Pedagogical Agents on the Web

Erin Shaw, W. Lewis Johnson, and Rajaram Ganeshan
Center for Advanced Research in Technology for Education
USC / Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292-6695 USA
{shaw, johnson, rajaram}@isi.edu
http://www.isi.edu/isd/carte/

Abstract
Animated pedagogical agents are lifelike animated characters that facilitate the learning process. This paper describes Adele, a pedagogical agent that is designed to work with Web-based educational simulations. The Adele architecture implements key pedagogical functions: presentation, student monitoring and feedback, probing questions, hints, and explanations. These capabilities are coupled with an animated persona that supports continuous multi-modal interaction with a student. The architecture supports client-side execution in a Web browser environment, and is able to inter-operate with simulations created by off-the-shelf authoring tools.

Introduction
Animated pedagogical agents are animated characters that facilitate learning in computer-based learning environments. These agents have animated personas that respond to user actions. In addition, they have enough understanding of the learning context and subject matter that they are able to perform useful roles in learning scenarios.

Although pedagogical agents build upon previous research on intelligent tutoring systems (Wenger 1987), they bring a fresh perspective to the problem of facilitating on-line learning, and address issues that previous intelligent tutoring work has largely ignored. Because pedagogical agents are autonomous agents, they inherit many of the same concerns that autonomous agents in general must address. (Johnson and Hayes-Roth 1998) argue that practical autonomous agents must in general manage complexity. They must exhibit robust behavior in rich, unpredictable environments; they must coordinate their behavior with that of other agents, and must manage their own behavior in a coherent fashion, arbitrating between alternative actions and responding to a multitude of environmental stimuli. In the case of pedagogical agents, their environment includes both the students and the learning context in which the agents are situated. Student behavior is by nature unpredictable; and may exhibit a variety of aptitudes, levels of proficiency, and learning styles.

Our goal is to create agents that have life-like personas, that are able to interact with students on an ongoing basis. This contrasts with other pedagogical agent work (e.g., Ritter 1997) that ignores personas. Animated personas can cause learners to feel that on-line educational material is less difficult (André et al 1998). They can increase student motivation and attention (Lester et al 1997). But most fundamentally, animated pedagogical agents make it possible to more accurately model the kinds of dialogs and interactions that occur during apprenticeship learning and one-on-one tutoring. Factors such as gaze, eye contact, body language, and emotional expression can be modeled and exploited for instructional purposes.
This paper focuses on a particular pedagogical agent developed at USC: Adele (Agent for Distance Education – Light Edition). Adele shares many capabilities with our other pedagogical agent, Steve, who was described at previous Autonomous Agents conferences and elsewhere (Johnson et al 1998, Johnson and Rickel 1998, Rickel and Johnson 1998, Rickel and Johnson 1997). Adele extends the pedagogical capabilities of Steve, and applies them to a wider range of educational problems. But whereas Steve was originally designed to operate in immersive virtual environments, Adele is designed to operate over the Web. Web-based delivery both constrains the available modalities of interaction with the user and imposes strong requirements on the implementation. This paper describes Adele’s capabilities and discusses issues relating to hosting such agents in a Web-based learning environment.

Figure 1. Adele explains the importance of palpating the patient’s abdomen.

**Design objectives**

Adele is designed to support students working through problem-solving exercises that are integrated into instructional materials delivered over the World Wide Web. Adele supports both single-user, single-system tutoring and multi-user, multi-system collaborative exercises.

Figure 1 shows a typical case-based diagnosis exercise in which students are presented with a simulated patient in a clinical setting. In the role of physicians, students are able to perform a variety of actions on the simulated patient; they may ask questions about medical history, perform a physical examination, order diagnostic tests, and make diagnoses. Adele monitors the student’s actions and provides feedback accordingly. Depending upon the instructional goals, Adele may highlight aspects of the case, suggest correct actions, provide hints and rationales for particular actions, reference relevant background material, and provide contextual assessment, to test a student’s understanding.
In the time-critical trauma care exercise, shown in Figure 2, the patient’s state changes over time. Adele monitors the patient’s state as it changes, as well as the student’s actions, and alerts the student if she detects an instability in the patient’s condition to which the student does not promptly react. The trauma care exercise is designed to be collaborative, as is typically the case in an emergency room, and currently supports the roles of physician and nurse.

![Figure 2. Adele advising in a critical care scenario on the World Wide Web.](image)

**Agent-oriented approach**

Adele’s design was based on an autonomous agent paradigm instead of an intelligent tutoring system paradigm. The design was based heavily upon earlier work on Steve. In the case of Steve, the distinction with conventional tutoring systems is fairly clear. Each Steve agent is able to operate in a dynamic environment incorporating multiple students and multiple other Steve agents. Steve can manipulate objects in the virtual environment, in order to demonstrate how to perform tasks, or in order to collaborate with students. Steve can sense where the student is and what he or she is looking at, and can adapt instructionally. He can use gaze, body position, and gesture to engage the student in multi-modal dialog.

Because Adele is confined to a conventional desktop GUI interface, she has fewer options for interacting with students. Nevertheless, the agent-based approach offers advantages, even with the more limited interface. Adele’s use of gaze and gesture, and her ability to react to student actions, makes her appear lifelike and aware of the user, and her use of facial expressions can have a motivating influence on a student. Adele was designed modularly and can be integrated with Web-based exercises and simulations that are authored with off-the-shelf tools that include an accessible programming interface that supports Adele’s interface requirements. Our generalized approach to authoring cases makes it easy to create large numbers of them, making the process cost-effective.
Figure 3. Architectural overview of the single-user system. In the multi-user system, the Reasoning Engine is server-based, as is a new component, the Session Manager.

Architectural overview

Adele’s system, shown in Figure 3, consists of four main components: the pedagogical agent, the simulation, the client-server, and the server store. The pedagogical agent consists further of two sub-components, the animated persona and the reasoning engine. A fifth component, the session manager, is employed when the system is run in multiple-user mode. The central server is used to maintain a database of student progress and when appropriate, to provide synchronization for collaborative exercises being carried out by multiple students on multiple computers.

The reasoning engine performs all monitoring and decision making. Its decisions are based on a student model, a case task plan, and an initial state, which are downloaded from the server when a case is chosen, and on the agent’s current mental state, which is updated as a student works through a case. Upon completion, a record of the student’s actions is saved to the server where it is used to assess the level of the student’s expertise and determine how Adele will interact with the student in future cases.

The animated persona is simply a Java applet that can be used alone with a Web page-based JavaScript interface or incorporated into larger applications, such as the simulation-based exercises we describe here. We chose to create our own animated agent instead of using an off-the-shelf persona like Microsoft Agent, to ensure platform independence and extensibility. The persona applet allows animation frames to be easily added and exchanged to support a choice of personas.

The simulation can be authored using the language or authoring tool of one’s choice. For example, the simulation for the clinical diagnosis application was built in Java while that for a trauma care application was built using Emultek’s RAPID, a rapid-prototyping tool whose simulations can run in a Web browser inside a plug-in. All simulations communicate with the agent via a common application programming interface (API) that supports event (e.g., the student orders a lab) and state change (e.g., the lab value is updated) notifications as defined by the simulation logic.

The integrated system is downloaded to and run on the client’s side for execution efficiency. This is in contrast to the architecture of most other Web-based Intelligent Tutoring Systems where the intelligent tutor sits on the server side, resulting in increased latency in tutor response to student
actions (e.g., Brusilovsky et. al 1997). Reducing latency is especially critical when one considers the overhead of animating an agent’s response to a student’s action, in order to achieve the perception of awareness in a shared workspace.

**Task representation**

The representation scheme used in Adele was designed to be simple and general, yet enable Adele to provide useful feedback to students. Simplicity is essential in order for the agent reasoning engine to run efficiently on the client side. It is also essential in order to support knowledge acquisition and authoring; we want domain experts to be able to specify domain knowledge for Adele, with a minimum of intervention by knowledge engineers. Generality is important so that Adele can be applied to as wide a range of courses as possible. Adele is intended to be used in a range of Internet-based science courses, not just a specific discipline such as clinical medicine.

Adele’s knowledge representation focuses on the steps that the student should take in the task, the dependencies between them, and their rationales. The task steps and their dependencies are represented in a Task Plan. Each task plan is described in a general enough fashion to allow students to perform actions in whatever order they wish, as long as critical ordering constraints are satisfied. The task plan framework is not specific to medicine or dentistry, but could be applied to a range of skills. The rationales typically refer to underlying domain knowledge such as disease properties. In order to ensure simplicity, Adele’s knowledge is specialized for each case. This allows us to avoid formalizing extensive amounts of background knowledge in Adele. Instead this knowledge is provided informally in the rationale texts associated with each task plan step, and in supporting Web-based reference materials.

**Task plan**

Diagnosis and treatment are canonical tasks in the health science settings in which Adele is deployed. Depending on the context, a task can be a simple sequence of steps or it can be a complex and non-linear partial ordering on a set of steps. Adele represents all procedural tasks using a standard hierarchical plan (Russell & Norvig, 1995). A plan hierarchy is comprised of steps, each of which is either a primitive action (e.g. corresponds to a simulation event) or a complex action (e.g. is itself a plan). Adele’s plan representation is based on Steve’s, except that where Steve’s plans are converted into Soar productions and processed by Soar (Laird et. al, 1987), Adele’s plans are read as Java-based objected-oriented data structures and processed by Adele’s Java-based reasoning engine.

```plaintext
(step palpate-axillary
  :effect ((palpate-axillary-done = T)
           (set palpate-axillary-value))
  :phrase "palpate" "palpating" "the axillary nodes")

(step examine-lymphnodes
  :precond (examine-lesion)
  :steps (and palpate-axillary-nodes
           palpate-clavicular-nodes
           palpate-cervical-nodes)
  :hint "Is Lymphadenopathy indicated?"
```


Lymphadenopathy would occur in infectious diseases such as TB, viral, fungal, and some bacterial infections, and in cancer.

The distribution of the nodes involved gives a clue based on their drainage.

**Figure 4.** A primitive step (top) and a complex step (bottom) in a task plan hierarchy.

Figure 4. shows two steps in a task plan hierarchy. The top step, *palpate-axillary-nodes*, is a primitive action and is shown with its effects and dialogue hints. Below it is the complex action, *examine-lymphnodes*, which consists of three required steps, as denoted by the Boolean *and*, as well as a precondition, hint, rationale, verbose rationale, context and role. A step’s end conditions are implicitly defined by its sub-steps and effects, but may be described explicitly as well.

Preconditions and end conditions are represented by a Boolean expressions in conjunctive normal form. They can also be steps as in Figure 4, where the step *examine-lesion* is a precondition of the step *palpate-lymphnodes*. Steps are supported as goals by sub-classing the expression tree to include a new type of Boolean, a *step expression*, which, like Boolean constants, logical expressions, and comparative expressions, evaluates to true or false. Evaluating a step expression is equivalent to evaluating a step’s end conditions, which is done recursively in the case of complex plans. This is an authoring efficiency, and an extension to support the creation of both goal- and task-based plans.

The plan hierarchy is evaluated at each step to account for the dynamic nature of a simulation and the unpredictability of a student’s actions. Actions whose goals become ‘undone’ are automatically re-executed while those whose goals are implicitly satisfied are skipped. In this way, Adele’s plan is dynamically updated.

**Feedback**

The reasoning engine can be run in three modes. In its most restrictive mode, it will simply block actions whose preconditions are unsatisfied. Adele uses this opportunity to provide unsolicited feedback about what *should* be done to satisfy the desired step’s preconditions. The persona displays a *Hint* button so that a student may also ask for hints directly, before guessing or taking an incorrect action. In *practice* mode, the engine does not block – the student can make mistakes – and Adele does not provide unsolicited feedback, but still allows a student to ask for hints. In *exam* mode, Adele is not available. The modes are analogous to those of the SICULE tutor (Alexe&Gecsei, 1996).

Adele uses authored hints when they are available. She proceeds from the top-most relevant point in the plan hierarchy, which is found by working upwards from the desired action to the root of the tree until an unsatisfied node, i.e. sub-plan, is found. If the hint(s) for the plan has already been given then the search continues down to the sub-sub-plans and ultimately to the unsatisfied task itself. As a rule, the higher the sub-plan, the more general its hints will be. For example, for the hierarchical path *diagnosis-root-give-physical-examine-lymphnodes-palpate-clavicular-node*, the hints might be ‘Proceed as would in a clinical setting’, ‘Have you examined the patient?’, ‘Is Lymphadenopathy indicated?’, and ‘Are the clavicular nodes well drained?’, respectively. Adele will follow the path from general to specific when giving feedback.
Authored hints are only given once, to avoid unwarranted repetition, and may not be given at all. For example, asking “Have you examined the patient?” does not make sense if, for example, a student tries to palpate the lymphnodes before examining the lesion (a precondition of palpate-lymphnodes). In this case, Adele skips right to the hint for examining the lesion and invalidate all the general hints above it in the hierarchy. When no authored hints exist, Adele will automatically generate a suggestion using the phrase hints associated with a step.

**Situation-based reasoning**

While the task plan representation described above is adequate for diagnostic tasks, it does not extend well to dynamic settings like trauma or critical care where unforeseen complications may arise. For example, in one exercise, the patient's oxygenation level goes below 30, which is indicative of a breathing problem. If the problem is not corrected immediately the patient's life is at risk. Adele must have the knowledge to guide a student through these complications as they arise.

We have borrowed the notion of a *situation* from Marsella and Schmidt (1990, Marsella & Johnson, 1998) to structure this kind of knowledge in Adele. Marsella and Schmidt introduce the *Situation Space* as a means of structuring the space of states associated with a domain so that it can be used to guide planning activity in dynamic domains. A situation is defined by a name, world state, goal expression, priority, and set of transitions. The world state and goal expressions are partial state descriptions. Priority is used to order situations when more than one is appropriate. Transitions describe possible situations that can result from this situation whenever the associated conditions become true in the world state. Typically, when a situation is entered, a situation-appropriate sub-plan is instantiated to achieve the goal expression. Because the tutoring domain allows us to ‘know’ all possible situations *a priori*, there is no need to generate a plan in real time for each situation. Instead, all situational plans are pre-authored, taking into account potential negative interactions, for example, that of a newly-added step undoing the effects of a previously-executed step, or conflicting with the effects of an existing step that is required to satisfy another goal. Thus, what remains for Adele's reasoning engine is a *situation-monitoring* task.

![Figure 5. Transition from a current situation to a priority situation.](image)

There is always a current situation defined in Adele's reasoning engine, which monitors the world state for changes that may cause a transition to another situation. The current plan that forms the basis for tutoring changes with the situation. In Figure 5, below, *Normal-trauma* is the primary situation, and remains current until the value of oxygen becomes less than or equal to thirty. The value change triggers a transition to a new situation, *Breathing problem*, which then invokes a higher-priority plan to solve the breathing problem and reduce the value of oxygen, triggering a transition back to the original situation.
Pedagogy
Adele has been extended to support some additional instructional capabilities that Steve presently lacks. Using knowledge about both the student and the context she can intervene to present quizzes, as shown in Figure 6, provide pointers to relevant reference materials, and help motivate students by commenting on their progress.

Opportunistic learning
Situation-based reasoning can also be applied to the problem of recognizing pedagogical opportunities as a student works through a given task. For example, when a student orders a diagnostic test, a situation might provide Adele an opportunity to ask the student questions about the results of the test, as shown in Figure 6. Questions can be adapted to reflect to a student’s current understanding. Unlike domain-specific situations like the breathing-problem described above, there is no need for a plan to deal with these types of pedagogical opportunities.

However, by maintaining an awareness of the situation the agent can undertake situationally-appropriate interactions with the student, for example, allowing the student one or more chances to answer a question or asking follow-up questions. In this way situations are used to represent the knowledge that allows the pedagogical agent to react to changes in the state of the simulated world, not only for dealing with domain-specific conditions but also for exploiting pedagogical opportunities.

Figure 6. Adele instructs a student to answer a quiz when the student selects a urine dipstick test.
A situation object and a reference action are shown in Figure 7. The situation, showFNAReference, becomes current when its conditions are true. This occurs when the student schedules the patient for a fine needle aspiration (FNA), and it is the case that the student has not been told about the video on the FNA procedure. This type of situation is called a momentary situation because it simply triggers an event, unlike a quiz situation, below, which requires processing. Here, it triggers a refer action named FNAVideo, which causes Adele to comment to the student, in this case to tell them about a video that is available in the reference library. If a URL is specified, a refer action will also enable the agent persona’s show button; a click on the button will bring up a web page from which the video is accessible.

(situation showFNAReference
 :type momentary
 :condition (and (schedule-fna-done == T)
 (knowsAboutFNAVideo == false))
 :actions (:refer FNAVideo))

(refer FNAVideo
 :comment "Did you know that there is a video that describes how to do a fine needle aspiration in the reference library?"
 :effects ((knowsAboutFNAVideo = true))
 :url videos/fna.avi)

Figure 7. Description of a situation that triggers a reference action.

Giving a quiz is another example of how situation-based reasoning can be used for opportunistic learning. When the quiz situation askUrinalysisQuiz in Figure 8 is current, Adele will invoke a quiz named urinalysisQuiz and guide the student through it; she will introduce it, answer questions about it, and finally, give feedback on the answers chosen. A quiz action is more complex than a refer action, but it is described similarly. Adele currently supports single and multiple choice, and click-on-the-area type quizzes.

(situation askUrinalysisQuiz
 :type quiz
 :condition (and (order-urinalysis-done == T)
 (knowsAboutUrinalysis == false))
 :actions (:quiz urinalysisQuiz)
 :hint "Two of these options are appropriate."
 :rationale "The quiz tests your understanding.")

Figure 8. Description of a situation that triggers a quiz action.

Contextual references
Because Adele is part of an instructional system, we must support course authors who wish to provide references to materials that are relevant to the instructional goals. The materials may range from documents and images to videos and animations and are part of a Web-based reference library. In a typical scenario, a user who wished to find a particular document would go to the library’s home page
and initiate a search. While we cannot yet read a student’s mind, we, or at least Adele, can guess the
topic of the search based on the actions a student makes. For example, when a student begins to
examine a patient, Adele knows that the context is the physical examination, and she may know
more. If the student references the library at this time, he will find himself on a page that explains
how to do a physical examination. This is accomplished by a context tag in the task plan.

Figure 9. The evaluation on the left lists all unnecessary tests and their costs. The evaluation of the
diagnosis on the right hand side compares the incorrect diagnosis to the correct one.

Student assessment
Adele continuously monitors and records a student’s interaction with a simulation. During a
procedural task, Adele verifies that the task steps are done in the correct order and gives feedback as
described above. She also tracks the student’s responses to quizzes and referrals. We use a standard
overlay model to track the data and feed it back into the processing, similar to the one used in
previous tutoring work in the medical domain (Eliot & Woolf, 1995).

When a task is finished, Adele’s assessment module analyzes the student’s record and provides
domain-appropriate feedback. For example, in a clinical domain, Adele provides three types of post-
task assessment: 1) An evaluation of the diagnosis; 2) an evaluation of the diagnostic costs incurred,
and 3) an evaluation of the steps taken. See Figure 9. The diagnosis assessment uses information
about differential diagnoses, such as their confirmatory diagnostic tests, to comment on a correct
diagnosis, or to compare an incorrect diagnosis to the correct one, as shown in Figure 9. Different
domains will require different assessment modules, and Adele’s feedback will differ accordingly.

Implementation
To facilitate Web-based delivery, Adele is implemented in Java, making it possible to download
Adele-enhanced course modules over the Web. This approach offers long-term advantages, although
in the near term, incompatibilities between Java virtual machines make portability somewhat
difficult. High quality text-to-speech synthesis is platform-dependent, so variants of Adele are provided to take advantage of the text-to-speech synthesis capabilities available on each platform.¹ (Many of our problems with speech synthesis stem from our decision to implement Adele as a female character. Synthetic female voices in general have much lower quality than synthetic male voices.)

**Persona**

On the persona side, Adele has a repertoire of facial expressions and body postures that represent emotions, such as surprise and disappointment. These permit Adele to respond in a more lifelike fashion to student actions. On the instructional side, we have developed ways of building more instructional guidance into Adele’s interactions with the student. Based on a student’s action, Adele may choose to intervene, offering advice about what to do instead, e.g., “before you order a chest X-ray you should examine the condition of the lesion.” Alternatively, based on the context or action history, Adele can ask the student a ‘pop quiz’ question that must be answered before proceeding.

Adele’s animations are produced from two-dimensional drawings, instead of three-dimensional human figures. This makes it possible to run Adele on a variety of desktops, without relying upon specialized 3D graphics capability. The main drawback of 2D imagery is that it is difficult to compose behaviors, e.g., frown while looking to one side. We are experimenting with VRML browsers as a way of providing articulated human figures on a desktop setting. However, since adding a VRML browser adds complexity to the software installation, there will still be applications where 2D animations are preferable.

**Simulation interface**

Another implementation issue that influenced the design was the need to interface to externally authored simulations. Simulation authoring tools such as VIVIDS (Munro et. al, 1996) and Emultek’s RAPID are frequently used both to author simulation behavior and the simulation interface at the same time. For this reason, Adele was designed to run in a separate window, communicating with educational simulations using an interprocess communication link. We then developed behaviors for Adele to give the impression that she is integrated with the other displays running on the desktop, even though she runs in a separate window. When the student clicks on a button in the simulation window, Adele turns her head to look toward where the student clicked. She has a pointer that she can use to point toward objects in the other windows, similar to the pointer used by André’s PPP Persona (André et al. 1998). These behaviors partially compensate for the inability to actually manipulate objects directly, as Steve does.

**Current status**

We are currently developing three new cases for the medical domain; two are differential cases for our original learning unit on Myeloma, and the other is for a new unit on lung cancer. We are also adding support for multiple skill levels, so that Adele can be used both by medical students and by practicing physicians, and are implementing a persistent student model as part of our server-side work. Finally, we’ve begun a project with a group of software engineering students to create a user-
friendly system for knowledge acquisition that will automatically generate a task plan from forms authored by domain experts.

Meanwhile, we have initiated a new project in collaboration with the USC’s School of Gerontology to create additional problem-solving exercises for other life science courses, all centered on health care for aging populations. The first of these courses will address clinical problem solving in geriatric dentistry and will be ready for use in a course this spring.

**Evaluation**

Over the past year, we have worked closely with the medical faculty and students at the USC Medical School to create two course modules: one for clinical diagnosis problem solving and one for emergency room trauma care training. We acquired the necessary knowledge for the tasks using script-style forms that contain detailed descriptions of each step of a procedure. Our implementation progress was monitored at many intervals. In-house usability experts and medical students on research rotations evaluated the system’s usability and pedagogy, and comments from physicians have been solicited at public and private demonstrations. We are scheduled to test the system with a large group of medical students in October of 1998.

As expected, we received many comments about the interface, and have hired a creative director to address these issues. More interestingly, our evaluators found many procedural details that were overlooked, even after several iterations of editing by physicians and developers. Ordering constraints for the diagnostic lab tests, for example, were missing, because we didn’t explicitly ask our collaborators for these *for all levels* during the knowledge acquisition phase. Viewers readily accept Adele’s persona and we have found no clear advantage of 3D over 2D as far as user acceptance is concerned. However, finding the right level of realism has been difficult. Text-to-speech synthesis quality has been a critical factor for Adele, and we continue to experiment with different speech synthesis systems in order to arrive at an acceptable solution.

**Conclusion**

Adele and other pedagogical agents like her have lessons to offer regarding autonomous agent design. Researchers in believable agents argue that the audience’s perception of an agent’s competence is more important than the competence itself (Sengers 1998), and that competence is but one of many factors that agent authors should take into account (Reilly 1997). A key question is to what extent these claims are true for pedagogical agents. User feedback from Adele does indeed suggest that agent design must take the student’s perspective into account. Behaviors such as gaze shifting are essential in order to give students the impression that these agents are aware of them and understand them. Presentation details such as body posture, facial expression, and tone of voice have a big impact on students’ impressions of these agents. However, students differ from the typical “audiences” for believable agents in that they can engage agents in instructional dialogs. Giving students the ability to probe an agents’ knowledge requires a certain depth of knowledge on the part of the agent. The main lessons to be learned are: 1) Pedagogical agents need enough domain knowledge to support the anticipated instructional dialogs; 2) An agent’s behavior and appearance enhance the perception of expertise in the agent; 3) Users can react to agents in unexpected ways, so prototyping and experimentation are essential.
Acknowledgments

CARTE staff members Kate Labore and Dr. Jeff Rickel, ‘A’ Team members Ami Adler, Andrew Marshal, Anna Romero, and medical researcher Michael Hassler all contributed to the work presented here. Our collaborating organizations provided indispensable assistance, Drs. Allan Abbott, Demetrio Demetriatis, William La, Sidney Ontai, and Beverly Wood at the USC School of Medicine, and Carol Horwitz and Craig Hall at Air Force Research Laboratory. This work was supported by an internal research and development grant from the USC Information Sciences Institute.

References


