Improved Performance Support through an Integrated Task-Based Video Case Library

Christopher L. Johnson[1*], Larry Birnbaum[1], Ray Bareiss[2], and Tom Hinrichs[3]

[1]Northwestern University, Department of Computer Science
1890 Maple Avenue, Evanston, IL 60201
cj@mitre.org, birnbaum@ils.nwu.edu

[2]Cognitive Arts Corporation
115 E. 57th Street, 10th Floor, New York, NY 10022
bareiss@cognitivearts.com

[3]Cognitive Arts Corporation
1840 Oak Avenue, Evanston, IL 60201
thinrichs@cognitivearts.com

Abstract. Case-based retrieval and other decision support systems typically exist separately from the tools and tasks they support. Users are required to initiate searches and identify target case features manually, and as a result the systems are not used to their full extent. We describe an approach to integrating an ASK system—a type of video case library—with a performance support tool. This approach uses model-based task tracking to retrieve cases relevant to how a user is performing a task, not just to the artifacts that are created during the process.

1 Introduction

A common scenario for using case-based help retrieval begins with a person describing the particulars of a problem to a retrieval system, often by filling out a form or answering a series of questions. However, there are circumstances where this scenario is less than ideal. Requiring users to build a target case description in order to access help requires added effort which they may not be willing or able to give.

For instance, the hardest problem for large organizations implementing knowledge management initiatives is frequently cultural, not technical (e.g., [1]). Convincing members to add "extra" tools and processes to their already busy work schedule is one of the difficulties. As another example, users may be performing a time-critical task and unable to devote extra time to searching for help. Or the complexity of a task may require a relatively unreasonable amount of time to identify all relevant case features. At other times users don't even realize that they need help, much less that cases are available which apply to their current situation.

---

* Christopher Johnson currently works for The MITRE Corporation, Cognitive Science & Artificial Intelligence, 1820 Dolly Madison Blvd., McLean, VA, 22102.

D.W. Aha and I. Watson (Eds.): ICCBR 2001, LNAI 2080, pp. 675-689, 2001.

This paper describes a system we built to address this problem in the use of an ASK system, a type of video-based organizational memory. The Air Campaign Planning Advisor (ACPA) uses model-based task tracking to integrate an ASK system with a plan construction tool used by air campaign planners.

2 Issues with Non-integrated ASK Systems

Over the past several years, the Institute for the Learning Sciences at Northwestern University and the Cognitive Arts Corporation have developed an effective approach to creating organizational memories called ASK Systems [2]. This family of hypermedia systems aims at providing conversational access to large case libraries for training and performance support. Each ASK system contains a large number of 30-second to 2-minute video clips in which experts recount "war stories" of their actual experiences in these domains. These systems represent our attempts to develop organizational memories that capture the less tangible elements of human expertise.

One of the largest ASK systems we have built, Trans-ASK, contains 21 hours of video detailing the experiences of United States Transportation Command (USTRANSCOM) personnel in planning for Operations Desert Shield and Desert Storm, as well as the Somalia and Hurricane Andrew relief operations [3]. Shortly after Trans-ASK was completed, we tested it with novice transportation planners from USTRANSCOM for utility and usability. The test consisted of several exercises. Novices were given the same kinds of scenarios, constraints, information, and instructions that they would receive in real operations and were then asked to develop an acceptable transportation plan.

The test was a success: Trans-ASK did indeed provide natural and easy access to potentially useful information. Novices found it quite easy to navigate through the system to retrieve relevant answers to their questions. (Expert planners observing the test assessed the relevance of the information they retrieved.) However, during the assessment a few problems emerged which seemed to limit the potential utility of Trans-ASK. First, even though novice planners were able to find good answers to their questions, they weren't necessarily able to apply those answers to produce better transportation plans. Second, during focus group discussions following the exercises, participants agreed nearly unanimously that they would probably *not* use such a system in day-to-day operations—not because it wasn't valuable, but because it wasn't integrated with their normal work environment. To use Trans-ASK, planners would need to stop whatever they were doing, open a new application, and traverse a hierarchical menu system, specifying their current role, task, and problem to find a starting point in the case base—a time-consuming and distracting chore for someone in the midst of resolving a problem.

Finally, Trans-ASK presented one additional problem. Many of the experiences related in ASK systems deal with problems involved in performing a task. The best time for users to hear these experiences is when they are experiencing the same type of problem. But with a stand-alone ASK system, users first have to realize that they are having a problem before they can get help for that problem. Unfortunately, the nature of problems is that they are often not noticed until after the fact.

After observing these problems with Trans-ASK, we realized that, ultimately, organizational memory systems will be truly useful only if they are integrated with

the tools that directly support users' tasks, i.e., performance support systems. Additional studies support this need. For instance, an extensive survey of *lessons learned systems* found that most systems were implemented with stand-alone retrieval tools, no deployed systems used active or proactive lesson dissemination, and systems generally failed to promote knowledge reuse and sharing [4].

3 Air Campaign Planning Advisor

ACPA is composed of a Web-based ASK system linked to a performance support tool through a model-based task tracking system. The ASK system itself contains approximately 1,000 video clips—about 20 hours—in which 12 expert air campaign planners relate their experiences in the Gulf War, Bosnia, Somalia, and Haiti, as well as several training exercises. The JFACC Planning Tool (JPT), to which the ASK system is linked, supports the authoring of hierarchical air campaign plans. (JPT, which was built by ISX Corporation, has been adopted by the US Air Force.) Finally, the task tracker that links them, which we call Argus, enables the entire system to provide contextual help and advice relevant to the current state of planners' planning process as they use JPT. The system is also capable of providing proactive critiques of the planning process as well as of the current plan itself. Figure 1 illustrates the basic architecture of ACPA.

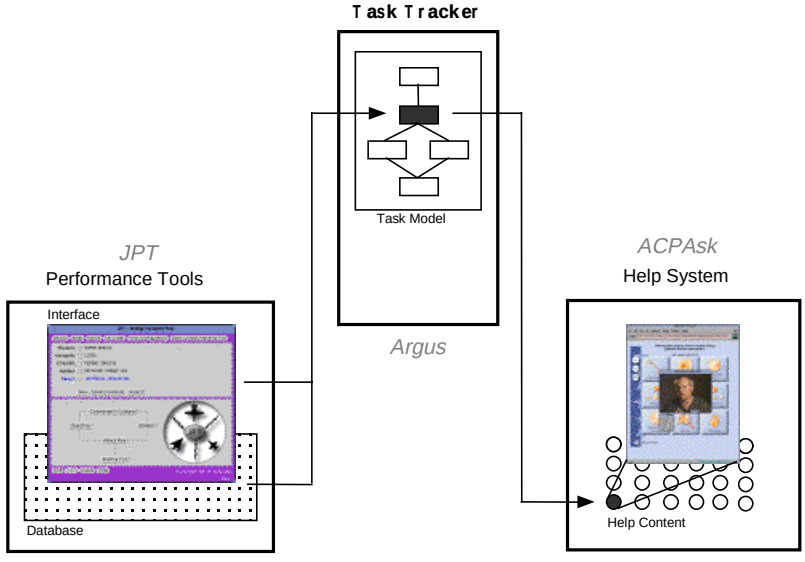

Fig. 1. An architecture for integrated performance support

The ACPA system embodies our vision of an intrinsic and integrated organizational memory. Our vision entails using performance support tools as a framework that helps ground the advice contained in the organizational memory. At

the same time, by tracking users' progress through the task, such an integrated system could provide instantaneous access to relevant cases, thus relieving users of the burden of leaving the current task context to search for help. Again, our goal is to minimize the user's effort in obtaining guidance: any time spent away from the task in navigating, searching, or answering questions to identify appropriate case features for retrieval is too disruptive.

4 ACPA in Detail

The air campaign planning (ACP) task, greatly simplified, is the process of finding answers to three questions: What enemy targets need to be addressed? When do these targets need to be addressed? What resources should be used to address these targets? While answering these questions, planners should be guided by the dominant goals of the campaign. In other words, what would be the result of a successful military operation against an opponent? Each target should then directly support these goals. This is the task that ACPA was built to support.

4.1 Components of ACPA

The performance support tool in ACPA is the *JFACC Planning Tool* (JPT). JPT is an authoring tool for air campaign plans that also provides easy access to information and other tools required during the authoring process. The tool allows planners to create better air campaign plans more quickly and fosters greater collaboration and plan reuse. JPT is one of the actual tools that planners use to create air campaign plans today; it was developed to address planning problems encountered in the Gulf War and has been in active use since the first Bosnia operations.

The central activity of a planner using JPT is creating a six-level hierarchy of objectives and targets, ranging from national security objectives down to individual targets. This hierarchy of objectives is intended to both structure and archive the strategy behind a given list of targets, which allows others to better understand, critique, and modify the plan. As planners create objectives and targets, they often need to perform research or use additional software tools. To accommodate these needs, JPT also provides aids such as online access to commander's guidance and intelligence documents.

In ACPA, the Air Campaign Planning ASK system (for clarity, ACPAsk) plays the part of the organizational memory. We defer to previous papers [3, 2] for detailed discussions of the ASK concept. Specifically, ACPAsk is a Web-based ASK system that currently contains approximately 1,000 video clips—13.5 gigabytes worth—about air campaign planning, related by 12 experts. The clips are richly interconnected by approximately 28,500 links.

The two main activities involved in using an ASK system are *zooming* and *browsing*. When zooming, ACPAsk allows planners to navigate down through a graphical representation of our air campaign planning task model (Figure 2). When planners select a task in the model, ACPAsk either displays a more detailed model of the subtasks that form that task or, if the task has no subtasks, displays a set of questions related to the task. These questions are divided into three categories:

questions related to the *process* of performing the task, questions related to *concepts* associated with the task, and questions related to *problems* involved in performing the task.

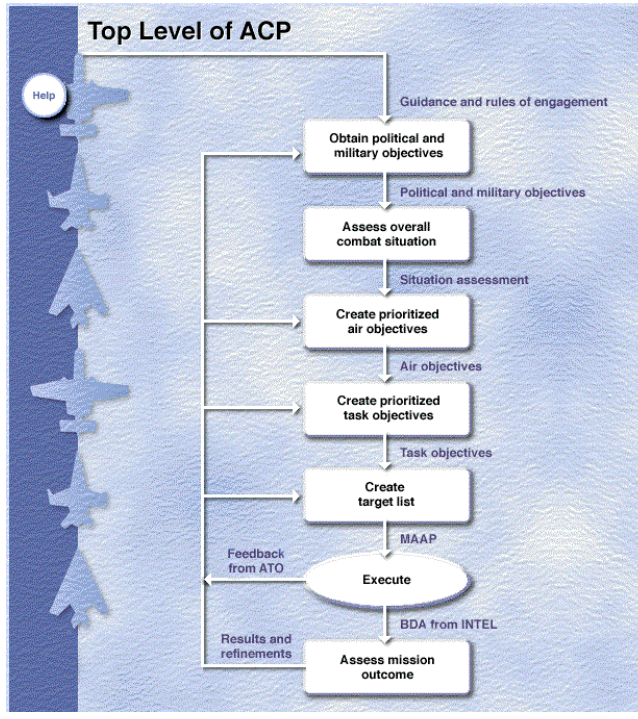

Fig. 2. Top level of the air campaign planning task model

Each question leads to a story page that allows planners to watch a video of an expert air campaign planner telling a story that addresses their question.

After listening to the story, planners can browse through related stories in ACPAsk by selecting one of eight relational links surrounding the story. In this manner, planners can choose to view other stories addressing follow-up questions they may have about *examples*, *alternatives*, *opportunities*, *warnings*, *causes*, *results*, *background information*, or *details* relative to the issues discussed in the story.

Each of these relational links leads to another set of questions, each question leads to another story, and each story has the same relational links that can lead to eight more sets of questions.

The final component of ACPA—the adhesive that binds planners' performance in JPT and the expertise in ACPAsk—is *Argus*. The purpose of Argus is to monitor planners as they create an air campaign plan using JPT and then to provide the most appropriate ACPAsk stories to them at the most appropriate times. To achieve this, Argus tracks planners' progress through the explicit model of air campaign planning as they work with JPT. By monitoring the actions that planners take, such as opening editors, selecting options from menus, or adding air objectives to their plan, Argus

works to infer planners' current goals in the task model. Because stories in ACPAsk are also indexed by the task model, when planners ask for help, Argus can simply look at the task the planners are currently performing and give them the stories stored under that task in ACPAsk.

Argus can also infer specific problems planners might encounter as they build an air campaign plan, using problem identification heuristics associated with each subtask in the model. In this case, Argus proactively alerts planners to the problem by placing a small alert window on the top right side of the screen. Planners can then choose to view stories relevant to the problem or continue planning without interruption.

4.2 Relation of ACPA Components to Case-Based Reasoning

The focus of this research is on Argus, the task tracking component that links JPT to the ASK system. While not completely automating the retrieval phase of a case-based reasoning (CBR) process, we intend Argus to relieve much of the burden of initiating retrieval of cases by the human.

Other CBR issues—like indexing, case acquisition, and adaptation—tend to fall in the scope of developing ACPAsk. As mentioned earlier, ASK systems have been well established at Northwestern for awhile, so ACPAsk development played more of a supporting role for Argus. Again, we refer to previous papers [2, 3] for more detailed discussions of ASK systems.

But briefly, considered as a CBR system, ASK systems support a mixed-initiative CBR process where the human actually performs most of the work. The ASK system provides a browsable library of video cases to aid retrieval; the case index it provides is modeled around a conversation between a novice and an expert in order to make browsing as natural as possible. There is no detailed representation of an individual case, other than the way in which cases are linked to a task model and to each other (e.g., this case is related to task *x* and is an alternative to case *y*). Standard practice at Northwestern has been for groups of content specialists to index the ASK videos in this manner; there are tools that have been developed to assist them.

4.3 Sample Interaction with ACPA

Consider a novice air campaign planner using JPT to construct a plan. JPT itself looks and behaves no differently than it did without ACPA added. As mentioned earlier, the central activity supported by JPT is the creation of a six-level hierarchy of objectives and targets. Imagine that the planner has reached this point, and is in the middle of editing an air objective (we'll pretend that Major Jill Planner plays the part of our novice planner). Specifically, she has opened the Objectives Viewer by clicking on the *Objectives* button in the main control panel and chosen to create an air objective named *Destroy Libyan Computerized Networks for Information Transmittal.*

However, in the midst of creating this objective, the planner experiences a twinge of irritation. Why must she go through the trouble of filling out this complex hierarchy? She already has an idea of what the targets should be. What's the point?

She clicks the *Help* button. Because the objectives editor is open and is showing an air objective (one of five types of objectives in the hierarchy), ACPA has already

inferred that the planner is currently involved in the task of *Creating Air Objectives*. ACPAsk opens and automatically loads a set of questions about creating air objectives. ACPAsk initially displays a set of questions on the process of creating air objectives (Figure 3), and Jill sees the question *Why must tasks be derived from an objective hierarchy?* This seems relevant, so she selects it.

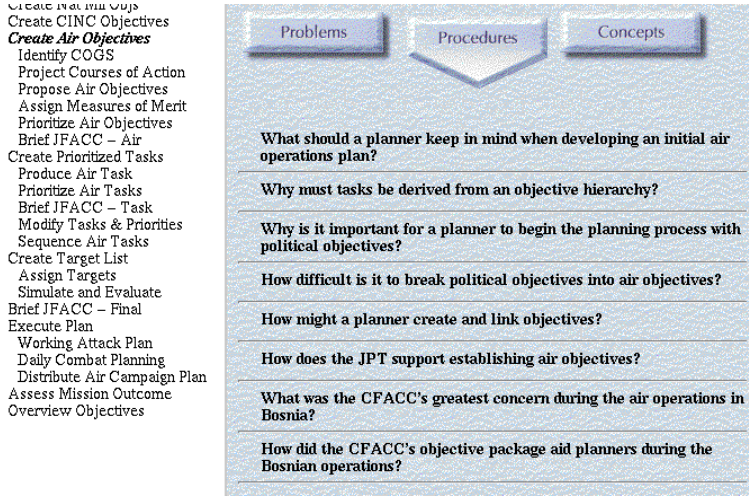

Fig. 3. A set of questions about procedures performing the task of creating air objectives

ACPAsk then opens the appropriate story page (Figure 4), and Jill can view an expert video, in which Lieutenant Colonel Joe Roberts talks about the importance of an objective hierarchy as a supporting rationale for the plan, thus avoiding wasted resources and possible failure of the entire campaign.

Jill still isn't satisfied. She would like to obtain more background on the nature of this "decomposition" process. She clicks the Background button, and ACPAsk displays a set of background questions. The question *How does a strategy to task review help a planner to identify targets?* looks interesting, so she selects it[1]. The appropriate story page opens and the planner watches a video of Col. Robert Plebanek discussing how this method allows planners to more quickly create plans with better emphasis on the important high level effects of an air campaign (e.g., leadership, communication), and because the supporting rationale is included, the plan can also be altered more quickly.

At this point, Jill decides that the time has come to return to JPT, so she minimizes ACPAsk and she's back at the objective editor. In fact, she also closes the objective editor in preparation for creating another objective. This creates a problem with her campaign plan. There are at least four components that every objective in an air campaign plan is supposed to have: a title, a center of gravity, a measure of merit, and

[1] The *strategy to task* approach advocates that each target should explicitly service at least one high-level goal [5]. JPT supports this approach by requiring targets to be directly linked to the highest level national security objective via a hierarchy of intermediate objectives.

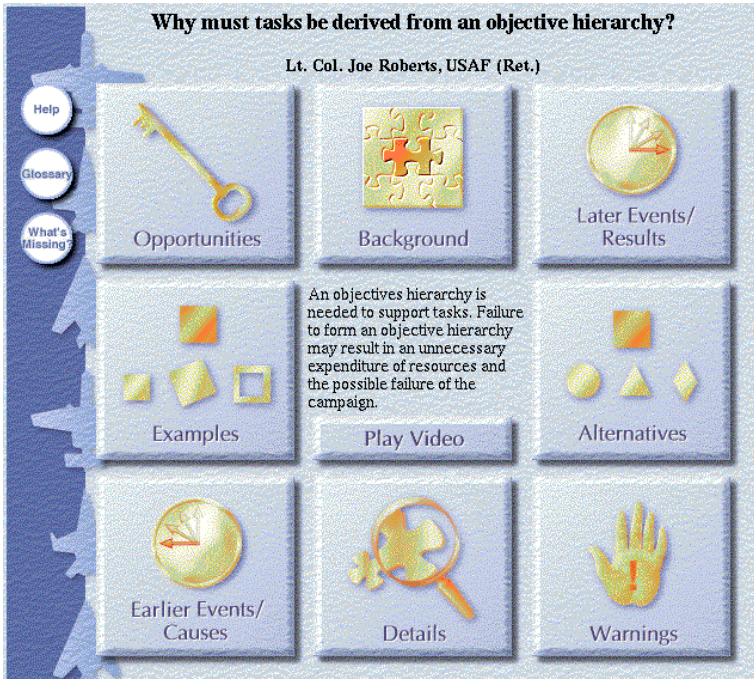

Fig. 4. A story page in ACPA. The user can either watch a video of the expert telling the story or click on one of the eight buttons surrounding the summary to see related questions and stories

links to parent objectives. When the planner closed the objective editor, she had yet to identify a center of gravity [6] for this objective.

In our scenario so far, we have seen one mode of support: the *reactive* help that ACPA provides when the planner explicitly requests it. However, ACPA provides another mode of help support as well: *proactive* help that ACPA presents automatically when it notices a problem with the planner's task. Here is just such a case.

When the planner closed the objective editor, ACPA assumed that she had finished with the Create Air Objectives task, but recognized that the objective she had been working on was missing a component. ACPA considers this to be a problem, and thus alerts the planner by displaying a small alert window. If the planner wishes not to be disturbed, she may ignore the alert window and it will quietly disappear in a minute. However, Jill is curious about what advice ACPA has to offer and clicks on the *Help* button located in the alert window.

ACPA appears and displays both the task problem that it thinks the planner is having and a set of questions related to that problem. In this case, the problem is *No Center of Gravity Identification* and only one question inhabits the set: *Should you ever skip analysis and COG identification because of time constraints?* Jill selects the question, which leads to a video in which Lt. Col. Roberts suggests that taking the

time to perform situation analysis and center of gravity identification actually saves time in the long run.

This scenario, although clearly compressed and simplified, illustrates the type of interaction that linking an ASK system to a performance support tool like JPT, via a task-tracking mechanism, can provide.

5 Benefits of Task Tracking

In many instances, case features can be retrieved directly from the product—a design, a plan, etc.—that a user naturally generates as a result of performing a task. For example, textual case-based reasoning techniques attempt to derive features from the documents people produce [7]. INSPECT [8], another system that supports the task of air campaign planning, retrieves advice based on an analysis of the air campaign plan itself. These methods certainly represent good opportunities for integrating case libraries and other kinds of aids into performance support systems.

However, in other instances crucial case features reside in the user's task process. The stories told by the expert planners in ACPA are as much concerned with how planners go about generating a plan as they are concerned with the final air campaign plan itself. What are users doing right now? What have they done previously? What methods are being used? Are planners developing targets without first developing a strategy? Have they fully assessed the current situation? Did they really think about that information they just entered, or did they simply toss in a value?

Of course, an analysis of the task product can provide clues to the state of the task process; ACPA makes use of these clues to the extent that the air campaign plan generated by JPT contains explicit correlations to the planner's tasks. However, ACPA can also examine interface activity and the task model itself when these correlations don't exist or ambiguity does. We discuss some examples of task and problem tracking techniques in the last section.

Essentially, task tracking offers an opportunity for more precise, just-in-time advice, resulting in tighter integration with performance support tools. Instead of receiving a broad collection of advice about an entire design or plan, users can get help on precisely the tasks and problems they're currently thinking about.

6 General Architecture for Integrated Performance Support

Our approach to providing integrated performance support like that of ACPA is based on the use of an explicit task model to organize and guide task tracking, to represent the current state of the task, and to drive the initial access into the ASK system. Designing systems around task models is a central technique in cognitive engineering approaches (e.g., [9]). However, with our approach we want to further the notion by incorporating the task model as an actual, computer-manipulable part of the final system. Essentially, we are putting the main index of the ACPAsk case library to work as an interface between all components of the system. Rather than embedding and scattering task knowledge throughout the system, in a wizard here and an agent there, we wanted the task model to be the central component of the entire system.

Figure 1 illustrates not just the architecture of ACPA, but a general architecture for similar integrated performance support systems.

The first step in designing such a performance support system, then, is to define a task model that identifies steps in the process and associated information flows. Given this task model, the following items should then be defined for each subtask:

- A software "tool" that supports the information processing involved in performing the subtask. (*performance tools*).
- Recognition conditions that indicate when a user is taking part in the subtask. (*task tracker*).
- Recognition conditions that indicate when a user has made an error or encountered a problem while performing a subtask. These conditions could include not completing the subtask, attempting to perform the subtask prematurely, or performing the subtask inadequately (*task tracker*).
- Help and advice relevant to the subtask. This could include a broad range of aid such as background knowledge, step-by-step instructions, extended case studies, sample work products, common problems and solutions, and alternatives. The system could provide this information in various ways, such as entering an ASK system or invoking an intelligent information agent (*information sources*).

We believe that the use of explicit task models and task tracking to link organizational memory and performance support shows promise as a general performance support architecture. Potential advantages include the following.

- *Improved Task Monitoring.* Because the task model is explicit and centralized, task tracking agents can access knowledge that allows them to make better inferences about task state and provide more specific help to users.
- *Improved Organizational Memory Access.* The relatively natural access to information that ASK systems already provide can be improved further by using task models to track tasks and link ASK systems to a performance environment, which removes the burden of searching from the users.
- *Improved Human-Computer Communication.* By using explicit task models to create a shared context, a performance support system can support better communication and negotiation between computers and users. Because an explicit task model can be "understood" by both the systems and its users, the users can more easily see what the system is doing, and why, when it suggests advice or warns about a problem. Likewise, when the system makes a mistaken inference, users can more easily tell the system what they think the correct inference should have been.
- *Improved System Integration.* Explicit task models can act as an interface between components of a performance support system, resulting in better system integration. Instead of hard-coding particular stories, methods, or data objects to particular windows, interface actions, or tools, tasks can coordinate the interaction between these items. In turn, the performance support system becomes more flexible and adaptable.

7 Task Model Representation

Figure 2 illustrates the top level of our current air campaign planning task model. Many of the subtasks it contains are themselves complex tasks and are in turn broken down and modeled at a comparable level of detail. Task representations and their use have been extensively studied within the human factors and systems analysis communities, among others. Why did we choose to represent this particular content in this particular fashion? The simple answer is that we went with what worked for us—we based our model on empirical utility. Because our approach to organizational memory and performance support is grounded in case-based reasoning theory, with ASK systems playing a crucial role, our decisions were driven by how well the task model served as an indexing framework for the stories, information, advice, and tracking rules we had gathered and developed.

The result of this style of task modeling is a task model that seems relatively simple and intuitive. Several years ago, our research group pursued a more elaborate approach to the explicit modeling of planning tasks that resulted in an aggregation of complex predicate calculus clauses [10]. For this project, though, such complex and highly formal task representations appeared unnecessary for several reasons. First, such a model seemed likely to prove cumbersome when used to drive an application as dynamic as task tracking in a task as complex as air campaign planning.

Second, and more important, complex formal representations are difficult for typical users and designers to understand. And our project was not just about getting a computer to understand what users are doing, but about being able to communicate this understanding to users as well. In order to successfully collaborate with users, a performance support system must be *comprehensible* and *advisable*. Studies into interfaces and decision support for operators of complex, real-time, high-risk systems such as nuclear power plants have found similar needs [11]. Users must be able to trust and understand what the system is doing and why, and the system must be able to accept advice about what it did in a form that is natural to users. In other words, a user and a computer need to collaborate within a shared context. However, complex predicate calculus, or in fact any overly complex form of model, is not a natural basis for this shared context; rather than trust and understanding, it's more likely to foster confusion and frustration in the typical user.

The upshot is that our model of the air campaign planning task is represented in the form of a finite-state machine. There also are several levels representing the top level task and associated sub-tasks. Each level is composed of objects representing individual tasks (we used a simple frame system for our implementation) linked by sequencing information. Each task object is linked relevant cases and problem objects, which are also linked to relevant cases. Although we fully expect that it will ultimately be necessary to extend this approach, we believe that it provides an extremely useful start toward addressing the problems mentioned earlier. First, the simplicity of our approach translates well into simple graphical representations of tasks, which are easily understood by users and designers. Second, it encourages a comparably simple approach to task tracking, where tasks, problems, and transitions within the model are signaled by specific interface and database events. Third, it simplifies modification and extensions to the task model. Finally, the simplicity of the task model is naturally compatible with the mechanisms underlying many user interface management systems.

The most critical point, though, is this: it is the *content* of the task model, not its form, that drives the application. The model's power comes from the kinds of task and problem objects included, and how well they organize cases around a person's current context and thinking.

8 Techniques for Task Tracking

In ACPA, we were specifically interested in tracking two aspects of a task: the progress of users through the task, that is, what subtask a user is currently performing, and the problems encountered by users as they perform the task. These two categories also correspond to the two help modes of Argus: the reactive mode, in which a user explicitly asks for help and Argus responds, and the proactive mode, in which Argus notices a potential problem and notifies the user automatically.

8.1 Tracking Task Flow

While tracking task flow, we can analyze four kinds of information. First, we can look at *windows* (or agents, since windows are often simply manifestations of a specific tool), considering which are open, which are closed, which have "focus," which are visible, even which portions of a specific window are visible. Second, we can analyze *interface activity*—which buttons users click, which fields they type in, which text boxes they scroll, etc. Third, we can analyze *data activity* generated in the performance support tool's database by the users as they work. Fourth, we can analyze the *task history*, recording in the task model what users have achieved so far, and then using the task model itself as a source of information to guide inferences about their current location in the task.

These are the basics. There are also several other interesting task flow issues dealing with user perspective, task context, false starts, and so on, but in the remainder of this paper, we'd like to concentrate on tracking task problems.

8.2 Tracking Task Problems

Most of the especially interesting stories in ACPA—or any ASK system actually—concern problems. Failures are inherently interesting, and according to case-based reasoning theory, a crucial component of learning [12]. People are often most open to new knowledge and forming new cases when their expectations about the world around them fail. Consequently, explicit task problem representation and tracking compose a major part of our integrated performance support methodology.

When we began to investigate what problem recognition techniques might be, we knew that some problems would be easy to recognize, because the *syntax* inherent in the process of using JPT, or in the plan generated by JPT, directly reflected the *semantics* of the air campaign planning task. For instance, tracking whether or not a planner has skipped the *identify measure of merit task* could be cued by the absence of a value in the measure of merit space of the generated plan.

However, we also knew that there would also be a group of problems that didn't fit this pattern, and as a result, would be extremely difficult to track. For example, Table 1 lists the set of these problems types we found in ACPAsk related to developing strategy.

Table 1. A summary of air campaign planning problems that are difficult to track

Developing Strategy	
Not thinking strategically	Not addressing appropriate top priority
Objectives driven by measurables	Developing weak measurables
Objective too broad	Immediate creation of targets
Not thinking creatively	No viable solution
Failing to maintain perspective	Conflicting plan choices
Creating a serial rather than parallel plan	Failing to allow for uncertainty
Misinterpretation of standard domain concepts	

For us, then, the most interesting question became, Can we go beyond tracking the easy problems and help the planners with the difficult ones? In other words, how can we get these interesting, useful experiences stored in the organizational memory to planners when they really need them?

The answer we came up with is an idea we call *discount diagnosis*. Accepting the fact that we may not be able to identify a given problem with certainty, could we still look for simple clues that would indicate the possible presence of the problem, at least in some cases?

Not Thinking Strategically. For instance, consider the first problem listed in Table 1. One story in the air campaign planning ASK system talks about how, at a certain training exercise, a group of planners failed to develop a strategy—they "just started doing stuff." How can we detect this problem? The planners are using the planning tools, but how can we tell whether they are thinking strategically as they do so?

Several trackable features may indicate that this problem exists. For instance, we can look at the planner's task process, or how the planner performs the task. Perhaps a planner jumps directly from consulting the commander's guidance to creating targets. If the planner skips intermediate strategy-building tools, he may not be thinking strategically. Skipping tasks that support the development of strategy but are not used for directly recording strategy, such as situation assessment, may also indicate non-strategic thinking.

We can also look at the task product, or the items that are generated by the task process. For instance, in JPT, the air campaign plan's strategy is represented by a six-level hierarchy of objectives that connect top-level national security objectives and individual targets. If one of these objectives in the hierarchy is connected to too many parents or too many children, that objective may be "too broad," which in turn may mean that the planner is not thinking the situation through. Target ratios may also be an indication that the planner is not thinking strategically. For example, if a plan contains a large number of "low-impact" targets (e.g., anti-aircraft, ground forces) that have little impact on the plan's primary objectives, compared with "high-impact" targets (e.g., communications facilities, power plants), then the plan may not possess much of a strategy.

Patterns of Evidence. We have found the patterns of evidence listed in Table 2 useful for tracking these types of problems.

Table 2. Patterns of evidence for tracking task problems

Task order: Are there abnormal patterns that appear in the order of the subtasks? For instance, a user who jumps around in the task may be confused; a strategic planner who jumps immediately to creating target lists may not be thinking strategically.	*Quantities*: Is the user creating too much or too little of a particular type of component? Is the user performing too much or too little of a particular type of action?
Intervals: Has the user failed to do something after a certain interval of time?	*Key events*: Has the user done something or failed to do something after a certain event?
Ratios: Is there an imbalance in the quantity of one type of component in the work product compared to another type of component? Is there an imbalance in the number of times a user performs one type of action compared to another type of action?	*Modifications*: Has the user changed or failed to change something after a certain interval of time? Has the user selected a component from an archive of previous components and changed or failed to change it?
Searching: Does the user appear to be searching for long periods of time?	*Phrasing*: Is the user invoking words or phrases that indicate problematic or vague concepts?
Thrashing: Does the user appear to be moving quickly back and forth between a certain group of tools?	

9 Conclusions and Tools for the Future

Limited space prevents us from presenting a detailed comparison between ACPA and related work. However, ACPA is particularly distinguished by the fact that it retrieves cases based on features of a person's entire task process, not just features of a task product (e.g., a design, a plan, or a text). Furthermore, ACPA does this through integration with a performance tool and the use of task tracking to infer task features, without requiring the person to manually specify features to a stand-alone tool. Also, task tracking and case retrieval are supported by an explicit task model that is an internalized component of the final system.

As for future work, we believe that there exists a set of abstract, standardized, reusable task models, general enough to cover a wide variety of domains, and yet containing enough specific content to direct the construction of useful interfaces, performance support, and so forth. For example, the task of air campaign planning can be seen as a specific case of the more abstract task of strategic planning, which itself is the root of similar tasks in other domains. These tasks, such as strategic business planning, share common processes, common tools, and common problems.

We eventually hope to create a library of such task models, as well as development tools that are driven by them. We believe the result will be easy-to-use, easy-to-build, easy-to-modify, highly intelligent performance support and training systems.

Acknowledgements. We thank everyone at the United States Air Force, the ISX Corporation, and Northwestern University who helped in the development of ACPA. This work was supported in part by the Defense Advanced Research Projects Agency and Rome Laboratory under contract F30602-95-1-0019. Chris Johnson was supported in part by a graduate fellowship from the National Science Foundation.

References

1. Davenport, T.H., Prusak, L.: Working Knowledge: How Organizations Manage What They Know. Harvard Business School Press, Boston (1998)
2. Ferguson, W., Bareiss, R., Birnbaum, L., Osgood, R.: ASK Systems: An Approach to the Realization of Story-Based Teachers. The Journal of the Learning Sciences 1,2 (1992) 95-134
3. Bareiss, R., Osgood, R.: Applying AI Models to the Design of Exploratory Hypermedia Systems. Hypertext'93 (Seattle 1993). ACM Press 94-105
4. Weber, R., Aha, D.W., Becerra-Fernandez, I.: Intelligent Lessons Learned Systems. Expert Systems with Applications 20 (2001) 17-34
5. Thaler, D.E.: Strategies to Tasks: A Framework for Linking Means to Ends. RAND, Santa Monica, CA (1993)
6. Warden, Colonel J.F. III: 1989. The Air Campaign: Planning for Combat. Pergamon Press, Washington, DC (1989)
7. Lenz, M., Ashley, D.: Textual Case-Based Reasoning: Papers from the 1998 AAAI Workshop. AAAI Press, Menlo Park, CA (1998)
8. Valente, A., Blythe, J., Gil, Y., Swartout, W.: On the Role of Humans in Enterprise Control Systems: the Experience of INSPECT. DARPA-JFACC Symposium on Advances in Enterprise Control (San Diego 1999)
9. Rasmussen, J.R.: Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering. North-Holland, New York (1986)
10. Collins, G., Birnbaum, L., Krulwich, B., Freed, M.: The Role of Self-Models in Learning to Plan. In: Meyrowitz, A.L., Chipman, S. (eds.): Foundations of Knowledge Acquisition: Machine Learning. Kluwer, Boston (1993) 117-143
11. Mitchell, C.M.: Models for the Design of Human Interaction with Complex Dynamic Systems. Cognitive Engineering Systems in Process Control (Kyoto, Japan 1996)
12. Schank, R.: Dynamic Memory: A Theory of Reminding and Learning in Computers and People. Cambridge University Press, New York (1982)