Final Submission 2006-08-07

An Intelligent Personal Assistant for Task and Time Management

Karen Myers[1] Pauline Berry[1] Jim Blythe[2] Ken Conley[1] Melinda Gervasio[1]

Deborah McGuinness[3] David Morley[1] Avi Pfeffer[4] Martha Pollack[5] Milind Tambe[6]

SRI International[1]
Menlo Park, CA
{firstname.lastname}@ sri.com

Stanford University[3]
Stanford, CA
dlm@ksl.stanford.edu

Univ. of Michigan[5]
Ann Arbor, MI
pollackm@eecs.umich.edu

USC/ISI[2]
Marina del Rey, CA
blythe@isi.edu

Harvard University[4]
Cambridge, MA
avi@eecs.harvard.edu

Univ. of Southern California[6]
Los Angeles, CA
tambe@usc.edu

Abstract

We describe an intelligent personal assistant that has been developed to aid a busy knowledge worker in managing time commitments and performing tasks. The design of the system was motivated by the complementary objectives of (a) relieving the user of routine tasks, thus allowing her to focus on tasks that critically require human problem-solving skills, and (b) intervening in situations where cognitive overload leads to oversights or mistakes by the user. The system draws on a diverse set of AI technologies that are linked within a Belief-Desire-Intention agent system. Although the system provides a number of automated functions, the overall framework is highly user-centric in its support for human needs, responsiveness to human inputs, and adaptivity to user working style and preferences.

Keywords: intelligent assistants, agents, task management, time management, personalization

Introduction

A typical knowledge worker must juggle a broad range of tasks and responsibilities. While doing so, she must maintain awareness of deadlines and resources, as well as tracking current activities and new information that could impact her objectives and productivity. Much of her work will require coordination and collaboration with a broad range of people, both within and outside of her immediate organization. As organizations seek to improve cost effectiveness and efficiency, workloads for many knowledge workers are increasing. Furthermore, workers are being inundated with vastly increased volumes of information that must be filtered and absorbed. The net result is high levels of cognitive overload in the work place (Kirsh 2000).

This paper describes a system, called the Project Execution Assistant (PExA), that has been developed to improve the productivity and effectiveness of a knowledge worker by aiding her in organizing and performing tasks. From a functional perspective, PExA focuses on two key areas: *time management* and *task management*. *Time management* refers to the process of helping a user manage actual and potential temporal commitments. Time management critically involves meeting/appointment scheduling, but further includes reminder generation and workload balancing. *Task management* involves the planning, execution, and oversight of tasks. Such tasks may be *personal* in that they originate with the user, or may derive from responsibilities associated with a project.

PExA has been designed to aid the user with tasks along a spectrum of complexity. Some of PExA's value derives from its ability to relieve the user of responsibility for frequently occurring, routine tasks (e.g., meeting scheduling, expense reimbursement). PExA can also assist the user with tasks that are larger in scope and less precisely defined (e.g., arranging a client visit).

PExA incorporates a significant body of sophisticated AI technologies for knowledge representation, reasoning (probabilistic and symbolic), planning, plan execution, agent coordination, adjustable autonomy, explanation, and learning. These technologies are integrated into a tightly coupled framework, drawing on a shared ontology and an agent architecture. This linkage has enabled a number of important capabilities within the system, including dynamic procedure learning, integrated task and calendar management, and real-time execution monitoring and prediction.

PExA has been developed as a key component of the CALO (Cognitive Agent that Learns and Organizes) project -- a large-scale effort to build an adaptive cognitive assistant situated in an office environment (see `http://caloproject.sri.com/` for details). Complementing PExA's capabilities for time and task management in CALO are a *meeting assistant* and an *information assistant*. The meeting assistant is designed to enhance a user's participation in a meeting through mechanisms that track the topics that are discussed, the participants' positions, and any resultant decisions. The information assistant provides tools to organize information within the user's desktop environment in order to support more efficient access and improved decision making (Cheyer et al. 2005).

The version of PExA described in this paper was delivered to the CALO project in 2006. Although only mid-way through the five-year project, PExA already provides a significant body of functional capabilities (task performance, time management, execution monitoring, team coordination) and qualities (advisability, explainability, adaptability) for an intelligent personal assistant. A version of PExA will be deployed for daily use within the research team later this year.

The paper is organized as follows. We begin by presenting the model of assistance underlying PExA. Next, we describe the overall PExA system and the core components for time and task management, as well as the main user interface for the system. After that, we present a use case that highlights select system functionality. We conclude with a discussion of issues for mixed-initiative systems, related work, and future technical directions.

Model of Assistance

Different styles of assistive technology suit different applications, depending on the types of problem to be addressed and the balance of expertise and knowledge between the user and the system. In situations where human problem-solving skills are weak or compromised in some way, an assistant can provide value by watching over the shoulder of the user and intervening to provide guidance when the user reaches impasses or makes mistake. Systems designed to aid people with cognitive disabilities such as memory decline fall into this category (Pollack 2005). When human cognitive skills are an essential part of the problem solving, a more appropriate design is to have an assistant relieve the user of routine tasks to enable her to focus on strategic decision making. Mixed-initiative planning technologies such as MAPGEN (Bresina et al. 2005) and PASSAT (Myers et al. 2003) provide examples of this type of assistant in the way that they support a human in making strategic planning decisions by aiding with the management of constraints and requirements. In situations where there is a distribution of problem-solving skills between the user and the system, a *collaborative* assistant would work in conjunction with its user to complete a shared task (Allen et al. 2002; Rich & Sidner 1998).

The demands of a busy office environment can often lead to situations where performance degrades as a result of information or task overload (Kirsh 2000). Typically, the user possesses the necessary problem-solving skills to perform her job effectively, but simply has too many things to do and too much information to track in the time available. Motivated by the objective of reducing user workload, we chose to adopt a *delegative* model of interaction between the user and the system (Myers & Yorke-Smith 2005). Within this model, the user decides what needs to be done and which tasks she feels comfortable allocating to the system. PExA works on behalf of its user by executing tasks that have been assigned to it, thus paralleling how a user might assign tasks to a well-trained human assistant. The system operates in a fairly autonomous manner within bounds set by the user, but interacts to solicit necessary information and to confirm important decisions.

Delegation is the primary interaction style within the system, but other forms of interaction can be incorporated as well. PExA currently includes several forms of proactive assistance in which it initiates communication with the user to inform her of problems, to provide reminders of user commitments, and to provide feedback on user requests. This proactive behavior is motivated primarily by information overload: in many situations, the system may have a better awareness than the user of requirements, commitments, or current state, and so can provide value by drawing on this knowledge within a given problem-solving context. Although not present within PExA currently, incorporation of a collaboration capability would be useful to address tasks that require the user and system to work together to solve problems.

Within the delegative interactions model, several desiderata were identified for interaction characteristics that the assistant would need in order for the technology to be both useful and usable by a busy knowledge worker.

- *Directable*: Although the assistant should be capable of operating in an autonomous manner, it must accept explicit directions from the user on what to do (or not), and how to do it.
- *Personalizable*: The assistant should learn a model of the user's preferences and adapt its behavior accordingly.
- *Teachable*: It should be easy for the user to communicate new or modified problem-solving knowledge to the assistant over time.
- *Transparent*: The assistant should be able to communicate succinctly what it is doing and why in order to provide the user with insight as to the status and strategy of its actions.

System Overview

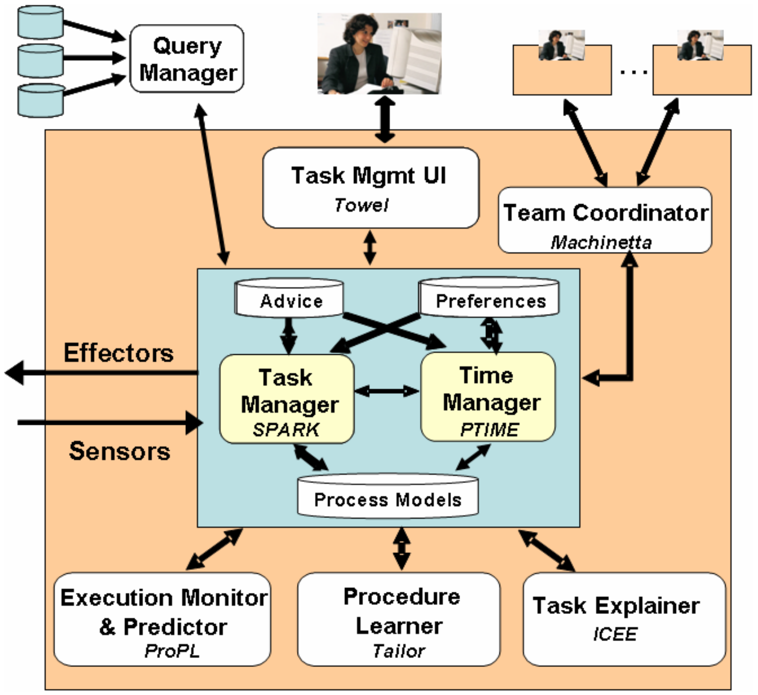


Figure 1. PExA Architecture

Figure 1 depicts the PExA architecture. Each PExA is associated with a single user but interacts with the PExAs of other users to exchange information and coordinate on tasks.

The heart of the system consists of the components for task and time management. These two elements are unified through the use of a common agent framework (the SPARK system, described below) and a shared ontology. Additional components within the system provide complementary capabilities for execution monitoring and prediction, procedure learning, task explanation, and team coordination. The different components within an individual PExA interact through an asynchronous messaging scheme to exchange data, queries, and requests to perform tasks (Cheyer & Martin 2001), resulting in a distributed and modular environment that simplified overall system development.

PExA's knowledge is distributed throughout the system. Key knowledge sources include a taxonomy of general office concepts, the user's personal calendar, an email server, a model of user preferences and a current situation model. The CALO Query Manager provides a unified interface to this knowledge through an integrated collection of reasoning and retrieval facilities (Ambite et al. 2006).

Although PExA will be deployed for operational use in the near future, it operates currently within a realistic but partially simulated office environment. While some effectors cause real actions to occur (e.g., calendar modifications, sending of email, information retrieval from the web), others are necessarily simulated (e.g., purchasing and conference registration actions).

Task Management

The role of the Task Manager is to organize, filter, prioritize, and oversee execution of tasks on behalf of the user. Tasks may be posed explicitly by the user or another PExA agent, or adopted proactively by the Task Manager in anticipation of user needs. The Task Manager can both perform tasks itself and draw on other problem-solving entities (including the user) to support task execution.

The Task Manager is built on top of a Belief-Desire-Intention (BDI) agent framework called SPARK (SRI Procedural Agent Realization Kit) (Morley & Myers 2004). SPARK embraces a procedural reasoning model of problem solving, in the spirit of earlier agent systems such as PRS (Georgeff & Ingrand 1989) and RAPS (Firby 1994). Central to SPARK's operation is a body of process models that encode knowledge of how activities can be undertaken to achieve objectives. The process models are represented in a procedural language that is similar to the hierarchical task network (HTN) representations used in may practical AI planning systems (Erol et al. 1994). However, the SPARK language extends standard HTN languages through its use of a rich set of task types (e.g., achievement, performance, waiting) and advanced control constructs (e.g., conditionals, iteration).

The Task Manager includes a library of process models (alternatively, plans or procedures) that provide a range of capabilities in the areas of visitor planning, meeting scheduling, expense reimbursement, and communication and coordination. The processes were designed primarily to automate capabilities but include explicit interaction points where the Task Manager solicits inputs from the user.

Rather than blindly accepting tasks from the user, the Task Manager analyzes requests to determine whether they are appropriate and feasible in the context of the user's current commitments and activities (Myers & Yorke-Smith 2005). For problematic requests, the system can make recommendations on how to eliminate the problems and work with the user to implement those recommendations. This type of proactive behavior on the part of the system can be useful in situations where the user may have limited awareness of the context in which tasks are to be performed, and hence may be posing requests that are unsuitable in certain ways. For example, if the user assigns the task of registering for a conference and the Task Manager determines that the user lacks sufficient travel funds, the system will notify the user and make suggestions to address the shortfall such as applying for a departmental travel grant or postponing a planned equipment purchase. The assessment of the resource feasibility of a task makes use of a capability to project necessary and sufficient bounds on resource usage by analyzing the space of possible hierarchical decompositions of a task (Morley et al. 2006).

SPARK also maintains a collection of *metalevel predicates* (or *metapredicates*) that track key internal state and processing steps of the system, such as what tasks it is performing, which procedures are being applied to those tasks, and what decisions have been made in the scope of executing a task. These metapredicates allow the system to reflect on and dynamically modify its behavior at runtime in order to adapt to evolving requirements; they are also used to support explanation (discussed further below).

Time Management

Time management within PExA is focused on aiding a user in managing their temporal commitments. Such time management is intensely personal, with many people being reluctant to relinquish control over their schedules. The PTIME component of PExA provides the user with personalized time-management support that is responsive to her needs and preferences, and adaptive to changing circumstances (Berry et al. 2006a). PTIME learns its user's preferences through a combination of passive learning, active learning, and advice taking. As a result, the user can become progressively more confident of PTIME's ability, and thus allow it to make increasingly more autonomous decisions, for example, whether to accept a meeting request, when to schedule a meeting, how much information to share when negotiating the time and location of a meeting with others, and when to issue reminders about upcoming deadlines.

PTIME is, in effect, a single-calendar scheduler that interacts with PTIME components of other PExA agents to coordinate shared calendar entities. In PExA, PTIME is used in an open, unbounded environment in which issues of privacy, authority, cross-organizational scheduling, and availability of participants are more significant than in prior work on automated calendar management. For example, (Franzin et al. 2002) assume complete privacy, while (Payne et al. 2002) and (Modi & Veloso 2004) assume more cooperative environments. With the exception of (Modi & Veloso 2004), most prior systems focus on requests formulated as hard constraints, and produce only feasible scheduling options. In contrast, PTIME treats the underlying scheduling problem as a soft constraint satisfaction problem; it makes use of individual preferences and the context of the user's current workload and deadlines (i.e., it does not just handle meetings, but also "to do" items), and it may interact with its user to find the best solution.

PTIME integrates commercial calendaring tools and user interface technology with new algorithms for constraint satisfaction, goal-directed process management, and preference learning. Scheduling requests are formulated in a highly expressive language that combines temporal and finite-domain constraints with disjunction and preferences. This degree of expressiveness is necessary to cover many real-world requirements, such as a preference to meet with a member of the sales department before noon or after 3:00 on Friday. Scheduling is accomplished through the use of powerful constraint satisfaction techniques designed to support this expressive language (Peintner & Pollack 2004; Berry et al. 2006b).

PTIME provides support for scheduling a single meeting, a set of related meetings, or an agenda of events. It also supports the user in canceling events, rescheduling events, requesting another person to commit to an event, and committing to a requested event. PTIME tracks its user's manipulations of her calendar to maintain schedule integrity and to alert her to consequences of her actions in the context of her current workload. PTIME also supports the exchange of information (availability, commitments) and requests between PExA agents. A SPARK-based controller manages the interactions between the user and the constraint reasoner, and interagent communication.

Execution Monitoring and Prediction

An intelligent assistant must be able to reason about the tasks it attempts to undertake: to determine whether a task is feasible, and to adapt to developments that occur during the course of execution. Determining whether a task is logically feasible given the user's current commitments is the responsibility of the Task Manager. However, even tasks that are logically feasible may be likely to fail because of uncertain future events. Furthermore, tasks that were likely to succeed at their initiation may become likely to fail because of unexpected events. It is the responsibility of the Execution Monitor and Predictor (EMP) component of PExA to reason about the course of a task's execution.

The EMP uses models written in ProPL, a probabilistic process modeling language (Pfeffer 2005). A ProPL model is a description of how a process evolves, including the specification of uncertainty over the success of subprocesses, their results, and execution times. ProPL models, which are derived from SPARK process models, are transformed into dynamic Bayesian networks in which there is a variable for every subprocess that could possibly be executed.

The Task Manager notifies the EMP of key events as they occur: the beginning and end of a subtask, the success or failure of a subtask, and the results produced by a subtask. At all points in time, the EMP maintains a probability distribution over the current state of the process using particle filtering (Doucet et al. 2001). The Task Manager and the EMP can interact in two ways. In *on demand* mode, the Task Manager can query the EMP for the probability that the task will succeed given the possible current states, or the expected time to completion. In *continuous* mode, the EMP runs at all time points: when the probability of success falls below a threshold, or it appears that the task will not complete by its deadline, the EMP notifies the Task Manager.

Team Coordination

Given the team-oriented nature of office activities, an effective intelligent assistant should be aware of other individuals and their assistants with whom the user needs to interact, and facilitate collaboration with them. PExA includes a capability for coordinating multiple PExA agents and their users based on the Machinetta framework for team coordination (Schurr et al. 2004). Machinetta, derived from the earlier STEAM (Tambe 1997) and TEAMCORE (Pynadath & Tambe 2003) coordination architectures, enables high-level team-oriented programming, thus avoiding the need to write tedious low-level coordination algorithms. Machinetta is used in PExA to provide team-level interagent communication capabilities, and to support dynamic task reallocation.

Dynamic task reallocation involves both the initial allocation of tasks within a team, and reallocation in response to anticipated problems in meeting deadlines. Machinetta employs a distributed constraint optimization algorithm for task allocation that draws on a model of user capabilities (Varakantam et al. 2005). Deciding when to reallocate tasks is difficult because of uncertainty both in observations of the user's task performance and in expectations of future user progress. Machinettta uses partially observable Markov decision processes (POMDPs) to enable task reallocation decisions despite such observational and transitional uncertainty. While the system must not autonomously make a reallocation decision in haste, it may have more current and detailed information about task dependencies and deadlines than the user, and must guard against situations where the user is unable to respond quickly. The POMDP policy enables PExA to adjust its own autonomy, asking the user if she will complete the task on time, and reallocating the task if she cannot or if she does not respond in a timely fashion.

Enhancing the User Experience

Layered on top of the core task and time management capabilities described in the previous section are functionalities that address the requirements of *directability*, *personalization*, *teachability* and *transparency*.

Directability

Directability within PExA is grounded in a mechanism for *advisability*. This mechanism allows the user to express guidance as to (a) the strategy to be used in solving a particular task or class of tasks, and (b) the scope of the system's autonomy (Myers & Morley 2001). Advice is expressed in a high-level language that gets operationalized into constraints that direct PExA's decision making at execution time.

The first class of advice can be used to designate or restrict procedures to be used, as well as to constrain how procedure parameters are instantiated. For example, the guidance "*Avoid rescheduling meetings with Bill*" expresses a preference over approaches for responding to scheduling conflicts. The directive "*Get approval for purchases from Ray or Tom*" restricts the choices for instantiating parameters that denote possible sources for approval.

The second class can be used to declare conditions under which an agent must obtain authorization from the user before performing activities (e.g., "*Obtain permission before scheduling any meetings after 5:00*"). As well, it can be used to designate a class of decisions that should be deferred to the user. These decisions can relate to either the selection of a value for parameter instantiation (e.g., "*Let me choose airline flights*") or the selection of a procedure for a task (e.g., "*Consult me when deciding how to respond to requests to cancel staff meetings*").

Personalization

Given that work styles are highly personal, an effective intelligent assistant must adapt to the preferences and characteristics of its user. Users may have preferences over a wide range of functions within the system, including how tasks are performed, how and when meetings are scheduled, and how the system interacts with the user. To date, the focus for personalization in PExA has been on the online, unobtrusive acquisition of user scheduling preferences through machine learning techniques.

In response to a meeting request, PTIME presents the user with a small number of candidate schedules. The user can select one from among that set, or request that the system generate additional candidates from which to choose. The learning exploits the preference information implicit in these user actions to produce a preference model in the form of an evaluation function over schedules. PTIME initially learned only general preferences over temporal constraints (i.e., preferences over days and times) in underconstrained situations (Gervasio et al. 2005). PTIME was extended recently to learn a multicriteria evaluation function that encompasses a number of additional factors, including meeting-specific temporal preferences, preferences over finite-domain constraints (e.g., meeting participants), preferences over global schedule characteristics (e.g., fragmentation, stability), and preferences over the scheduling preferences of other users. This expanded set of criteria improves PTIME's ability to capture the tradeoffs faced by users in real-world settings (Berry et al. 2006b).

PTIME employs *active learning* (Cohn et al. 1992) to inform the decision of which candidate schedules to present. In a typical active learning setting, the sole purpose is learning, thus no restrictions are imposed on the sets of candidate solutions presented to the user. However, PTIME's active learner is deployed within a functional scheduling system and thus must present reasonable solutions to actual meeting requests. To balance the often-competing requirements of exploring the solution space to benefit learning and of presenting desirable solutions to satisfy the user, the active learner uses a seeded entropy-based metric to find a diverse set of desirable solutions.

Initial results demonstrated that this learning approach can produce an accurate preference model for the user given a sufficient number of training examples. Informal studies have shown, however, that users have a low tolerance for bad scheduling suggestions, which are likely to occur during initial use of the system. To address this problem, PTIME was extended to provide a bootstrapping phase in which users can specify general preferences as well as certain types of tradeoffs between those preferences using a specialized graphical tool. Based on these explicitly stated preferences, PTIME induces a multicriteria evaluation function of the same form as the learned preference model. By combining these two functions, PTIME can provide reasonably good solutions early on while employing learning capabilities to refine the initial model to account for the unstated or evolving preferences of the user.

Teachability

The ability to expand and modify procedure knowledge at runtime is essential for a persistent problem-solving agent like PExA, which will need to incorporate new capabilities and adapt to changes in policy or expectations. The complexity of the procedures invoked by the system lies well beyond what current fully automated learning methods can synthesize. For this reason, we have focused on supporting a mixed-initiative model of procedure acquisition and extension.

PExA includes a tool called Tailor (Blythe 2005a; 2005b) that can modify PExA's process models in response to user instructions in the form of short sentences. Tailor can be used to correct problems with procedure knowledge detected at runtime or to create new procedures; additions or corrections can be dynamically loaded into the executor, thus enabling problem-solving behavior to adapt on the fly.

Tailor bridges the gap between the user's description of an intended change in behavior and a corresponding modification to the procedural knowledge and domain ontology. This is done through a combination of search for potential modifications and reasoning about their effects on PExA's global behavior. For example, given a user instruction *"Prepay with my credit card when the fee is less than $500"*, Tailor identifies the affected action from those that are part of PExA's current tasks, uses search to find expressions corresponding to terms like 'the fee' and possibly omitted action parameters or objects, simulates one or more potential modifications to test their effect on other parts of the plan, and presents the most promising choices to the user for confirmation. Testing has shown that users can add new steps, modify conditions and change step orderings using Tailor without knowledge of PExA's procedure representation or precise domain ontology.

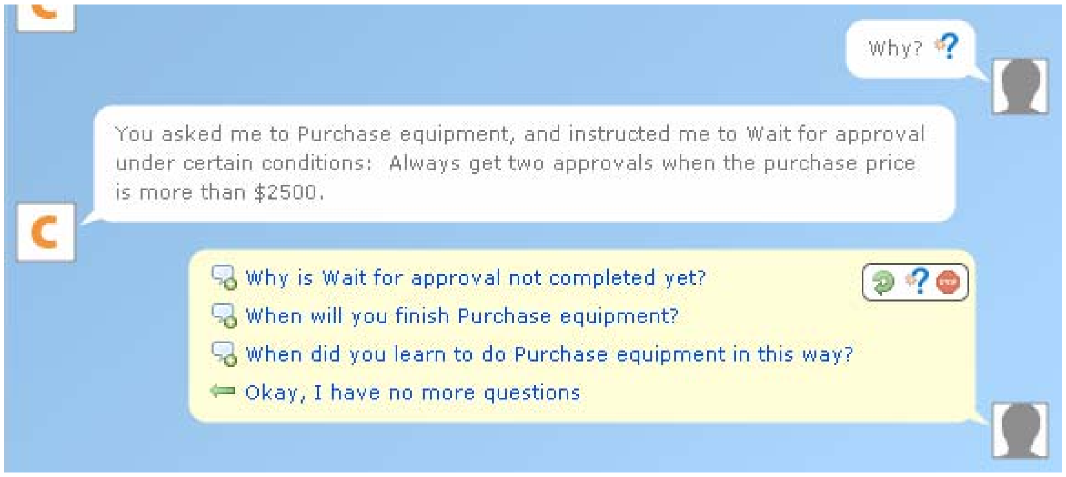
Transparency

For users to embrace personal assistant software, they need to have access to explanations as to why decisions are made and actions taken, and what information and processes those actions and decisions depend on. PExA has been designed with this requirement in mind and provides explanation capabilities through its Integrated Cognitive Explanation Environment (ICEE).

ICEE is capable of explaining a variety of questions including why PExA is currently performing a task, why the task is not yet finished, what information PExA relies on, and how PExA will execute something. When an explanation is requested, ICEE chooses one of a number of strategies to answer the question, depending upon context and a user model. For example, one strategy for answering the question *"Why are you doing <task>?"* leverages provenance information about the source of the task process, including processes that have been modified through various learning methods, and produces the explanation "*I am trying to do <high_level_task> and you instructed me to do <task> when <condition> holds.*" Other strategies for this question include exposing preconditions or termination conditions, revealing meta-information about task dependencies, showing task hierarchies and abstractions, and explaining further provenance information related to task preconditions or other task knowledge.

ICEE relies on the Inference Web infrastructure for explaining distributed systems (McGuinness & Pinheiro da Silva 2004) and leverages the Proof Markup Language (PML) proof interlingua (Pinheiro da Silva et al. 2005) for representing justifications. To produce the explanation, ICEE generates PML from the SPARK metapredicates (described earlier) that track which processes are executing, what goals they are servicing, and what termination conditions are not met. ICEE uses a dialog model and Inference Web components to present relevant information and possible context-appropriate follow-up questions for the user to ask. Follow-up questions might include requests for additional detail, clarifying questions about an explanation that has been provided previously, or requests to employ an alternate strategy for answering a previously posed question. For example, possible follow-up questions to the explanation given above are *"Why is <task> not yet completed?"* and *"When will you finish <task>?"*.

Figure 2 presents a sample dialogue in which the user has asked for the motivation for a particular task (e.g., it is waiting for PExA to complete the purchase of a piece of office equipment requested by the user). Relevant follow-up questions in this case ask why the task has not yet completed, when the high-level task will complete, and when the system learned the process it is executing for the high-level task.


Why?

You asked me to Purchase equipment, and instructed me to Wait for approval under certain conditions: Always get two approvals when the purchase price is more than $2500.

Why is Wait for approval not completed yet?
When will you finish Purchase equipment?
When did you learn to do Purchase equipment in this way?
Okay, I have no more questions

Figure 2. An example explanation dialog, with suggested follow-up questions.

User Interaction

User interactions for specifying, managing, tracking, explaining, and collaborating on tasks are made through a unified user interface (UI). In this way, the diversity and complexity of the underlying AI technologies are hidden from the user. Additional specialized interfaces are provided for viewing and modifying calendar information, and for procedure modifications in Tailor, as those activities require detailed interactions that are not well suited to a general-purpose interface.

The interface, called Towel, uses a lightweight, peripheral display. The primary window (see Figure 3 – left side) summarizes the user's tasks and provides visual representations of task metadata such as deadlines and priority. Towel provides a variety of mechanisms to organize, search, prioritize and view tasks, including tagging, property assignment, and grouping capabilities. Associated with each task in Towel is a rich body of information about task provenance (source, time of creation), requirements (e.g., deadlines, priority), current state, and relations to other tasks (semantic groupings, task/subtask relations).

Towel draws on ideas from prior systems designed to support 'to do' management (e.g., (Bellotti et al. 2004)); however, the emphasis within those systems is to help the user track and prioritize tasks that she needs to complete. Towel goes beyond such systems through its support for (a) the delegation of tasks to other PExA-enabled teammates, and (b) the dispatching of tasks for automated execution by the Task Manager.

For task dispatching, a user can assign a task to the Task Manager for execution by selecting from a menu of automated capabilities. Alternatively, the user can define a task informally by providing a textual description of it (e.g., "Schedule project review"); Towel will provide the user with a list of possible tasks that it believes it could perform to help accomplish this task (see Figure 3 – right side). Currently, this list is generated through simple keyword matching on task and procedure descriptions, but more sophisticated mechanisms based on semantic matching will be employed in the future.

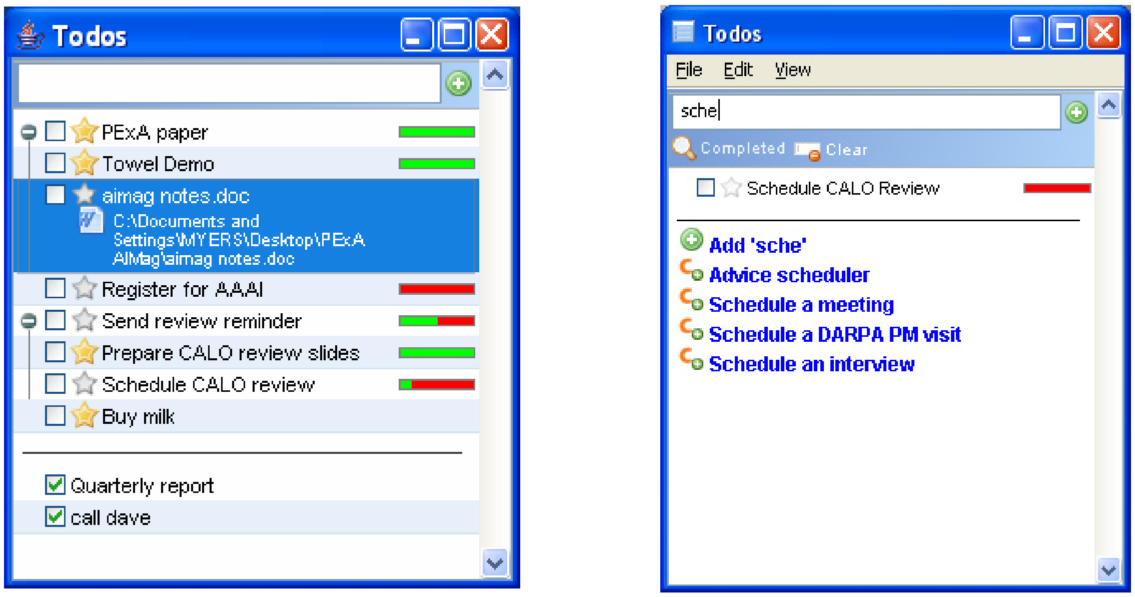
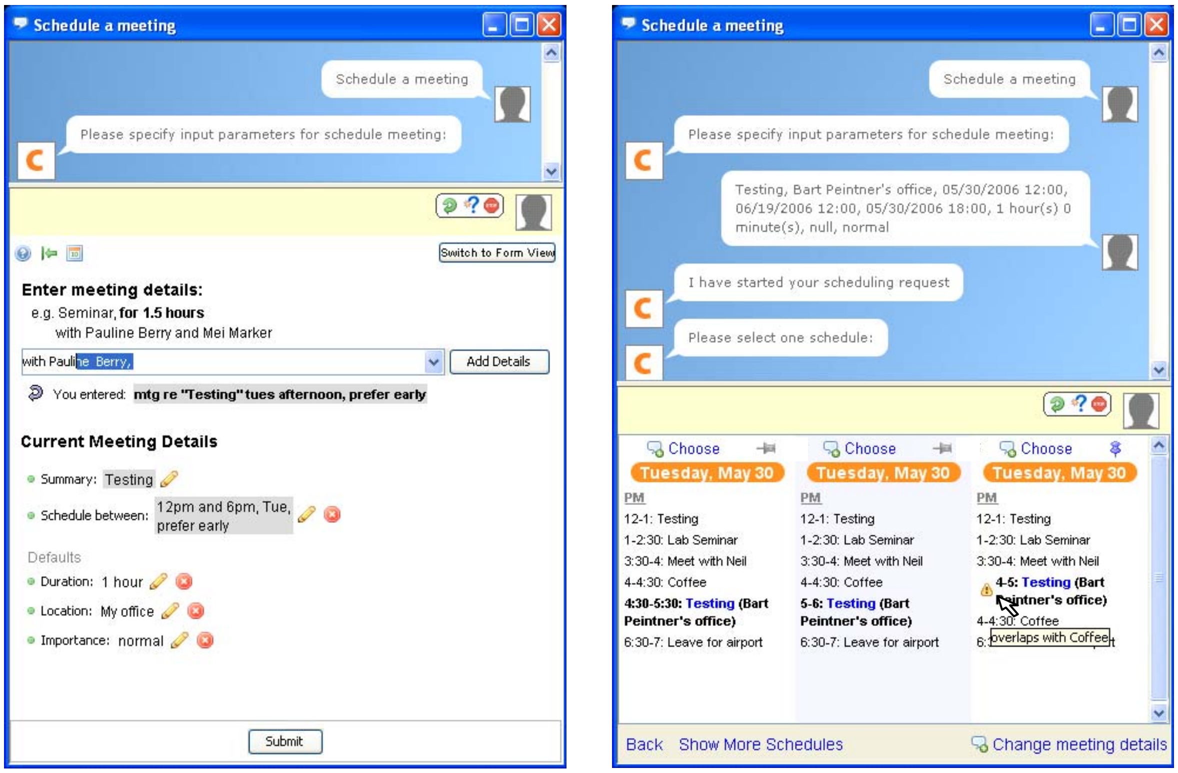


Figure 3. The Towel ToDo Manager User Interface: task list (left) and task specification (right) displays.


Schedule a meeting

Schedule a meeting

Please specify input parameters for schedule meeting:

Switch to Form View

Enter meeting details:
e.g. Seminar, **for 1.5 hours**
with Pauline Berry and Mei Marker

with Pauline Berry,

Add Details

You entered: **mtg re "Testing" tues afternoon, prefer early**

Current Meeting Details

- Summary: Testing
- Schedule between: 12pm and 6pm, Tue, prefer early

Defaults
- Duration: 1 hour
- Location: My office
- Importance: normal

Submit

Schedule a meeting

Schedule a meeting

Please specify input parameters for schedule meeting:

Testing, Bart Peintner's office, 05/30/2006 12:00, 06/19/2006 12:00, 05/30/2006 18:00, 1 hour(s) 0 minute(s), null, normal

I have started your scheduling request

Please select one schedule:

Choose	Choose	Choose
Tuesday, May 30	Tuesday, May 30	Tuesday, May 30
PM	PM	PM
12-1: Testing	12-1: Testing	12-1: Testing
1-2:30: Lab Seminar	1-2:30: Lab Seminar	1-2:30: Lab Seminar
3:30-4: Meet with Neil	3:30-4: Meet with Neil	3:30-4: Meet with Neil
4-4:30: Coffee	4-4:30: Coffee	**4-5: Testing (Bart Peintner's office)**
4:30-5:30: Testing (Bart Peintner's office)	**5-6: Testing (Bart Peintner's office)**	4-4:30: Coffee
6:30-7: Leave for airport	6:30-7: Leave for airport	6:[illegible]t

overlaps with Coffee

Back Show More Schedules Change meeting details

Figure 4. Sample task collaboration window in which the user first defines a scheduling task (left side) and then explores and selects a solution (right side).

The task list UI within Towel provides a summary view of tasks and access to common operations for manipulating and viewing them. Given the temporally extended nature of many of the tasks that the Task Manager can perform on the user's behalf, Towel further includes a *collaboration window* for supporting ongoing interactions between the user and the Task Manager for an individual task (Figure 4). In particular, each task has an associated collaboration window that is used solely for communication related to that task. An earlier design placed all interactions within a common window: while that approach provided a convenient, single place for interaction, test users were confused by conversations about multiple tasks occurring within the same dialog. The design for the collaboration window was inspired by the user interface of the DiamondHelp system for collaborative task guidance (Rich et al. 2006).

The collaboration windows also support the use of specialized displays tied to specific functions, to simplify certain types of interaction (similar in design to). For example, the bottom portions of the two windows in Figure 4 display specialized interaction frames for soliciting detailed information about a meeting request (left side) and exploring alternative solutions (right side). The specialized frame for entering meeting requests supports an expressive, albeit restricted, natural language input to enable users to quickly and naturally specify requirements and preferences. In contrast, traditional form-based input interfaces for calendar systems are ineffective for communicating constructs such as constraints and preferences. Although specification of preferences such as *"prefer Tuesday at 2pm"*, *"Bob is optional"*, and *"can overlap"* is possible with dialog form widgets, the inclusion of appropriate widgets for all possible constraints and preferences would result in an overly complex interface; furthermore, such interfaces have been shown to elicit untrue preferences (Viappiani 2006). In contrast, the specialized graphical display for exploring candidate solutions helps the user understand the key differences among schedules, refine or relax input constraints, explore alternative solutions, and select a preferred candidate. The warning sign seen in the right-most schedule indicates that the constraint that meetings not overlap had to be relaxed (as indicated by the tool tip *"overlaps with Coffee"*). If existing meetings must be moved (which is not the case here), the display would indicate whether the effort required to reschedule is high, medium, or low.

Use Case

We present a sample use case that highlights select PExA capabilities.

Scene 1: Visitor scheduling
Helen receives an email request from the sponsor of one of her projects to visit next Wednesday for a review. Helen asks PExA to initiate the planning process. Her PExA starts arranging a tentative agenda based on its knowledge of typical requirements of such visits. It identifies a set of project demonstrations and presentations for the meeting based on its knowledge of project activities; the agenda also includes time for lunch, coffee breaks and a private meeting with Helen at the end of the day. PExA determines various constraints for the meeting, including durations, participants, and a range of start/end times for each agenda item. Helen's PExA interacts with the PExAs of the other project members to finalize the agenda (i.e., get schedules, update their calendars).

PExA is unable to find an agenda that satisfies all requirements, so it presents three candidate solutions to Helen that relax the scheduling problem in various ways. Helen indicates that none of the solutions is satisfactory. PExA then presents a set of strategies for simplifying the scheduling problem (e.g., drop a presentation, shorten presentation times, extend the day, change participants). Helen suggests (a) dropping one of the demonstrations and (b) getting Dave to replace Joe. PExA produces a new set of candidate agendas based on the recommendation and Helen selects one. PExA distributes the agenda to the relevant people (i.e., project members, sponsor) and assigns tasks to members of the project team to prepare for the visit.

Scene 2: Conference Registration
While preparations for the visit continue, Helen asks PExA to register her for the AAAI conference. PExA warns Helen that she has insufficient discretionary funds for the trip. PExA suggests that Helen could afford the trip if she: (a) cancels her ICAPS trip, (b) postpones the planned purchase of a laptop, or (c) applies for a departmental travel grant. Helen asks PExA to apply for the grant; she has high certainty that the grant will be approved, so asks PExA to continue the registration process. Registration requires payment of the conference fee. PExA knows several payment strategies; the preferred option (based on prior advice from Helen) is for the company to prepay, if time is available. PExA determines that there is sufficient time and requests that procurement initiate the payment.

Time progresses but PExA does not receive confirmation of the prepayment from procurement. PExA eventually decides that the risk of missing the prepayment deadline is too high. PExA recommends to Helen that it perform the prepayment directly using Helen's credit card; Helen agrees, so PExA makes the payment and cancels the corporate prepayment. PExA also initiates the expense reimbursement process for the registration fee.

Scene 3: Three Days Before the Visit
Three days before the visit, Helen asks PExA why she has not yet received reimbursement for the registration fee. PExA reminds Helen that the reimbursement process for conference fees typically takes five working days, but the request was submitted only three days ago.

Scene 4: Sixty Hours Prior to the Visit
PExA monitors progress by the team members on the tasks associated with the visit. It notices that Dave is behind schedule in his demonstration tasks; furthermore, Dave's calendar indicates that he is heavily committed for the next two days. PExA infers that Dave is behind schedule, and assigns one of his two demonstration tasks to Alice.

Scene 5: One Day Before the Visit
PExA informs Helen that the reimbursement for the conference fee should have arrived but has not. PExA sends email to Travel Audit to check on the situation.

Scene 6: Visit Day
The visit is scheduled to start at 8:00 in the morning. At 8:15, Helen receives a page from the sponsor indicating that he will not arrive until 10:00. Helen informs PExA of the delay. PExA attempts to reschedule the agenda but cannot accommodate everything. PExA works with Helen to reschedule the agenda and then notifies the participants of the changes.

Scene 7: The Day After the Visit
PExA informs Helen that her reimbursement check is available.

Issues for Mixed-initiative Assistants

The introduction to this special issue of the AI Magazine identifies a set of design issues for mixed-initiative assistants. In this section, we discuss how these issues are addressed within PExA.

Tasking. Tasking focuses on the division of responsibility between the human and the assistant. As noted above, PExA embodies a delegative model of assistance in which the user first determines what needs to be done and then explicitly assigns appropriate tasks to the system. The system can operate in a fairly autonomous manner but interacts with the user to obtain necessary inputs and to validate important decisions. This design enables the user to delegate responsibility for routine or less important tasks to the system, thus increasing the amount of time that the user can dedicate to more cognitively demanding activities.

Control. The *raison d'etre* for PExA is to serve the needs of its user. For this reason, the system has been designed to provide the user with a high degree of control over system behavior: the delegative model provides control over what the system *does*, while user advice provides control over *how* the system does things. Nevertheless, the system retains a strong measure of autonomy. Subsequent to the assignment of tasks by the user, the system and the user address their individual responsibilities in a fairly independent manner, initiating interactions with the other as needed. Interactions typically focus on eliciting information from the other party that would contribute to problem solving. In addition, the user may ask questions to develop an understanding of what the system is doing and why. Similarly, the system has the ability to take initiative in terms of asking questions of the user and acting proactively in areas such as reminder generation and dynamic task reallocation. In this regard, the user and the system could be characterized as engaging in *cooperative* problem solving, in contrast to the collaborative problem-solving style of (Allen et al. 2002; Rich & Sidner 1998).

Awareness. Because the user and the system operate in a loosely coupled manner, it is unnecessary for either to have full awareness of what the other is doing. Shared awareness is necessary only for aspects of mutual interest. Much of the shared awareness in the system is grounded in the to do interface. From the perspective of the system, the to do interface provides a characterization of (at least some of) what the user wants to accomplish along with associated priorities and deadlines. From the perspective of the user, the to do interface summarizes current system actions and status. As desired, the user can access additional information from the system about its current and projected activities through the explanation capabilities.

The use of activity recognition technology to determine what the user is doing, based on her recent actions, would be a welcome addition to PExA. In particular, activity recognition could be used both to identify opportunities where the system could intervene to assist the user with a given task and to improve how and when the system interacts with the user. Several activity recognition efforts are underway within the CALO project; currently, we are exploring how to capitalize on these capabilities within PExA.

Personalization. As noted above, personalization has been a major focus of the PExA project in order to ensure that the technology can be used in a way that complements individual work styles. To date, our focus for personalization has been on learning preferences for meeting scheduling. One important area for future work is to learn preferences for interactions with the user.

Evaluation. Two factors complicate the evaluation of mixed-initiative technologies. First, evaluation must assess the combined performance of the human with the system rather than the system itself, thus introducing the need for expensive and time-consuming user studies. Second, mixed-initiative technologies are of appeal in domains for which full automation is not possible, typically because of the complexities and scope of the problems to be solved. In particular, there is often a broad range of capabilities that would need to be assessed rather than an easily isolated piece of functionality. Another complication derives from the fact that many such complex domains lack objective standards for performance assessment, as measures of result quality tend to be highly subjective. For these reasons, detailed evaluations of mixed-initiative technologies are rare.

Although we have conducted some informal user studies of portions of the overall system, no comprehensive evaluation has yet been undertaken. As we introduce PExA into daily use within our research team, we intend to collect data to determine which aspects of the system work well for users, aiming eventually for a large-scale user study to asses the merits of the technology.

Related Work

The work on intelligent assistants most similar to PExA is the Electric Elves project (Chalupsky et al. 2002), which provided personal assistants for a team of about twelve researchers. Key functionalities included rescheduling meetings, selecting presenters for meetings, and ordering meals. While Electric Elves provided an early model of how teams of personal assistants could be productively employed in office environments, the individual agents were significantly limited in the services that they provided to individual users compared to PExA, for example, lacking capabilities of learning for personalization, detailed time management, and explanation.

There has been much recent work on assistive technologies to aid people with memory decline or other cognitive disabilities in managing their daily activities (Pollack 2005, Haigh et al. 2004). Unlike PExA, these systems do not perform tasks for the user; instead, they monitor a person's actions in order to understand the user's goals, and then determine reminders and information that would be helpful to the person in achieving those goals.

The Lumière system (Horvitz et al. 1998) provides intelligent assistance to software users. Lumière focuses on understanding user goals and needs in order to provide proactive assistance. It does not provide the kinds of task delegation capabilities that lie at the core of PExA.

Summary

PExA constitutes an ambitious effort to develop an intelligent personal assistant that can increase the productivity of a busy knowledge worker through its support for task and time management. The demands of the office environment can often lead to situations where information and task overload lead to reduced performance. For these reasons, the system focuses on performing routine tasks delegated to it by the user to enable the user to address more cognitively challenging activities, while also offering proactive assistance in situations where it has knowledge or insights that the user lacks.

PExA draws on a diverse set of AI technologies, linked together within a BDI-based agent framework. The use of these AI technologies enables a number of desirable qualities for the assistant, including personalizability, directability, teachability, and transparency of operations. Although the system provides a number of automated functions, the overall framework is highly user centric in its support for human needs, responsiveness to human inputs, and adaptivity to user working style and preferences.

One critical driver for the project is to develop a system that will be adopted for regular use. The time management component of the system was recently deployed to a small set of users within the research team, while the Task Manager will be deployed over the next few months. To support the Task Manager deployment, various simulated effectors within the system are to be replaced by connections to corporate web services (e.g., for filing expense reports or requesting travel authorization), thus enabling users to request the system to perform real-world tasks on their behalf.

During the remaining two years of the CALO project, work on PExA will focus on increasing the overall utility and flexibility of the system. One key technical thrust will be to provide additional forms of procedure learning to complement the learning by instruction framework within Tailor. In particular, we will be incorporating into PExA procedure learning capabilities grounded in the methodologies of learning by demonstration and learning by discussion. A second thrust will be on increasing the scope of proactive behavior of the system through the use of reflective reasoning. Although the current system can take initiative for interacting with the user (for reminder generation, critiquing of inappropriate task requests, communication of important events, soliciting user inputs), these capabilities have been explicitly engineered into the system. More generally, an intelligent assistant should have the ability to deliberate over a general theory of its capabilities and its user's objectives to determine when it should take proactive measures to assist the user.

Acknowledgments. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010. The authors thank the following individuals for their contributions in conceiving and developing the PExA system: Victoria Bellotti, Hung Bui, Jim Carpenter, Cynthia Chang, Adam Cheyer, Li Ding, Tom Garvey, Alyssa Glass, Bill Mark, Mei Marker, Michael Moffitt, Jonathan Pearce, Bart Peintner, Ray Perrault, Shahin Saadati, Paulo Pinheiro da Silva, Peter Schwartz, Jim Thornton, Mabry Tyson, Pradeep Varakantham, Julie Weber, Michael Wolverton, and Neil Yorke-Smith.

References

Allen, J., Blaylock, N., and Ferguson, G. 2002. A Problem Solving Model for Collaborative Agents. In Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy.

Ambite, J.L., Chaudhri, V., Fikes, R., Jenkins, J., Mishra, S., Muslea, M., Uribe, T., and Yang, G. 2006. Design and Implementation of the CALO Query Manager. In Proceedings of the 18th Innovative Applications of Artificial Intelligence Conference, Boston, MA.

Bellotti, V., Dala, B., Good, N., Bobrow, D, and Ducheneaut, N. 2004. What a To-Do: Studies of Task Management Towards the Design of a Personal Task List Manager. In Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI2004), Vienna, Austria, 735-742.

Berry, P., Conley, K., Gervasio, M., Peintner, B., Uribe, T., and Yorke-Smith, N. 2006a. Deploying a Personalized Time Management Agent. In Proceedings of the Fifth International Conference on Autonomous Agents and Multiagent Systems, Hakodate, Japan.

Berry, P., Gervasio, M., Peintner, B., Uribe, T., and Yorke-Smith, N. 2006b. Multi-Criteria Evaluation in User-Centric Distributed Scheduling Agents. In Proceedings of the AAAI Spring Symposium on Distributed and Schedule Management, Stanford, CA.

Blythe, J. 2005a. Task Learning by Instruction in Tailor. In Proceedings of the International Conference on Intelligent User Interfaces, San Diego, CA.

Blythe, J. 2005b. An Analysis of Procedure Learning by Instruction. In Proceedings of the 20th National Conference on Artificial Intelligence, Pittsburgh, PA.

Bresina, J., Jonsson, A., Morris, P. and Rajan, K. 2005. Activity Planning for the Mars Exploration Rovers. In Proceedings of the 15th International Conference on Automated Planning and Scheduling, Monterey, CA.

Chalupsky, H., Gil, Y., Knoblock, C., Lerman, K., Oh, J., Pynadath, D., Russ, T., and Tambe, M. 2002. Electric Elves: Applying Agent Technology to Support Human Organizations, AI Magazine, 23(2): 11-24.

Cheyer, A. and Martin, D. 2001. The Open Agent Architecture. Journal of Autonomous Agents and Multi-Agent Systems, 4(1/2):143-148.

Cheyer, A., Park, J., and Giuli, R. 2005. IRIS: Integrate. Relate. Infer. Share. In Proceedings of the Workshop on the Semantic Desktop, International Semantic Web Conference, Galway, Ireland.

Cohn, D., Atlas, L., and Ladner, R. 1992. Improving Generalization with Active Learning. Machine Learning, 15 (2): 201-221.

Doucet, A., de Freitas, N. and Gordon, N. 2001. Sequential Monte Carlo Methods in Practice: Springer.

Erol, K., Hendler, J., and Nau, D. 1994. Semantics for Hierarchical Task-Network Planning. Technical Report CS-TR-3239, Computer Science Department, University of Maryland.

Firby, J. 1994. Task Networks for Controlling Continuous Processes. In Proceedings of the Second International Conference on AI Planning Systems, Chicago, IL.

Franzin, M. S., Freuder, E. C., Rossi, F., and Wallace, R. 2002. Multi-Agent Meeting Scheduling with Preferences: Efficiency, Privacy Loss and Solution Quality. In Proceedings of the AAAI Workshop on Preference in AI and CP, Edmonton, Canada.

Georgeff, M. and Ingrand, F. 1989. Decision-Making in an Embedded Reasoning System. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Detroit, MI.

Gervasio, M., Moffitt, M., Pollack, M., Taylor, J., and Uribe, T. 2005. Active Preference Learning for Personalized Calendar Scheduling Assistance. In Proceedings of the International Conference on Intelligent User Interfaces, San Diego, CA.

Haigh, K., Kiff, L., Myers, J., Guralnik, V., Geib., C., Phelops, J., and Wagner, T. 2004. The Independent LifeStyle Assistant: AI Lessons Learned. In Proceedings of 16th Innovative Applications of Artificial Intelligence Conference, San Jose, CA.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K. 1998. The Lumière Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In Proceedings of the 14th Conference on Uncertainty in AI, Madison, WI.

Kirsh, D. 2000. A Few Thoughts on Cognitive Overload. Intellectica, 19-51.

McGuinness, D. L. and Pinheiro da Silva, P. 2004. Explaining Answers from the Semantic Web: The Inference Web Approach. Journal of Web Semantics 1(4), 397-413.

Modi, J. andVeloso, M. 2004. Multiagent Meeting Scheduling with Rescheduling. In Proceedings of the Fifth Workshop on Distributed Constraint Reasoning, Toronto, Canada.

Moffitt, M. D., Peintner, B., and Pollack, M. E. 2005. Augmenting Disjunctive Temporal Problems with Finite-Domain Constraints. In Proceedings of the 20th National Conference on Artificial Intelligence, Pittsburgh, PA.

Morley, D. and Myers, K. 2004. The SPARK Agent Framework. In Proceedings of the Third International Conference on Autonomous Agents and Multiagent Systems, New York, NY.

Morley, D., Myers, K., and Yorke-Smith, N. 2006. Continuous Refinement of Agent Resource Estimates. In Proceedings of the Fifth International Conference on Autonomous Agents and Multiagent Systems, Hakodate, Japan.

Myers, K. L., Jarvis, P. Tyson, W. M., and Wolverton, M. J. 2003. A Mixed-initiative Framework for Robust Plan Sketching. In Proceedings of the 13th International Conference. on Automated Planning and Scheduling, Trento, Italy.

Myers, K., and Morley, D. 2001. Human Directability of Agents. In Proceedings of the First International Conference on Knowledge Capture, Victoria, Canada.

Myers, K. and Yorke-Smith, N. 2005. A Cognitive Framework for Delegation to an Assistive User Agent. In Proceedings of the AAAI Fall Symposium on Mixed-Initiative Problem Solving Assistants, Arlington, VA.

Palen, L. 1999. Social, Individual and Technological Issues for Groupware Calendar Systems. In Proceedings of the Conference on Human Factors in Computing Systems (CHI-99), Pittsburgh, PA, 17-24.

Payne, T. R., Singh, R., and Sycara, K. 2002. RCAL: A Case Study on Semantic Web Agents. In Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy.

Peintner, B. and Pollack, M. 2004. Low-Cost Addition of Preferences to DTPs and TCSPs. In Proceedings of the 19th National Conference on Artificial Intelligence, San Jose, CA.

Pfeffer, A. 2005. Functional Specification of Probabilistic Process Models. In Proceedings of the 20th National Conference on Artificial Intelligence, Pittsburgh, PA.

Pinheiro da Silva, P., McGuinness, D. L., and Fikes, R. 2006. A Proof Markup Language for Semantic Web Services. Information Systems 31(4-5): 381-395.

Pollack, M. 2005. Intelligent Technology for an Aging Population: The Use of AI to Assist Elders with Cognitive Impairment. AI Magazine 26(2): 9-24.

Pynadath, D.V. and Tambe. M. 2003. Automated Teamwork among Heterogeneous Software Agents and Humans. Journal of Autonomous Agents and Multi-Agent Systems 7:71–100.

Rich, C. and Sidner, C. 1998. COLLAGEN: A Collaboration Manager for Software Interface Agents. User Modeling and User-Adapted Interaction 8(3/4): 315-350.

Rich, C., Sidner, C., Lesh, N., Garland, A., Booth, S., and Chimani, M. 2006. DiamondHelp: A New Interaction Design for Networked Home Appliances. Personal and Ubiquitous Computing 10(2): 187-190.

Schurr, N., Okamoto, S., Maheswaran, R., Tambe, M., and Scerri, P. 2004. Evolution of a Teamwork Model. In Cognitive Modeling and Multi-Agent Interactions, ed. R. Sun. Cambridge University Press, U.K.

Tambe, M. 1997. Towards Flexible Teamwork. Journal of Artificial Intelligence Research, 7:83–124.

Varakantam, P., Maheswaran, R., and Tambe, M. 2005. Exploiting Belief Bounds: Practical POMDPs for Personal Assistant Agents. In Proceedings of the Fourth International Conference on Autonomous Agents and Multiagent Systems, Utrecht, Netherlands.

Viappiani, P. Faltings, B., and Pu, P. 2006. Evaluating Preference-based Search Tools: A Tale of Two Approaches. In Proceedings of the 21st National Conference on Artificial Intelligence, Boston, MA.