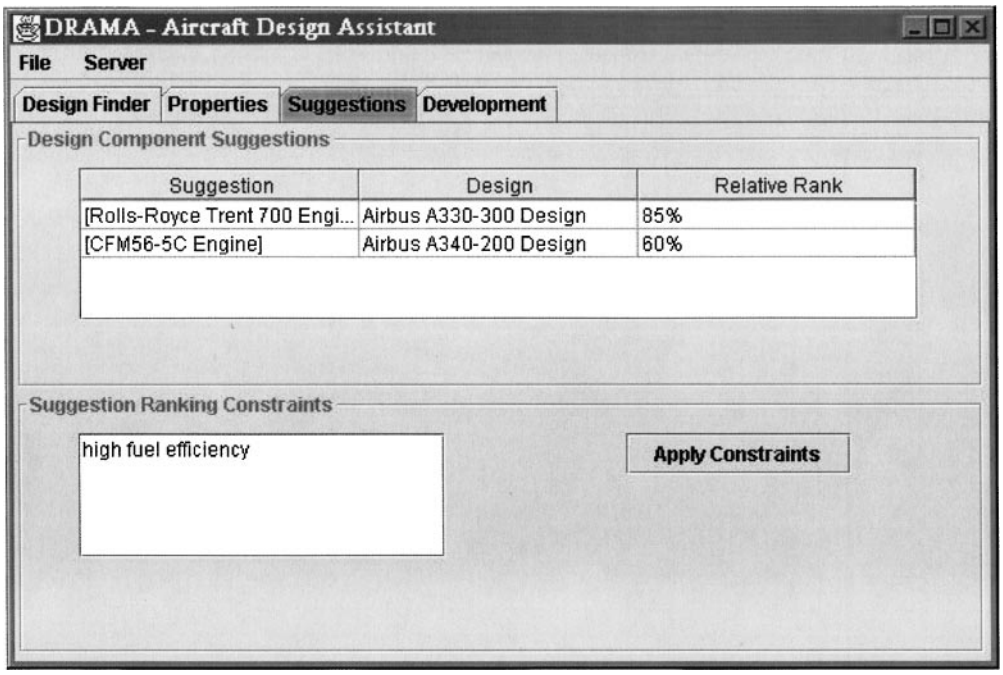

Figure 4. Interface for presenting engine suggestions and entering additional constraints to refine ordering by matching against textual rationale information.

Adaptations of concept maps can be thought of as falling into three general categories corresponding to the support that they require: additions, deletions, and substitutions. Our framework supports the designer's performance of these operations as follows:

- Additions: The designer may use the plain-text retrieval capability to retrieve potentially-relevant components to be linked into the design.
- Deletions: The system can warn of potential deletion issues by proactively retrieving similar deletions, checking them for problems, and presenting those problems to the designer.
- Substitutions: The system can support substitution by retrieving and suggesting candidate substitutions, using both the explicitly-stated criteria and contextual information from the current map to guide the retrieval. It retrieves these from two sources: from stored adaptation cases encapsulating prior substitutions, and from analogous nodes in similar designs. Section 8 discusses experiments examining this support process.

When the designer states a goal and finds a suitable substitution, the system learns "adaptation cases" packaging the query, information about the CMap that was used as context for the search, and the selected result, following research on case-based adaptation learning [24, 25]. We are still refining methods for these processes.

6.2.3. Storing Rationale. After the designer performs a substitution, the designer is prompted to enter an optional textual annotation of why the new alternative is preferable to the old (e.g., an engine might have been replaced to increase fuel efficiency). Asking the designer to address this question focuses rationale capture: The designer does not record a rationale for the component as a whole (which could involve countless factors), but simply for why it is *better* than another component in the current context. Focusing the explanation process in this way is related to the common CBR idea of generating expectations for behavior and explaining only deviations from those expectations (e.g., [26–28]). During future adaptations, this rationale will be provided with other information about the component, and will be used as an additional index when retrieving possible substitutions.

6.2.4. Storing Generated Cases. Adapted cases are placed into the system's hierarchies of cases at the point where the designer found the most similar previous case.

7. DRAMA's Method for Retrievals to Support Adaptations

Suggesting new components during adaptation requires comparing the current concept map to those in memory. Given a user-selected component (e.g., a particular engine) to be adapted, and given the goal of finding other engines from similar designs, DRAMA first retrieves a set of similar designs. The similarity assessment process begins by establishing correspondences between nodes in the old and new designs by matching their node labels. This process is equivalent to finding a matching between roles in role-filler representations. When the role appears in both old and new designs (e.g., both have a feature labeled "engine"), it is trivial to establish this correspondence. However, because concept maps are not standardized, there is no guarantee that equivalent roles will have identical labels. After pairing all the nodes which have identical labels, the matcher pairs the remaining nodes (roles) of the new design with the remaining nodes of the old design, trying to pair the roles whose fillers are most similar. For example, the "engine" role of one map might be paired with the "propulsion" role of another, because both roles are filled with engines. The matcher uses a greedy algorithm to establish this pairing while (ideally) minimizing the sum of the distances between each pair of corresponding role-fillers. In the current implementation, left-over features are ignored. Although matching cost is a potential issue, it has not proven a problem in current tests, and we have not attempted to optimize this process.

After the roles of the old and new CMaps are matched, the distance between the two CMaps is calculated by a weighted nearest-neighbor algorithm applied to the pairs of role-fillers. The distance between each pair of role-fillers is the number of links that must be traversed from one to the other in the design component abstraction hierarchy, normalized by the maximum possible distance in the hierarchy.

After retrieving concept maps for similar designs, DRAMA collects potential substitute components from each of the similar designs. The results are ranked by the inverse of each design map's distance from the current context. This gives an indication of the relative goodness of each suggestion within the overall pool of suggestions.

Once candidate concepts have been retrieved and displayed, the user can adjust the relative ranking by entering textual descriptions of desired properties—"focus features"—to compare with the properties annotating the component. The entered text is matched against any textual annotations of a suggested component; the degree of match is calculated by counting term matches in the text and normalizing this sum by the frequencies with which the term is used in all annotations. The resulting value is taken as an additional feature added into the weighted sum for design distance.

The Significance of the Approach. Ideally, case retrieval should reflect both high-level goals and concrete design features. Applied CBR systems tend to rely on the user to explicitly provide this information (whether all at once or incrementally). One method for guiding the process is to prompt the user for information, using strategically organized questions in order to minimize the number of questions that must be asked to reach a solution. This is one of the principles of successful conversational case-based reasoning, and automated methods are now proposed to improve question organization [29]. In some applications contexts, however, it may be quite difficult to craft these questions. Another approach to improving retrievals is for the system to learn about the importance of particular types of features to guide its future retrievals (e.g., [30, 31]).

The research focus of DRAMA's retrieval is on how to automatically generate feature values from the task context. Its aim is to integrate the CBR process tightly enough into the user's task process to infer a substantial part of the needed contextual features directly from monitoring the user's task, saving the user the effort of query formulation. This approach is closely related to research on proactive case retrieval [32], and we believe it will become increasingly important in fielded CBR systems.

Because of the close coupling between the CBR system and the user's task interface, the system has access not only to the user's retrieval request (e.g., to find a substitute engine), but also to a significant part of the context surrounding the request that will determine the relevance of the retrieval (e.g., the aircraft for which the engine is needed). The designer may augment this context with information that is not available from the current step in the design process (e.g., that the goal is to find a more fuel-efficient engine that could substitute), but is not required to do so. This approach raises a number of questions about efficiency and performance, such as what levels of performance can be achieved early in the adaptation process, when the available context may differ significantly from the final design for which a component is needed. We examine such questions in the following section.

8. Testing DRAMA's Context-Based Retrieval

We performed experiments comparing DRAMA's context-based retrieval to two baseline manual retrieval methods, for the task of finding replacement aircraft engines. This tests the benefits of DRAMA's method for performing automatic context-based retrieval (step 3(b)ii of Fig. 2).

A set of test retrieval problems and a set of retrieval targets were generated, as described in Section 8.1. The experiments compared DRAMA's retrieval to the two baseline methods according to two criteria. The first was retrieval quality, measured by the distance of the actual retrieval from a target case to retrieve, according to a nearest-neighbor distance metric (e.g., [5]). The second was the amount of user effort they required, measured by the number of design questions (features) that the user needed to answer (specify) in order to retrieve the target.

In the first baseline method, manual *engine-based* retrieval, an engine is retrieved by directly specifying a set of engine features. A user interactively specifies desired engine features, with the best-matching engine returned for each new feature set (ties are broken arbitrarily), until a target engine (the engine in the case library that best matches the engine in the target design) is retrieved or until all feature values are specified (see Fig. 5). In the second baseline method, manual *aircraft-based* retrieval, a user interactively specifies desired aircraft features to retrieve an aircraft similar to the target design, and the result is the engine of the retrieved aircraft.

```
Initialize Q&A list
Repeat
        Select unanswered question
        Select answer based on target
        Update Q&A list
        Candidate-design =
            Retrieve-NN(Q&A list, case-base)
Until   {(Candidate-design == target) or
            (no unanswered questions)}
Return candidate-design
```

Figure 5. Steps in the manual retrieval process.

Our experiments examine how the dependent variables of quality and efficiency are affected by four independent variables: *retrieval method* (manual engine-based, manual aircraft-based, or DRAMA's method), *user expertise* (for manual methods, this determines the user's ability to select accurate and discriminating features and appropriate values to match the target design), *point in design process* (for DRAMA's method, this determines how well the available design context approximates the target design), and the *magnitude of changes* required to transform the starting point into the new design.

8.1. Generating Problems and Retrieval Targets

Test problems were generated by a problem generator that randomly selects a set of aircraft design cases from the case library as starting points, each to be adapted into a new design. A target aircraft design is generated by applying hand-coded perturbation rules to the features of the selected initial design. These rules (1) adjust randomly-selected features of the aircraft design, and (2) adjust related aircraft and engine features to assure consistency of the overall design. For example, a change to an aircraft that increases passenger capacity should increase aircraft weight as well, and an increase in aircraft weight may require increasing the thrust of the engine. The rules propagate effects of design changes to produce a consistent target design. The number of independent feature changes and the magnitude of the changes are parameters of the problem generator.

The design generation process results in a new aircraft design as a whole, including desired features for the engine. In our experiments, these engine features are taken as the ideal features that the simulated designer is attempting to approximate by retrieving the closest available engine. Consequently, to select the target engine that will be used as the standard for judging retrieval quality, the problem generator uses the list of desired features to select the closest actual engine from DRAMA's case library of known engines.

8.2. Simulating the Manual Retrieval Processes

In many interactive CBR systems, the user interacts with the case retrieval process by repeatedly selecting and answering questions from a list of candidates. These questions determine the feature values of the problem being described, which in turn are the basis for retrievals. Each time a question is answered, the system retrieves and presents the cases that best match the current case description (e.g., [29]). In our experiments, this interactive retrieval process is simulated by a program that plays the role of the user, repeatedly selecting questions (unspecified features whose values need to be filled in), selecting feature values to answer them, and using the feature values in a nearest-neighbor retrieval process to retrieve a candidate design from the case library. This process is summarized in Fig. 5. In the simulated user, answers are selected based on the features of the target aircraft, because the target aircraft corresponds to the conception of the design that the simulated designer is attempting to achieve. As each question is answered, the manual retrieval method returns the closest case in memory according to a nearest-neighbor retrieval algorithm. Because we assume that the user can identify whether the retrieved engine has the desired properties, the simulated user answers additional questions until either the target engine is selected (a user success) or until values are specified for all features, in which case the last candidate case is returned as the result.

Question selection is based on a simple model of how expertise levels affect question choices and the order in which the user chooses to answer different questions. Based on a parameter for expertise level, the simulator adjusts the probability that the next question answered by the simulated user will be the maximally-discriminating unanswered question, according to a manual ordering of the expected discriminating power for the available features. Specific questions are selected by sequentially passing over the list of unanswered questions about a particular aircraft or engine, ordered by expected discrimination power. The probability of choosing the next question in the list is given by the user's expertise level, and candidate questions are visited in order until one is chosen or the list is exhausted. If no question is chosen, an unanswered question is selected at random. Once the question is chosen, the probability that the user's answer to the question matches the target is again determined by the expertise level of the user. If the simulated user does not answer the question exactly, the answer is chosen from a distribution about the target value, based on the user's expertise level. Thus as expertise increases the possible choice distribution converges towards the target value.

Modeling the Effects of Changing Retrieval Context. The experiments simulate retrievals during an ongoing

design process. During this process, an initial design is being adapted into a design that approximates the target. We assume that the simulated designer may adapt components of the initial design in any order; in particular, the designer may choose to adapt the initial engine—and, consequently, to retrieve the new substitute engine—at any point in the design process. This affects the quality of context that the current design provides to DRAMA's retrieval method. If the replacement engine is retrieved early, the current design will be close to the initial design, which may be quite different from the target design. On the other hand, if the replacement engine is retrieved late in the design process, the current design providing context for DRAMA's retrieval will be very close to the target design.

To test how the quality of DRAMA's retrieval depends on when retrieval occurs, our experiments generate a sequence of design steps starting with the initial design and ending with the target, reflecting the context available at each point when an engine may be retrieved. Retrievals are done at different points in this sequence. When each retrieval is performed, the simulated user randomly selects one of the target engine's features (e.g., high fuel efficiency) as a "focus feature" to be added to DRAMA's automatically generated retrieval features. This is the only feature that the user must specify during DRAMA's retrieval, so user effort is constant.

8.3. Experimental Setup

Tests were performed using an aircraft case library consisting of 62 cases. Each aircraft had 20 features and each engine had 12 features. Twelve of the cases were manually-entered designs representing different types of real jet aircraft (six passenger airliners ranging from small commuters to jumbo jets, two military heavy cargo planes two military reconnaissance planes, and two military fighters) and their engines. Fifty additional designs were derived from the original twelve using the perturbation rules from the problem generator. Experiments were repeated for varying perturbation levels (numbers of changed features) and perturbation degrees (magnitude of feature-value changes). The test set consisted of 10 randomly-selected initial designs and derived targets. For each test problem, retrieval method, and level of expertise, the retrieval process was performed 10 times. Results were averaged over these 10 trials, and then averaged over the set of problems, to obtain the final reported result in each condition. Nearest-neighbor distances were normalized to values between 0 and 1.

8.4. Experimental Results

Retrieval Quality. Our predictions for comparative performance were based on expectations about the quality and types of information available to each method to guide its retrievals. We predicted that the retrieval quality achieved by expert users with manual engine-based retrieval methods would surpass DRAMA's method, because expert users should be able to choose a focused and accurate engine-feature context directly related to the engine type. Our expectation was that DRAMA would achieve similar or somewhat better quality than expert users with aircraft-based manual methods. Experts might make better initial choices for questions to answer to retrieve relevant aircraft, but DRAMA's method has access to an additional "focus feature" to provide additional guidance about an important engine feature, and this information is not considered in aircraft-based retrieval. We also expected that DRAMA would provide better solutions than novice users: DRAMA uses all of the present available design context, which should provide a reasonably good starting point for retrieval, whereas novice users may choose features and values that comprise an inaccurate retrieval context for the desired target. Because the quality of context available to DRAMA increases as the design approaches completion, we also expected that DRAMA's retrieval quality would depend on the point in the design process at which retrievals were done, being lowest early in the design process and increasing as the design process progressed.

Figure 6 shows representative results for 3 different points in the design process for changes of random magnitude. As expected, there are crossover points determined by expertise, below which DRAMA's retrieval outperforms manual engine-based retrieval, and above which it performs worse. DRAMA's retrieval also matches or outperforms aircraft-based retrieval methods except for very low-magnitude changes in the designs. This matches expectations for most changes, for which information about the focus feature may help DRAMA discriminate between similar aircraft to select an appropriate engine. At very low-magnitude changes, the new design is sufficiently similar to the old design that we expect the marginal benefit of the extra information available to DRAMA's method—the focus feature—to be small or nonexistent. In addition,

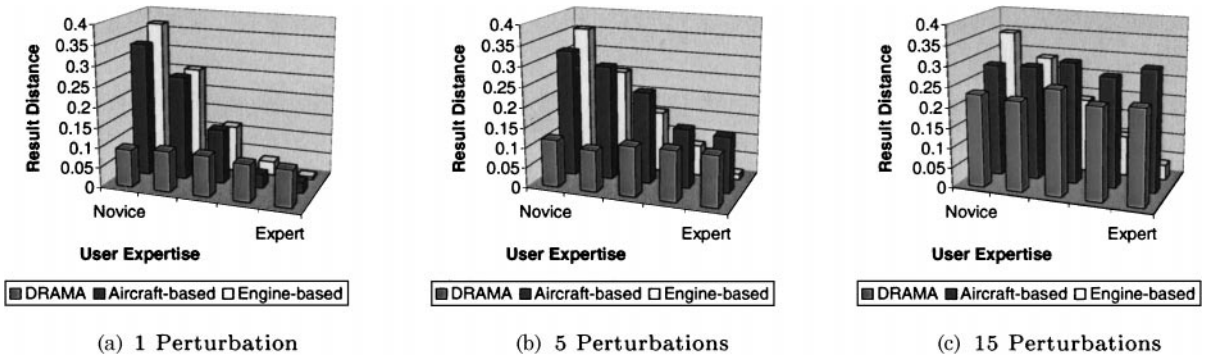

Figure 6. Retrieval quality for 3 sets of perturbations of random magnitudes.

because DRAMA's retrievals are based on the current state of the design, which differs from the anticipated final design (which determines the expert's choice of features), the DRAMA method will be expected to show some limited error based on that difference, in all conditions. DRAMA's retrieval quality may be impaired by using the current design state to define retrieval features, rather than having the expert's knowledge of the target design to choose the right features to specify.

Efficiency. The experiments also provide information on the expected efficiency of manual retrievals for this task (measured in terms of number of questions that have to be answered to retrieve the target engine), and hence on the comparative benefits of DRAMA's method, which requires the user to specify only one focus feature. Figure 7 shows representative efficiency results for three perturbation levels of random degree. Recall that in these tests, it is assumed that the designer can recognize the target engine as the best solution, and will continue specifying retrieval features until either that engine is selected or all features have been specified.

The relative difference between the number of questions that must be answered for aircraft- and engine-based retrieval is due to the different total numbers of aircraft and engine features (20 vs. 12). Based on our results, even experts are expected to specify a substantial number of features, although they achieve high quality retrievals with those choices. Novice users must specify many features and still achieve relatively low-quality retrievals. Because DRAMA requires only the specification of a single focus feature, DRAMA's method provides a significant reduction in the effort required to build a retrieval context while retaining reasonable quality.

We expected that increased expertise would decrease the number of features required for the manual methods, as can be seen in Fig. 7, graphs (*a*) and (*b*). In some tests, however, expertise had little effect on reducing the number of features required. In these examples, individual features were generally not highly discriminating; we are investigating the reasons for this trend.

Point in the Design Process. We predicted that the greatest benefits from DRAMA's retrieval method

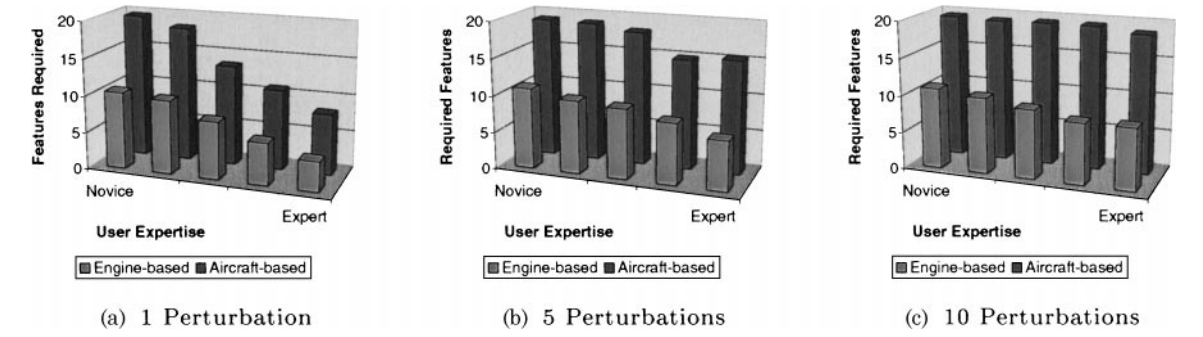

Figure 7. Efficiency for 3 sets of perturbations of random magnitudes.

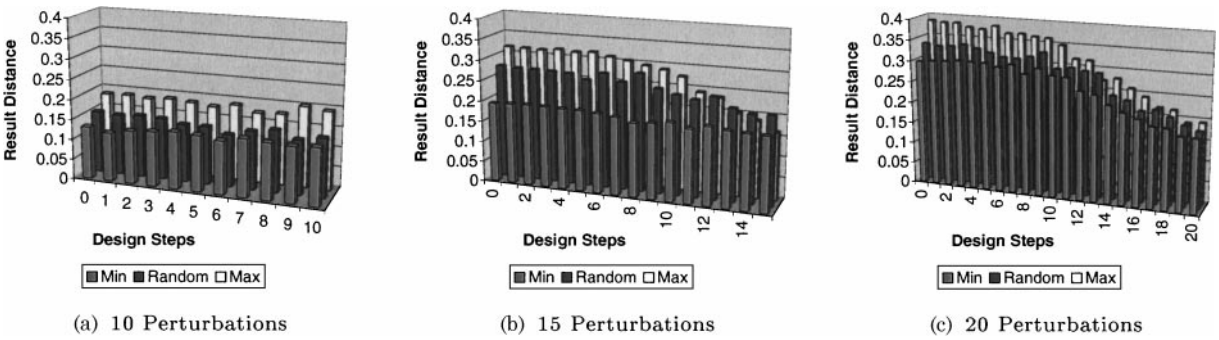

Figure 8. Retrieval quality at successive steps in the design process for 3 sets of perturbations.

would come from designs that were closest to completion, and that importance of closeness to completion would increase with the magnitude of the perturbations performed to generate the target design. Figure 8 shows the average distances of DRAMA's retrievals from the targets at design stages for minimal, random, and maximal perturbation magnitudes for three representative total numbers of perturbations. The retrieval benefits from closeness to completion are apparent in two ways. First, distances of retrievals from the target increase for targets generated using more perturbations, because the starting point of the design context is further away from the target. Second, once the context has been narrowed enough by the design approaching completion, a marked decrease in distance can be seen.

At low perturbation magnitudes, there is enough context that even moderate numbers of perturbations in generating the target do not significantly affect retrieval, so distance from the target is consistent. With higher perturbation magnitudes, after most of the perturbations in a design have been performed, the design context is sufficient for general guidance, with reasonable distances from the target, but is not specific enough to make fine distinctions. When a critical context level is achieved (here when the designs are approximately half completed), finer distinctions are made with each added piece of context, and marked improvements are seen in retrieval distance. Figure 8 shows that the magnitude of changes affects the importance of distance in that minimal magnitude perturbations provide a constant set of low values, while random and maximum show the reduction effect.

The three sets of experiments suggest that DRAMA's context-based retrieval can provide good suggestions with considerable savings in effort and with quality advantages over manual methods, depending on the quality of context available and on the designer's expertise in manually selecting retrieval features.

9. Perspective on Issues and Methods

The key point of the previous section is DRAMA's capability to use the context of a current design to make useful suggestions about replacement components during the design process. DRAMA's approach is also relevant to a number of fundamental issues for developing practical case-based applications. This section briefly summarizes how the project relates to other research in those areas.

9.1. Interactive Case Acquisition

CBR is often seen as a means for overcoming the classic knowledge engineering problems associated with rule-based systems. However, successfully deploying CBR may require significant "case engineering" [6, 29, 33–35]. Traditional structured case representations enable powerful processing at the cost of considerable case generation effort; research in textual CBR attempts to alleviate this problem by capturing information in textual form [36], but must overcome problems in processing unstructured case information. Concept map representations are at a middle level between these two approaches. They include structural information, but do not enforce a standard syntax or standard set of attributes. This makes them more difficult for a CBR system to manipulate autonomously than conventional cases, but also more manageable than pure textual information. In addition, we expect this representation to facilitate non-experts in AI encoding their knowledge and to enable users to devise their own representations as needed. We have just begun to study the tradeoffs

involved in this level of representation and how to address the potential problems.

For example, when cases are represented in a non-uniform way it may be difficult to identify relevant cases to retrieve. DRAMA uses two methods to help standardize CMap representations without sacrificing flexibility for the designer. First, when a user generates a CMap and is about to fill in a new link or node, the system presents the user with a menu of alternatives from previous maps. If one of these is suitable, the user selects it. This builds up a set of standard types over time. Second, the baseline process for generating new design CMaps is modification of previous designs. The system is intended to begin with a set of CMaps that reflect the conceptualizations of a particular expert designer, reflecting that designer's coherent view of the factors important in a design. When new designs are generated by adaptation, significant portions of old representations are brought to new tasks, encouraging—but not requiring—the use of representations with similar structure. In this way, the case library and case representations are built in parallel. Practical use will give an indication of the adequacy of these methods and the overall quality of retrievals. By changing representations as well as the knowledge contained in the case library, this approach may also present new challenges for case-base maintenance (see [37] for a survey of case-base maintenance issues and projects).

9.2. Guiding Design Rationale Capture

Rationale capture has long been an important topic in AI for design. A number of projects have applied rule-based or model-based approaches to the problem of design rationale capture. However, explicitly encoding all the factors relevant to a design can be prohibitively expensive, as can maintaining the required knowledge as design problems change. Consequently, it is appealing to use machine learning techniques to build up and refine design rationale knowledge incrementally.

Because CMap design cases *already* capture an entire design situation as context, we believe that useful rationale capture can be achieved with fairly limited additional information: an annotation about why the designer chose a particular component, *given the implicit context of the previous components chosen*. Although to our knowledge, this specific approach to capturing design rationale is novel, the information it captures corresponds to the "weak explanations" advocated by Gruber and Russell [38] in providing just enough information to guide a designer's own reasoning process. DRAMA stores this information in an unanalyzed textual form to be compared by textual matching.

Automatic capture of detailed reasoning traces has been studied in CBR research on *derivational analogy*, which captures and replays traces of the decision-making process of another knowledge-based system [39–41]. Recent research has begun to apply this approach to interactive capture and replay of planning rationale to support human planners [42] and for guiding search for information [43]. Our project builds on these foundations but addresses two new issues. The first is how to capture and provide useful information when it is not practical to obtain a full derivational trace. The second is how to maintain and apply case libraries from multiple designers who may have different conceptualizations of the problem and domain.

In a related spirit, but storing much more structured rationale information than DRAMA, is Clark's [44] model of argumentation applied to geological appraisal. In that model, decisions are annotated with structured arguments (which could be represented by CMaps) aimed at helping experts to construct and improve their risk assessments. Their argumentation is a form of cooperative knowledge sharing, just as DRAMA is intended to enable designers to express, explain, and share their conceptualizations of designs.

9.3. Conversational CBR

Conversational case-based reasoning (CCBR) systems guide the retrieval process through an interactive dialogue of questions. As described in [29], this approach has gained widespread acceptance in CBR shells for help desk applications; it provides both flexibility in the order of information presentation and useful incremental feedback on promising alternative cases. However, because poor questions or question organization may prevent retrieval or slow identification of the right cases, a substantial case engineering effort may be required to craft the set of questions involved, in some cases preventing the method from being applied at all.

Like CCBR, the DRAMA approach is in the spirit of interactive retrieval, but under different constraints. The prior analysis required to craft carefully targeted questions for designs would be rejected by the intended end users of DRAMA, and, at least for expert designers, the situations for which examination of prior cases is most useful are likely to be hard to anticipate. On

the other hand, unlike CCBR systems used in a help desk context, for which it is necessary to build up a picture of the caller's situation, DRAMA can benefit from "free" contextual information: Because DRAMA is used as a design environment as well as a retrieval tool, it has considerable knowledge of the basic task context without the need of addressing any queries to the user.

Both DRAMA, in its initial retrieval phase, and CCBR systems are aimed at retrieving the most appropriate complete solution from previous cases. However, in its retrieval to support adaptation, DRAMA provides the ability to perform retrievals focused on subparts of the problem for the user to compose. As the user adapts part of the design, the retrieval context changes automatically, loosely corresponding to CCBR systems' adjusted rankings as more information becomes available.

9.4. Knowledge Navigation

Case-based reasoning principles have been applied to aiding navigation and guiding interface design for web-based information sources (e.g., [45, 46]), as well as for capturing and reusing knowledge about how to satisfy information needs using other information sources such as corporate databases [47] and hierarchical memories [48, 49]. All this work is relevant to the storage and reuse of information about how to navigate through a case memory of concept maps. The capability for a user to navigate directly through a hierarchy of cases, as in DRAMA's retrieval by browsing a CMap hierarchy, has recently been investigated in the HOMER system [50]. Interactive navigation through linked networks of cases has also been extensively studied in the context of ASK systems [51], structured hypermedia systems in which users navigate through explanatory conversations by selecting alternatives from a carefully-selected set of predefined relevant links. In contrast, the CMap framework described in this article focuses on retrieval when appropriate links are difficult to anticipate a priori.

10. Future Directions

Development of the DRAMA system is an ongoing project. The CMap tools are being used to define richer case bases to test the system in the context of a design project for the next generation of reusable spacecraft. The concrete experience from this test will provide feedback and data to adjust details of the interface, functionality, and indexing algorithms. It will also provide data for conducting controlled tests of the quality of recommendations provided by the system.

Because the CMap tools provide the capability to share CMaps across the World Wide Web, designs from multiple designers and sites can be imported into the system's design process. Work is under way at the University of West Florida to develop CMap facilities for managing concurrent CMap generation and modification. Ideally, the design context for a particular engine, for example, could be updated as designers at other sites make changes in its specifications.

In addition to refining the system as an aid to recording and reusing design information, we see a long-term opportunity to apply it to reuse of information about *design processes*. A CMap-style interface could be used to capture traces of the steps used in generating a design (e.g., conceptual design, specification, numerical simulations, etc.), to capture how a design was formulated and to guide reasoning throughout the design process.

11. Conclusions: Summary and Future Issues

The DRAMA project investigates an integrated interactive approach to case-based design support. This approach is motivated by the complexity of aerospace design, which is such that autonomous intelligent design tools are currently infeasible, but tools that interactively support and aid the designer have promising potential.

This article has described a knowledge-light framework for supporting human aerospace designers by capturing and reusing cases representing knowledge about specific prior designs, about circumstances in which those designs were useful, and about how those designs were adapted to prior needs. It has also presented experimental results providing predictions of the benefit of the system's retrieval mechanism to support adaptation for different classes of users, for different levels of innovation, and at different levels of design completeness.

The knowledge-light framework necessarily involves tradeoffs; for example, the system can suggest components but cannot evaluate the user's designs or adaptations. Future research will investigate providing warnings when new designs match prior design cases that have known problems.

In developing the initial version of DRAMA we have identified principles that we expect to have wide applicability to CBR integrations into interactive systems:

- Representations should be easily comprehensible and interactively adaptable by end users; visual representations may be especially useful.
- Support for representation generation should help assure consistent representations, but must not prevent the user from developing new representational elements when needed. CBR's "retrieve and adapt" process to build new cases can facilitate standardization by reusing prior representational components. This can naturally build up the case library and the representational vocabulary in parallel.
- Retrieval must tolerate representational discrepancies.
- Interactive support systems must be sufficiently integrated into the processes they support to be able to unobtrusively monitor and proactively exploit information about the task context.

The strongest overall conclusion is that interaction must be across all parts of the CBR system—initial knowledge capture, representation, retrieval, and adaptation—and across the larger task. Frameworks that allow the user and system to support each other in a shared task context, building up and using shared knowledge, have the potential to leverage off the strengths and alleviate the weaknesses of both system and user. Developing these frameworks is a challenging but promising research area.

Acknowledgments

This work has been supported in part by NASA under award No NCC 2-1035. We thank David Aha and Héctor Muñoz-Avila for their many helpful comments on a draft of this paper, Alberto Cañas and the Concept Mapping group at the University of West Florida for their generous assistance with the CMap tools, and Jim Newkirk for his contributions to the implementation of the DRAMA system. We also appreciate the valuable assistance of Kenneth Ford, Mary Livingston, and the Advanced Design Technology Testbed group at NASA Ames Research Center. Portions of this paper were adapted from [52].

References

1. A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *AI Communications*, vol. 7, no. 1, pp. 39–52, 1994.
2. J. Kolodner, *Case-Based Reasoning*, Morgan Kaufmann: San Mateo, CA, 1993.
3. D. Leake (Ed.), *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, AAAI Press: Menlo Park, CA, 1996.
4. C. Riesbeck and R. Schank, *Inside Case-Based Reasoning*, Lawrence Erlbaum: Hillsdale, NJ, 1989.
5. I. Watson, *Applying Case-Based Reasoning: Techniques for Enterprise Systems*, Morgan Kaufmann: San Mateo, CA, 1997.
6. H. Kitano and H. Shimazu, "The experience sharing architecture: A case study in corporate-wide case-based software quality control," in *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, edited by D. Leake, AAAI Press: Menlo Park, CA, pp. 235–268, 1996.
7. J. Novak and D. Gowin, *Learning How to Learn*, Cambridge University Press: New York, 1984.
8. A. Cañas, D. Leake, and D. Wilson, "Managing, mapping, and manipulating conceptual knowledge," in *Proceedings of the AAAI-99 Workshop on Exploring Synergies of Knowledge Management and Case-Based Reasoning*, AAAI Press: Menlo Park, 1999, pp. 10–14.
9. T. Bagg, *RECALL: Reusable experience with case-based reasoning for automating lessons learned*, NASA, Moffet Field, CA, 1997 (http://hope.gsfc. nasa.gov/RECALL/homepg/recall.htm)
10. W. Wilke, I. Vollrath, K.-D. Althoff, and R. Bergmann, "A framework for learning adaptation knowledge based on knowledge light approaches," in *Proceedings of the Fifth German Workshop on Case-Based Reasoning*, 1997, pp. 235–242.
11. J. Kolodner, "Improving human decision making through case-based decision aiding," *The AI Magazine*, vol. 12, no. 2, pp. 52–68, 1991.
12. M. Lenz, B. Bartsch-Spörl, H.-D. Burkhard, and S. Wess (Eds.), *Case-Based Reasoning Technology: From Foundations to Applications*, Springer: Berlin, 1998.
13. E. Domeshek, M. Herndon, A. Bennett, and J. Kolodner, "A case-based design aid for conceptual design of aircraft subsystems," in *Proceedings of the Tenth ieee Conference on Artificial Intelligence for Applications*, IEEE Computer Society Press: Washington, 1994, pp. 63–69.
14. F. Gebhardt, A. Voß, W. Gräther, and B. Schmidt-Belz, *Reasoning with Complex Cases*, Kluwer: Boston, 1997.
15. A. Goel, J. Kolodner, M. Pearce, and R. Billington, "Towards a case-based tool for aiding conceptual design problem solving," in *Proceedings of the DARPA Case-Based Reasoning Workshop*, edited by R. Bareiss, Morgan Kaufmann: San Mateo, 1991, pp. 109–120.
16. K. Hua and B. Faltings, "Exploring case-based design—CADRE," *Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, vol. 7, no. 2, pp. 135–144, 1993.
17. A.G. de Silva Garza and M. Maher, "Design by interactive exploration using memory-based techniques," *Knowledge-Based Systems*, vol. 9, no. 1, 1996.
18. I. Smith, C. Lottaz, and B. Faltings, "Spatial composition using cases: IDIOM, in *Proceedings of First International Conference on Case-Based Reasoning*, Springer Verlag: Berlin, 1995, pp. 88–97.
19. I. Vollrath, "Reuse of complex electronic designs: Requirements analysis for a CBR application," in *Proceedings of the Fourth European Workshop on Case-Based Reasoning*, edited by P. Cunningham, B. Smyth, and M. Keane, Springer Verlag: Berlin, 1998, pp. 136–147.

20. K. Sycara, R. Guttal, J. Koning, S. Narasimhan, and D. Navinchandra, "CADET: A case-based synthesis tool for engineering design," *International Journal of Expert Systems*, vol. 4, no. 2, pp. 157–188, 1991.
21. A. Cañas, K. Ford, J. Brennan, T. Reichherzer, and P. Hayes, "Knowledge construction and sharing in quorum," in *World conference on artificial intelligence in education*, Canas, Norfolk VA, pp. 218–225, 1995.
22. D. Jonassen, K. Beissner, and M. Yacci, "Explicit methods for conveying structural knowledge through concept maps," in *Structural Knowledge: Techniques for Representing, Conveying, and Acquiring Structural Knowledge*, Erlbaum: Hillsdale, NJ, p. 155, 1993.
23. G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing and Management*, vol. 24, no. 5, pp. 513–523, 1988.
24. D. Leake, A. Kinley, and D. Wilson, "A case study of case-based CBR," in *Proceedings of the Second International Conference on Case-Based Reasoning*, Springer Verlag: Berlin, 1997a, pp. 371–382.
25. K. Sycara, "Resolving adversarial conflicts: An approach integrating case-based and analytic methods," Unpublished doctoral dissertion, School of Information and Computer Science, Georgia Institute of Technology. (Georgia Institute of Technology, Technical Report GIT-ICS-87/26), 1987.
26. K. Hammond, *Case-Based Planning: Viewing Planning as a Memory Task*, Academic Press: San Diego, 1989.
27. L. Ihrig and S. Kambhampati, "Storing and indexing plan derivations through explanation-based analysis of retrieval failures," *Journal of Artificial Intelligence Research*, vol. 7, pp. 161–198, 1997.
28. R. Schank, *Dynamic Memory: A Theory of Learning in Computers and People*, Cambridge University Press: Cambridge, England, 1982.
29. D.W. Aha and L. Breslow, "Refining conversational case libraries," in *Proceedings of the Second International Conference on Case-Based Reasoning*, Springer Verlag: Berlin, 1997, pp. 267–278.
30. S. Fox and D. Leake, "Modeling case-based planning for repairing reasoning failures," in *Proceedings of the 1995 aaai Spring Symposium on Representing Mental States and Mechanisms*, AAAI Press: Menlo Park, CA, 1995, pp. 31–38.
31. Z. Zhang and Q. Yang, "Towards life-time maintenance of case base indexes for continual case based reasoning," in *Proceedings of the 1998 International Conference on ai Methodologies, Systems and Applications (AIMSA-98)*, Springer Verlag: Berlin, 1998, pp. 489–500.
32. D. Leake, L. Birnbaum, K. Hammond, C. Marlow, and H. Yang, "Integrating information resources: A case study of engineering design support," in *Proceedings of the Third International Conference on Case-Based Reasoning*, Springer Verlag: Berlin, 1999, pp. 482–496.
33. W. Mark, E. Simoudis, and D. Hinkle, "Case-based reasoning: Expectations and results," in *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, edited by D. Leake, AAAI Press: Menlo Park, CA, 1996.
34. E. Simoudis, K. Ford, and A. Cañas, "Knowledge acquisition in case-based reasoning: "...and then a miracle happens," in *Proceedings of the 1992 Florida AI Research Symposium*, edited by D. Dankel, Florida Artificial Intelligence Research Society, 1992.
35. A. Voß, "The need for knowledge aquisition in case-based reasoning—some experiences from an architectural domain," in *Proceedings of the Eleventh European Conference on Artificial Intelligence*, John Wiley, 1994, pp. 463–467.
36. M. Lenz and K. Ashley (Eds.), in *Proceedings of the AAAI-98 Workshop on Textual Case-Based Reasoning*, AAAI Press: Menlo Park, CA, 1998.
37. D. Leake and D. Wilson, "Case-base maintenance: Dimensions and directions," in *Proceedings of the Fourth European Workshop on Case-Based Reasoning*, edited by P. Cunningham, B. Smyth, and M. Keane, Springer Verlag: Berlin, 1998, pp. 196–207.
38. T. Gruber and D. Russell, *Generative Design Rationale: Beyond the Record and Replay Paradigm* (Knowledge Systems Laboratory KSL 92-59). Computer Science Department, Stanford University, 1992.
39. R. Bergmann, H. Muñoz-Avila, M. Veloso, and E. Melis, "Case-based reasoning applied to planning tasks," in *CBR Technology: From Foundations to Applications*, edited by M. Lenz, B. Bartsch-Spörl, H.-D. Burkhard, and S. Wess, Springer: Berlin, pp. 169–199, 1998.
40. J. Carbonell, "Derivational analogy: A theory of reconstructive problem solving and expertise acquisition," in *Machine Learning: An Artificial Intelligence Approach*, edited by R. Michalski, J. Carbonell, and T. Mitchell, Morgan Kaufmann: Los Altos, CA, vol. 2, pp. 371–392, 1986.
41. M. Veloso, *Planning and Learning by Analogical Reasoning*, Springer Verlag: Berlin, 1994.
42. M. Veloso, A. Mulvehill, and M. Cox, "Rationale-supported mixed-initiative case-based planning," in *Proceedings of the Ninth Conference on Innovative Applications of Artificial Intelligence*, AAAI Press: Menlo Park, CA, 1997, pp. 1072–1077.
43. D. Leake, A. Kinley, and D. Wilson, "Case-based CBR: Capturing and reusing reasoning about case adaptation," *International Journal of Expert Systems*, vol. 10, no. 2, pp. 197–213, 1997b.
44. P. Clark, "A model of argumentation and its application in a cooperative expert system," Strathclyde University, Glascow, UK, 1991.
45. K. Hammond, R. Burke, and K. Schmitt, "A case-based approach to knowledge navigation," in *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, edited by D. Leake, AAAI Press: Menlo Park, CA.
46. M. Jaczynski and B. Trousse, "WWW assisted browsing by reusing past navigations of a group of users," in *Proceedings of the Fourth European Workshop on Case-Based Reasoning*, edited by P. Cunningham, B. Smyth, and M. Keane, Springer Verlag: Berlin, 1998, pp. 160–171.
47. M. Fagan and S. Corley, "CBR for the reuse of corporate SQL knowledge," in *Proceedings of the Fourth European Workshop on Case-Based Reasoning*, edited by P. Cunningham, B. Smyth, and M. Keane, Springer Verlag: Berlin, 1998, pp. 382–391.
48. D. Leake, "Towards a computer model of memory search strategy learning," in *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum: Hillsdale, NJ, 1994, pp. 549–554.
49. D. Leake, A. Kinley, and D. Wilson, "Learning to improve case adaptation by introspective reasoning and CBR," in *Proceedings of the First International Conference on Case-Based Reasoning*,

Springer Verlag: Berlin, 1995, pp. 229–240.
50. M. Göker, T. Roth-Berghofer, R. Bergman, T. Pantleon, R. Traphöner, S. Wess, and W. Wilke, "The development of HOMER: A case-based CAD/CAM help-desk support tool," in *Proceedings of the Fourth European Workshop on Case-Based Reasoning*, edited by P. Cunningham, B. Smyth, and M. Keane, Springer Verlag: Berlin, 1998, pp. 346–357.
51. W. Ferguson, R. Bareiss, R. Osgood, and L. Birnbaum, "ASK systems: An approach to the realization of story-based teachers," *The Journal of the Learning Sciences*, vol. 1, pp. 95–134, 1992.
52. D. Leake and D. Wilson, "Combining CBR with interactive knowledge acquisition, manipulation and reuse," in *Proceedings of the Third International Conference on Case-Based Reasoning*, Springer Verlag: Berlin, 1999, pp. 203–217.

David Leake is an Associate Professor of Computer Science and member of the Cognitive Science faculty and School of Informatics at Indiana University. He received his Ph.D. in Computer Science from Yale University in 1990. His primary research interests are artificial intelligence and cognitive science, including case-based reasoning, intelligent information search, intelligent problem-solving environments, and knowledge management. He has over 70 publications in these areas. He is the author of *Evaluating Explanations: A Content Theory* (Erlbaum, 1992), co-editor of *Goal-Driven Learning* (MIT, 1995), and editor of *Case-Based Reasoning: Experiences, Lessons, and Future Directions* (AAAI Press, 1996). He co-chaired the Second International Conference on Case-Based Reasoning (ICCBR-97). He is the editor of *AI Magazine*.

David Wilson received his B.A. (1993) with high honors in Computer Science and Mathematics from Rockford College, and his M.S. (1995) and Ph.D. (2001) in Computer Science from Indiana University. His research interests include artificial intelligence and cognitive science, case-based reasoning, maintaining knowledge-based systems, introspective reasoning, intelligent problem-solving environments, memory, and knowledge representation. He is currently a College Lecturer in the Department of Computer Science at University College Dublin. His address is: Department of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland (david.wilson@ucd.ie).