International Journal on Artificial Intelligence Tools
Vol. 21, No. 1 (2012) 1250004 (30 pages)

DOI: 10.1142/S0218213012500042

THE DESIGN OF A PROACTIVE PERSONAL AGENT FOR TASK MANAGEMENT*

NEIL YORKE-SMITH

Olayan School of Business, American University of Beirut
Riad El-Solh, Beirut 1107-2020, Lebanon
and
Artificial Intelligence Center, SRI International
333 Ravenswood Ave, Menlo Park, CA 94025, USA
nysmith@AI.SRI.COM

SHAHIN SAADATI

Artificial Intelligence Center, SRI International
333 Ravenswood Ave, Menlo Park, CA 94025, USA
saadati@AI.SRI.COM

KAREN L. MYERS

Artificial Intelligence Center, SRI International
333 Ravenswood Ave, Menlo Park, CA 94025, USA
myers@AI.SRI.COM

DAVID N. MORLEY

Artificial Intelligence Center, SRI International
333 Ravenswood Ave, Menlo Park, CA 94025, USA
morley@AI.SRI.COM

Received 11 January 2011
Accepted 11 October 2011

Personal assistant agents capable of proactively offering assistance can be more helpful to their users through their ability to perform tasks that otherwise would require user involvement. This article characterizes the properties desired of proactive behavior by a personal assistant agent in the realm of task management and develops an operational framework to implement such capabilities. We present an extended agent architectural model that features a meta-level layer charged with identifying potentially helpful actions and determining when it is appropriate to perform them. The reasoning that answers these questions draws on a theory of proactivity that describes user desires and a model of helpfulness. Operationally, *assistance patterns* represent a compiled form of this knowledge, instantiating meta-

*Authors listed in reverse alphabetical order.

reasoning over the agent's beliefs about its user's activities as well as over world state. The resulting generic framework for proactive goal generation and deliberation has been implemented as part of a personal assistant agent in the computer desktop domain.

Keywords: Assistive agents; proactivity; task management.

1. Introduction

An enduring vision is the intelligent personal assistant agent that can aid people in their day-to-day lives.[8,57] Over the last 20 years, research has explored personalized assistance in domains as diverse as eldercare,[78] office applications,[42] military command and control,[74] and recreation and tourism.[9,1]

We are interested in aiding a human user in managing and performing complex tasks in an office desktop setting. As such, our overall goal is to reduce the amount of effort required by the human to complete the tasks that she wishes to perform. Effort here encompasses both the activities necessary to perform the tasks, and the cognitive load in managing and monitoring them; we use the term *task management* to describe broadly the planning, execution, and oversight of tasks. It follows that a personal assistant agent may aid its user directly by performing tasks on her behalf or in conjunction with her,[36] and indirectly through actions such as providing context for her work, triaging information, and offering suggestions and reminders.[70,16]

We have explored these ideas within a system for intelligent personalized assistance called *Cognitive Assistant that Learns and Organizes* (CALO).[84] The focus for a CALO agent is to support a busy knowledge worker — highly competent professionals whose tasks are complex, adaptable, and collaborative in nature — in dealing with the twin problems of information and task overload.[54] Specifically, CALO's task management capabilities are grounded in a module called the *Project Execution Assistant* (PExA).[60] This article describes the architecture and design decisions of the task management functionality within the implemented PExA module. For the sake of simplicity, in this article we will refer to the system only as CALO.[a]

Prior to the work described, CALO's task-related capabilities were manifest in *delegative behavior*, in which the agent adopts intentions only in response to being explicitly assigned goals by its user. CALO is able to perform a variety of routine office tasks delegated by its user, such as arranging meetings and completing online forms, as well as more open-ended processes such as purchasing equipment or office supplies and arranging conference travel.[60]

An obvious limitation of CALO's delegative model is the lack of a *proactive* capability that would enable the agent to anticipate the needs of its user, as well as to anticipate opportunities and problems, and then act on its own initiative to address them.

[a]Complementing PExA's capabilities for task management in CALO are a meeting assistant[90] and an information assistant.[10] 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 on them, 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.

Indeed, proactive behavior is seen as an essential characteristic of autonomous and semi-autonomous agents.[70] We are interested in developing proactive behaviors along these lines within CALO, to increase the overall effectiveness of the system as a personal assistant for task management.

This article commences with a theoretical model for proactive behavior in the context of a personal assistive agent. Building on prior attempts to distill the desiderata for proactivity, we lay out a set of theoretical principles. We propose an agent reasoning layer to realize these principles in a goal-directed manner. In the latter portion of the article, we describe the implemented proactive assistance capability of the CALO agent system, based on our agent framework.

Contribution. This article describes our approach to operationalizing proactive behavior within an actualized intelligent personal assistant agent. We augment a Belief-Desire-Intention (BDI)[79] framework with a meta-reasoning layer that reasons about *what tasks* may be appropriate for the agent to perform and *when* initiative should be taken to perform them. Such reasoning is inherently *meta level*:[14] in addition to requiring deliberation over a model of the user's state and intentions, it further requires that the agent reason about its capacity to perform potentially helpful tasks given its current commitments and capabilities. Further still, because of the importance of personalization in providing helpful assistance,[82,85] the agent's reasoning and behavior must adapt to the preferences and individual needs of its user.

Guided by a theory of proactivity, our implementation of this reasoning is grounded in *assistance patterns*. These generic rules represent a compiled form of knowledge (at present formulated by knowledge engineers) about how to assist with task management. Since the literature studies the techniques by which a proactive assistant can reason over the modality and timing of its activity — such as predicting the potential disruption versus the potential beneficial effects[81] — we do not focus on this aspect, instead drawing on the techniques available.

Our primary contribution is to the problems of determining both the situations in which to take initiative for task management and what actions to perform. We provide an approach that is theoretically grounded while being practically realizable. Further, the generic framework developed dovetails with mechanisms for improving agent behavior through automated learning. Our work provides the basis for proactive assistance in the implemented CALO system by means of context-sensitive suggestions.

Organization. After situating our work in Section 2, we characterize the properties desired of proactive behavior in Section 3. In Section 4 we present an extended BDI framework with a meta-level layer designed to support proactive goal generation, along with a theory of proactivity to drive this behavior. In Section 5 we describe the operationalization of our model of proactive assistance, as implemented within the CALO agent system. We conclude in Section 6 with a survey of future challenges.

2. Related Work

The vision behind the CALO project was a personal assistant for a busy decision-maker in an office environment, akin to an administrative assistant who can help facilitate routine tasks.[88,60] CALO thus fits into the heritage of user-assistive agents.[5,54,70,8,29] A primary objective of the project was to stimulate the development of Artificial Intelligence (AI) learning technology to support autonomous adaptation to the user.

Isbell and Pierce[45] describe an *Interface-Proactivity continuum* that ranges from zero to full automation. An analogous continuum focused on task management might be: "Do It Yourself", "Tells You What to Pay Attention To", "Makes Suggestions", and "Makes Decisions" (and possibly thus takes action). While systems developed by AI researchers often lean toward the autonomy side of this continuum — in contrast to work in human-computer interaction, which has emphasized the interface side — CALO, like some other previous and contemporary efforts to design assistive agents, spans several points. We next survey a selected subset of these efforts.

The Office Assistant in Microsoft Office was based "in spirit" on the Lumière project.[42] Although more limited in scope than CALO, Lumière shared an ambition to help a user. The domain of the project was user tasks in an office software package; Lumière deliberately considered proactive behavior. In expressing the rationale, Horvitz *et al.*[42] describes the agent reasoning:

> [W]e have also been interested in the question of deliberating about when to step forward to assist a user. We believe strongly that such intrusions should be done in a careful and conservative manner, with the express approval of users.
>
> We built versions of the Lumière prototype that employ Bayesian models to control such "speculative assistance" actions, coupled with user-interface designs that showed promise for minimizing disruption and for putting the user in control of interruptions. . . .
>
> We also designed a "background assistance tracking" feature that would simply watch in the background as a user worked. An analysis, including such compilations as a custom-tailored set of readings, would be made available for review or printing when the user requested such an overall critiquing, or context-sensitive assistance manual.

Lumière provides what we call *application-focused proactivity*: the agent offers assistance in the context of a single application. Other examples of application-focused proactivity include interface agents[54] and, although not often construed as an agent, adaptive user interfaces.[30,28] While this type of proactivity is certainly within the scope of our work, we are interested in broader forms of proactive behavior that extend beyond any single application. Accordingly, we will later introduce two other forms of proactivity.

The Electric Elves project[8] developed personal assistants with a range of functions related to supporting a busy office worker — an application domain similar to that of CALO — including a set of proactive capabilities designed to further specific user objectives related to meeting scheduling. These included, for example, delaying or rescheduling meetings when the user was likely to be late. Electric Elves was focused on shared activities within a team setting, unlike the single-user focus of CALO, albeit in a collaborative work setting.

The RADAR project,[31,29,25] a sibling of CALO, developed a personal assistant intended to help users cope with email overload as effectively as a human assistant. By parsing email messages, the system attempts to identify certain tasks, such as filling out a form or collating information for a briefing report, and offers to initiate them for mixed-initiative user-system completion. RADAR shares with our work an integration into the software desktop and commercial applications, but focuses on email triage, management, information extraction, and email-related task execution.

ANTIPA[71,72] is a cognitive assistant designed to proactively manage information on behalf of overloaded users. The agent integrates probabilistic plan recognition with constraint-based information gathering, with the aim of providing the user with relevant information in a timely manner. Although information gathering is an important way in which an agent can proactively assist its user, it is only one aspect of the task management assistance that we explore.

The value of proactive behavior has been explored in intelligent guides and personal tutors, and in Non-Player Characters in interactive entertainment. In the former, for example, Bellotti *et al.*[1] present an agent that makes suggestions of nearby places of interest, based on a model of user desires and goals. The scope is deliberately limited to more narrow domain tasks, such as guiding the user from the current location to an attraction.

There has been much ongoing work on assistive technologies to aid people with cognitive disabilities in managing their daily activities. Here, proactivity also plays an important role (e.g., Refs. 78 and 38). These systems monitor a person's actions to understand what she is doing, and interact when appropriate to provide reminders, situationally relevant information, and suggestions to aid in problem solving. The complexity of the domain and the challenges of situated or ambient intelligence assistance are substantial, and consequently the types of assistance developed to date are by comparison simple — compared to the types of assistance desirable in the computer desktop domain.

Theories of collaborative problem solving clearly relate to the notion of proactive assistance: user–agent collaborative activity can be viewed as one aspect of task management. For the most part, these theories extend BDI models of agency[79] to incorporate notions of joint beliefs and commitments. For example, Joint Intention theory[12] formalizes the communication acts made between agents to establish and maintain joint belief and intention: the obligations on what 'message' to communicate and under what circumstances to do so. A form of meta-level reasoning over agent state drives the collaborative behavior. Planned Team Activity theory[49] also captures collective intentionality by introducing plural-subject constructs; applications of this formalism focus mostly on team formation.

SharedPlans theory,[36] in contrast, contains only single-subject intentions, but augments them with an *intend that* construct. It specifies collaborative refinement of a partial plan by multiple agents, handling hierarchical action decomposition and partial knowledge of belief and intention. The theory expressly includes meta-cognitive tasks, such as cultivating collaborative goals by deliberating over commitments.

According to SharedPlans theory, members of a collaborative activity have the intention that the activity will be successful, which forms the theoretical foundation for members of a collaboration to support and assist each other. SharedPlans proposes decision-making rules for supportive behavior that suggest reasoning about costs and benefits of a supportive action for the collaboration, though it does not suggest a way of capturing these values.[b]

COLLAGEN[80] is an example of a system that instantiates these ideas, drawing on the SharedPlans theory. It provides a framework for building assistive agents that collaborate with a human to achieve tasks together. Its assistance is based around precoded models of tasks; the current task state is described as a collaborative dialogue consisting of a plan tree (tracking the state within the task model) and a focus stack (tracking the current focus of attention). By reasoning over these structures, COLLAGEN responds to user actions and utterances by acting or communicating appropriately.

Applications of SharedPlans include also task allocation and analysis of helpful behavior,[37] but are limited in terms of proactive scope. Agents' reasoning about themselves and other agents in a collaborative activity and a formalized model for reasoning about the cost and benefit of a supportive action on the collaboration have been developed.[48] Proactive sharing of information, in an extended SharedPlans formulation, has been studied in the multi-agent team setting.[47] SharedPlans has also been applied to discourse modeling.[52]

STEAM[87] is an agent system that a uses hybrid of Joint Intention and SharedPlans theories, with practical extensions for monitoring and replanning as well as decision-theoretic communication selectivity, to reason about team building and communication — including the effect of a supportive action on a collaborative activity. STEAM has been applied to coordinated helicopter operations and virtual soccer-playing agents.[87] This and related lines of work in flexible team coordination and collaborative problem solving agents differs from CALO, where a single agent is assisting a single human user, albeit in a shared work setting.

Applications of Joint Intentions have focused on dialogue management in agent collaboration (e.g., Ref. 46). STAPLE is an agent that uses Joint Intention theory to generate dialogue, as it collaboratively plans and acts to solve problems with other agents.[86] The nature of the collaborative tasks are joint goals shared between a team of cooperative agents; thus STAPLE's operation is not limited to the system – user tasks of COLLAGEN. STAPLE combines an extended BDI architecture and distinguished joint goals. By formal reasoning over its beliefs (albeit a computationally intensive exercise), a STAPLE agent acts and communicates with other agents to achieve the joint task.

Like STAPLE, the Smart Personal Assistant (SPA) is an assistive agent built around a BDI framework.[68] In contrast to COLLAGEN and STAPLE, the dialogue model of a SPA agent is encoded as part of its plans. However, domain-independent dialogue plans (handling discourse-level goals such as recognizing the user's intention) and domain-dependent plans (handling domain-level dialogue aspects and task achievement) are distinguished. Learning is incorporated into the BDI architecture to guide plan selection, so that the SPA

[b]We are indebted to one of the anonymous reviewers for amplification about SharedPlans.

agent adapts its responses according to the context of the conversation, the user's location and interaction device, and the user's interaction preferences.[69]

Collaborative problem solving and proactive assistance are both rooted in the notion of an agent taking action to assist another. Indeed, a collaborative agent will often need to act proactively to fulfill its commitments to its partner. Proactive assistance goes beyond collaborative problem solving, however, in that an agent may take actions unilaterally on behalf of its user without any joint agreement as to the desirability or suitability of the actions.[c] In fact, it is precisely this degree of autonomy that makes proactive assistance potentially valuable, as it enables the agent to support its user without interfering with her normal activities.

Therefore, although a *theory of proactivity* will have much in common with theories of collaboration such as Joint Intentions and SharedPlans, whereas those frameworks focus on high-level characterizations of how and when to provide assistance, an operational theory of proactivity requires further elaboration of these concepts. Theories of collaboration do not define how an agent weighs the cost and benefits of potential goals and the plans to achieve them to assess which are appropriate to perform.[82,26] Moreover, a proactive assistive agent must have an explicit model of user desires, in addition to current user goals and plans, as well as a theory that defines how those can be furthered by actions that the agent can perform. The requirements for and organization of these latter reasoning functions are our topic for this article.

Returning to our starting point, we can regard Isbell and Pierce's Interface-Proactivity continuum as a user-centred perspective on the notion of *adjustable autonomy*.[39] A large body of literature considers the question of initiative in the context of human–agent teams. STEAM is an example of a decision-theoretic approach to decisions such as when and how to communicate. The later Electric Elves project pursued the same questions in the context of assistive agents.[8] Learning from that project, Maheswaran *et al.* survey the key adjustable autonomy questions for personal assistive agents,[55] calling out challenges around capability differentials and personalization, and distinguishing permission and consultation requirements.

Research has shown that humans attribute projections to computers and devices — especially to agents, whether software or embodied.[65] Users' trust in a system and their willingness to ascribe authority for autonomous activity is influenced by system competence[22] and remit (the nature of the autonomy).[24] Awareness of and understanding into (semi-)autonomous behavior is found to be a major determinate of trust, especially for adaptive systems.[35,15] Performance and user interface considerations are significant, even in systems that successfully engender change in user work practices.[50]

When a system takes initiative, that initiative can include communicating with or consulting the user. The potential cost is the interruption of the user and her loss of cognitive focus — particularly if the topic of the interruption differs from what the user was working on.[56,82] Interruption management is usually approached by approximating and weighing

[c]One can construe an implicit agreement between user and agent that the agent act as the user's assistive agent; compare the discussion of the general desire $\hat{\mathbf{d}}$ in Section 4.

the potential benefits of the interruption — to the system's activity, albeit in service of a user: for example the value of information — and the potential costs of the interruption. Such cost-benefit analysis and related decision-theoretic estimation has been explored in multi-agent teamwork settings[81] as well as in assistive interface agent settings.[43,20,71]

3. A Characterization of Helpful Proactive Assistance for Task Management

Ethnographic studies of human work habits and task management (e.g., Refs. 2 and 16) reveal that people usually achieve all their important tasks. We become adept at multi-tasking and remembering what really matters; however, we fail to perfectly achieve tasks with soft deadlines, or to remember less-critical details. When overloaded, we tend to forget the non-essential tasks, and we achieve essential tasks as best we can. It is these areas where assistance technology can thus provide greatest utility.

We take as our starting point that the user has entered a description of her tasks and the tasks assigned to her agent into an electronic to-do list.[2,13] Thus we obviate the need to infer the intent of the user.[42,26,1] We take these to-do entries to be a concrete manifestation of intent descending from some of her goals. Similarly, we assume that the user employs electronic artifacts to keep track of her calendar, and works within an instrumented desktop environment. These are all valid for the desktop context and system infrastructure of a CALO agent.

Our setting is collaborative in nature. A CALO agent serves its user who works within a cooperative teamwork environment with other humans. The user's colleagues each may or may not have a CALO agent assisting him. While recognizing that some users might seek self-advantage in a collaborative setting, the CALO project does not encompass developing gaming-resistant mechanisms; such mechanism design is studied elsewhere.[7,75]

3.1. *Task management with a CALO agent*

We use the term *task management* to refer broadly to the planning, execution, and oversight of tasks associated with work assignments. Three technologies provide direct assistance with task management in the CALO system: a *Task Executor*, a *To-do Manager*, and a *Calendar Manager*. They are shown in Figure 1, together with the modules that operationalize the proactive behavior we describe in the coming sections. Further details on the task management architecture and components for CALO are given by Myers *et al.*[60] and in the references we cite below.[d]

The Task Executor provides the ability to perform tasks on behalf of or in conjunction with the user. Execution is enabled by a library of *workflows*, called the *task library*, that encode processes that can be followed to accomplish a particular task. Workflows are composed of lower-level tasks ("steps"), which are annotated to indicate which agent(s) are permitted to perform them. In addition to permissions, annotations may also indicate

[d]The interested reader may obtain the code of selected modules from the CALO project, which are collectively referred to as the 'CALO Framework', at `pal.sri.com`.

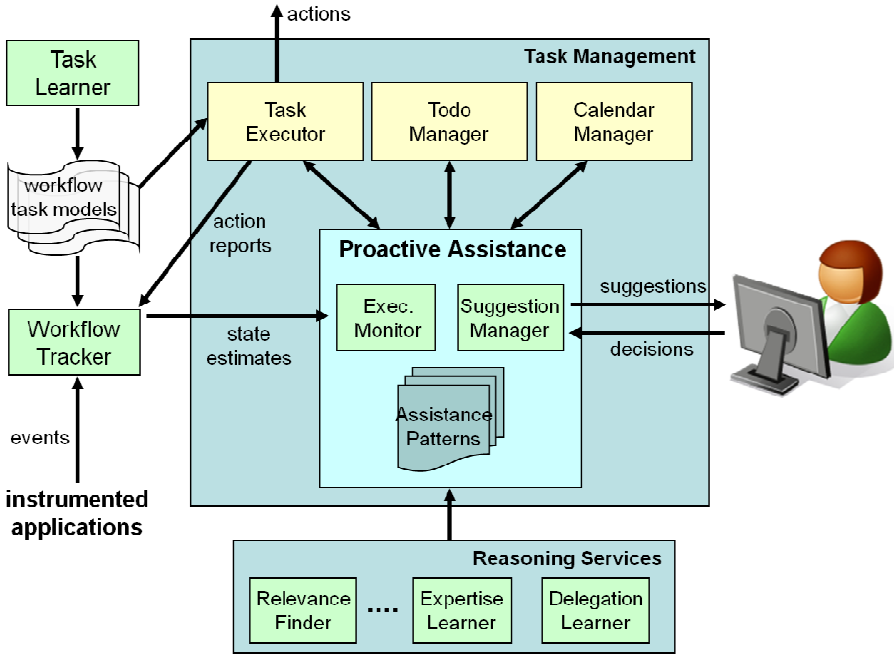

Fig. 1. Task management in the CALO system.

capabilities to perform a step. By default, CALO infers whether it can perform a step by determining whether it has a fully system-executable workflow for that step in its library.

Declaratively, the workflows describe the subtasks that the user, the agent, or other agents achieve to fulfill a goal, together with constraints between the subtasks (e.g., subtask ordering). Procedurally, the workflows provide recipes that CALO can instantiate into a plan to achieve a given (sub)task. Workflows in the task library might be executable only by the user, only by the agent, by either, or by both together.

The Task Executor is built as an "agent" in the SPARK agent system,[59] with the procedural aspects of workflows encoded in the SPARK procedure representation language. Workflows can be specified manually (by knowledge engineers), or be taught to the system (by end users) through a learning by demonstration framework.[33,83,61] The task learning framework also enables end users to edit workflows and to compose new workflows from the library of existing workflows.

Second, the To-do Manager provides a framework to aid the user in organizing and tracking tasks. The CALO To-do Manager, called *Towel*,[13] provides basic functions found in typical to-do managers that help the user remember what needs to be done and when. However, Towel goes beyond such systems through its support for both the delegation of tasks to teammates and the dispatching of tasks for execution by the Task Executor. For task dispatching, the user can assign a task to CALO by selecting from a menu of automated capabilities. Alternatively, the user can define a task informally by providing a textual description of it; Towel will provide the user with a list of possible workflows that

it believes it could perform to help accomplish this task.[34] Learned models derived from implicit user feedback, combined with lexical and syntactic parsing achieve these matchings. For example, given the task description "Plan project review meeting", Towel may suggest a `ScheduleMeeting` workflow as potentially useful. Thus, CALO may be told or can infer a mapping from a subset of the tasks to formal models within a *task ontology* (i.e., the tasks have associated semantic descriptions). It can also infer likely parameters for instances of tasks from the ontology.

Associated with each task in Towel can be a rich body of information about provenance (source, time of creation), requirements (deadlines, priority, expected duration), current state, and relations to other tasks (semantic groupings, task/subtask relations).

Third, the Calendar Manager supports the user in scheduling meetings, in negotiating over meeting requests received via email, calendaring application, or directly via CALO, and in managing temporal commitments.[3,4] Individual calendar entries are defined by their start and end times, the organizer, the participants, category, importance, and location. Like many other CALO components, the Calendar Manager acquires learned models of user preference and behavior. The Calendar Manager uses its acquired model of the user's scheduling preferences to compute preferred candidate schedules in response to a user scheduling request. It presents a subset of the candidate schedules to the user, and updates its preference model according to the user's choice among them.

These three task management tools provide a range of services and information that can inform proactive reasoning as to user and task state. One such capability is an estimation of the 'busyness' of the user, which is determined by combining temporal commitments from the user's calendar with estimated workloads calculated by considering deadlines and durations for tasks in the To-do Manager. A fourth tool, a *Workflow Tracker*, provides additional state information related to user focus and progress on tasks.[53] We describe it in detail in Section 5.

Additional services within the CALO system provide relevant background information to further inform reasoning for proactive assistance. A *Relevance Engine* identifies documents and emails that are potentially related to tasks.[13] An *Expertise Finder* identifies people within an organization who may have expertise on particular topics.[19] A *Delegation Learner* develops models that recommend specific individuals to whom tasks might be assigned, based on prior delegation events.[74]

3.2. *Characterizing proactive behavior*

Within such a setting, three challenges must be addressed in order to develop effective personal assistant agents. Each challenge involves the agent reasoning about its actions, and also meta-reasoning about the reasoning itself. We distinguish tasks performed solely by the user (*user tasks*) from those performed solely by the agent (*agent tasks*), and those performed jointly in partnership (*shared tasks*).

What form of initiative? A personal assistant agent acts when delegated tasks by its user, and when it is obliged to act by commitments it has made (e.g., an agreement with a mer-

chant to purchase an item on its user's behalf[11]). Our hypothesis is that the agent can act under its own initiative at other times, in order to assist its user. A pivotal control issue is answering the question of under what circumstances the agent should consider some proactive action.

What actions to take? We envision a personal assistant agent serving its user in many ways. On some occasions its assistance will be initiated proactively, on other occasions by the user or as a result of actions by another agent. Our focus is formed by the combined questions of determining the situations in which to take initiative for task management and what actions to perform to manifest it. Although we will not address the subject in detail, nonetheless it is important to recognize that the agent should weight whether the actions it is considering will be helpful to the user, by means of a cost–benefit analysis.[e] Moreover, since there will be a measure of uncertainty about the current situation, the user's goals and her focus of attention, and the effects of its actions, the agent should act with care, as we elaborate below. As will be explained, our approach is to have the agent by default offer suggestions that the agent act, rather than make decisions or take action in a fully autonomous fashion.

How to perform the actions? Once the agent has decided to or is compelled to act, it must deliberate further about the modality and timing of its action.[43,27] Is it better to do nothing, to suggest, to confirm then act, or to act without consulting the user? To act now or later? To interrupt or not? Ineffective decisions by the agent in this aspect strongly detract from the perceived utility of a cognitive assistant.[82]

To these three challenges we might add the challenge of *how to learn to do it better*: a helpful assistant should seek to broaden and refine its problem-solving skills through learning.[70,3]

3.3. *Examples of proactive behavior for task management*

Figure 2 provides a selected taxonomy of possible proactive activities that an assistive agent might perform, on behalf of its user, to support task management in an office setting such as that of CALO. We divide the list into four categories: *Act directly*, *Act indirectly*, *Collect information*, and *Remind, notify, ask*. Except those in italic, all the items in Figure 2 have been implemented in the CALO system. The following implemented scenario illustrates how a proactive behavior on the part of an intelligent assistant can provide value in the office domain.

CALO observes the items currently in your electronic to-do list, what you are currently working on, what you have delegated to your CALO and to other people, and your commitments for the week ahead. CALO assesses that your workload is likely to be uncomfortably

[e]The agent might even refrain from acting on a user-delegated task, if it can provide sufficient justification to the user. Of course, the agent's role as an assistant mandates that the user be able to override its decisions, as well as more broadly to advise its operation.

- **Act directly**
  - perform the next step or steps of a shared task
  - perform or prepare for future steps of a shared task now
  - initiate the first step of a shared or agent task
  - suggest (shared) tasks the agent can take over and perform
  - *establish a learning goal (i.e., to learn new capabilities)*
- **Act indirectly**
  - suggest a user task be delegated to a teammate, or that the user offer to take on the task of a teammate
  - suggest a meeting be rescheduled
  - suggest a lower-priority task be postponed to free resources
  - suggest a task be promoted or demoted in priority
  - *suggest (better) ways to achieve a (shared) task*
  - *anticipate failures of (shared) tasks and look for ways to reduce the failure likelihood or the impact of failure*
- **Collect information**
  - gather, summarize information relevant to a user or shared task
  - monitor the status of tasks delegated to a teammate
  - monitor and summarize resource levels and commitments
  - analyze possible consequences/requirements of a (shared) task
- **Remind, notify, ask**
  - remind of upcoming deadlines and events
  - remind of the user's next step in a shared task
  - ask for feedback or guidance from user
  - ask for clarification or elaboration of a (shared) task
  - *monitor and filter incoming messages*

Fig. 2. A taxonomy of some possible proactive activities in task management.

high at the end of the week. Via a message in a peripheral window, CALO offers you a reminder of an important meeting early next week, with the suggestion that a paper review (*on your to-do list*) *could be transferred to a colleague* (*whom CALO identifies as having appropriate expertise and time in his schedule*), *to leave you time to focus on the meeting. In addition, CALO begins to prepare background material for the meeting without being explicitly asked. It attaches the relevant documents to the item in your to-do list and the event in your calendar.*

This scenario illustrates two distinct types of proactive behavior for an agent. The first type, which we call *task-focused proactivity*, involves providing assistance for a task that the user either is already performing or is committed to performing; assistance takes the form of adopting or enabling some associated subtasks. Task-focused proactivity is exemplified in the above scenario by CALO collecting background information in support of

a scheduled meeting. We note that systems such as COLLAGEN and SPA are designed for task-focused operation, although not task-focused proactivity.

The second type of proactive behavior, which we call *utility-focused proactivity*, involves assistance related to helping the user generally with her set of tasks, rather than contributing directly to a specific current task. An example of this type occurs in the scenario when CALO takes the initiative to recommend transferring a paper review task in response to the detection of high workload levels. This action is triggered not by a motivation to perform (either partially or fully) an individual task on the user's to-do list, but rather in response to a higher-level motivation.

The vision for effective task management by CALO includes proactive behavior as a complement to the previous delegative-only behavior. In this vision, the agent more adeptly supports the user in task management. To do so, the agent can act according to its own initiative at times — the first of the three challenges described earlier – -subject to the adjustable autonomy control of its user. The CALO vision does not encompass the assistive agent performing fully autonomous task management: the servant does not become the boss.

3.4. *Principles for proactive behavior*

To guide the development of proactive agent behavior, we set out nine principles, akin to the principles for intelligent mixed-initiative user interfaces[41]:

- **valuable**: advances the user's interests and tasks, in the user's opinion;
- **pertinent**: attentive to the current situation;
- **competent**: within the scope of the agent's abilities and knowledge; [f]
- **unobtrusive**: not interfering with the user's own activities or attention, without warrant;
- **transparent**: understandable to the user;
- **controllable**: exposed to the scrutiny and according to the mandate of the user;
- **deferent**: gracefully unimposing;
- **anticipatory**: aware of current and future needs and opportunities; and
- **safe**: minimizes negative consequences, in the user's opinion.

These principles reflect the centrality of the user and her experience. The agent's actions are valuable only if they ultimately add value for the user. They assist only if they are performed in a manner that takes account of the user's focus and immediate as well as longer-term needs. Horvitz *et al.* for instance capture the behavior sought in the Lumière project: "The sensibility of an intuitive, courteous butler . . . potentially valuable suggestions from time to time . . . genuine value . . . minimal disturbance."[42] (See also the vision exposed by Negroponte.[67])

Prior research on assistive agents emphasizes ease of understanding by the user of the agent's operation, together with ease of directing, ignoring, and correcting the agent, as

[f] Iba[44] points out that competence by itself is not sufficient for helpfulness and, further, argues that limited incompetence need not preclude helpfulness.

well as working entirely without it.[8,38,41,35] Transparency and controllability are essential to build trust, which is especially important in an agent with an extended life cycle, such as a user's assistant,[70,35] and even more so if the agent acts on its own initiative.

Returning to the earlier example, CALO's actions are pertinent to the important upcoming meeting. CALO itself is not capable of reviewing the paper; identifying a colleague who potentially is able, CALO does not delegate the task from your to-do list automatically, but leaves you in control to take the suggestion or not. This suggestion and the preparation of background materials are both safe, defined in this case by an absence of changes of state other than a gain in information. Throughout, CALO's actions are unobtrusive: the communication is via a peripheral message with context, and the completed information gathering is again in context, attached to the relevant artifacts in your working environment.

3.5. *A second example scenario*[g]

You have put "hire engineer" on your to-do list. CALO suggests this could be an instance of a Hiring *shared task. When you confirm, CALO readies the Hiring Approval form, prepopulating what it can. It attaches the form to the task as a link. Later, after you complete the remainder of form, CALO asks whether it should submit it for you. You answer negatively, because you want to review the form tomorrow before sending it.*[h] *Accordingly, CALO adds a "Submit Hiring Approval form" to its suggestion list. The next day, when satisfied with the form, you accept this suggestion.*

The next step in the Hiring task must wait until you indicate that approval is received; CALO can prompt you, should it detect, e.g., an email notification. CALO prepopulates the Job Description and Advertise Job Listing forms. Based on its learned models of expertise, CALO suggests that the subtask of Write Role Summary for the former could be delegated to a member of your team.

In due course, as applications are received via the company website, CALO searches internet databases for each applicant, attaching any publications and patents it finds to the database entry for that applicant on the company intranet. Once you have begun to select candidates to bring for interview, CALO reasons over the expertise and availability of interviewers. CALO knows that preparations for a client meeting in three weeks' time are expected to occupy a significant amount of staff time. It suggests the options of inviting fewer candidates, rescheduling some less important meetings, or delaying your stated deadline for the hiring decision.

Later CALO supports you through scheduling the interviews and the decision meeting. It prepares the form rejection and offer letters according to templates, for your review, and with your approval offers to forward them to central administration for mailing.

This scenario shows examples of proactive behavior in each of the four categories of Figure 2. Observe how the cognitive assistant must support its user through the life of ex-

[g]This scenario intentionally extends slightly beyond the implemented CALO functionality (Section 5, also Refs. 60, 74 and 61) in order to communicate the vision for personalized task management assistance.

[h]Although CALO does have facilities for natural language understanding,[90] we need not envision use of such a modality here.

tended processes, as opposed to merely assisting with one-shot tasks. CALO acts directly, such as initiating the shared task of completing the Hiring Approval form. CALO acts indirectly, such as anticipating difficulty (leading to greater risk of failure) of the preparations for the client meeting. CALO proactively collects information, such as the patents of the interview candidates. Finally, CALO would remind, notify, and ask, such as regarding the submission of the Hiring Approval form. Throughout, CALO's actions are transparent, controllable, deferent, and safe.

4. A BDI-Based Framework for Proactive Assistance

Having characterized helpful proactive assistive behavior, we now introduce — more concretely — an extended BDI model of agency designed to support such behavior. Specifically, we define a meta-level layer that augments a BDI framework to enable (1) identification of potentially helpful actions, and (2) determination of when those actions should be performed. The following Section 5 will describe our implementation of the framework.

The well-known BDI model provides an explicit, declarative representation of three key mental structures of an agent: informational attitudes about the world (Beliefs), motivational attitudes on what to do (Desires), and deliberative commitments to act (Intentions).[79] The primary deliberative processes of a BDI agent can be broadly characterized as focusing on *goal selection* (i.e., identifying what intentions to pursue) and *action selection* (i.e., how to pursue them). This reasoning necessarily takes into account the current BDI cognitive state of the agent to determine what is feasible and desirable given current beliefs and commitments. BDI agent frameworks such as Jadex,[77] JACK,[92] PRS,[32] and SPARK[59] implement these decision-making processes as a combination of base- and meta-level reasoning:[i] simple strategies are hard-coded in the base level of the agent but more sophisticated agent-specific meta-level strategies can be invoked as needed. In particular, meta-level reasoning can support control of deliberation, conflict resolution, identification of learning goals, and — as described below — proactive behavior by an intelligent assistant.

The reasoning that enables CALO's current (non-proactive) task-related capabilities draws on an extension of the classical BDI framework, called the *delegative BDI model*.[64] In particular for our purpose, this extended framework distinguishes different types of goals: Candidate Goals (goals that may make the combined set of goals inconsistent if they are adopted) and Adopted Goals. The latter set has key properties (consistency, feasibility[6,18,17]) that are simply assumed of goals in many implemented BDI systems.[j] The delegative BDI model also incorporates forms of user-specified guidance and preferences on the execution of these tasks, and on the agent's cognition, called *advice*. However, the

[i]Meta-level refers to reasoning about the agent's reasoning itself.[14] The subjects of such meta-reasoning are decisions such as whether to enter reasoning, and which reasoning to undertake.

[j]As several authors have pointed out, many implemented BDI agents assume that the desires of the agent are consistent and feasible, and effectively treat goals and desires as equivalent; thus, these systems might better be called B(G)I implementations.[77,91]

framework in the various aspects of its meta-reasoning does not encompass deliberation in order to manifest proactive assistance by generating Candidate Goals. We next describe an architectural framework to enable such deliberation.

4.1. *Architecting for proactive assistance*

Figure 3 depicts proactive goal generation in an extension of the delegative BDI agent architecture. As usual in BDI-like formulations, the agent's base-level cognition reasons about how to realize Adopted Goals as intentions. Multiple forms of meta-cognition are depicted to the right. In addition to the usual BDI meta-cognition over aspects such as agent control — for example, over goal selection — we show *proactive goal generation* and *filtering*, an extension to the prior delegative BDI model. Other forms of meta-cognition include the development of learning goals, for example.

The framework includes, at a conceptual level, recognition of the user as a cognitive entity. However, it is not within our remit to explore models of user cognition. Computationally grounded models of user emotive state, for example, have been developed in service of the development of affective agents.[76]

A personal assistive agent can be thought of as holding an overarching meta-desire of being a helpful assistant to its user, which we denote $\hat{\mathbf{d}}$. We can envision a limited number of additional desires at both the base- and meta-levels. One desire might be to learn (although one could construe this as the agent bettering itself in order to become a better assistant). Indeed, in principle, the majority of an assistive agent's desires — or at least goals that might arise from them — can be considered as consequences of the overarching high-level desire $\hat{\mathbf{d}}$. In practice, an explicit representation of motivational attitudes will be chosen, to avoid the complexity of excessive first-principles reasoning during execution.

Adopted Goals are, for our purposes, a subset of Candidate Goals. Candidate Goals (CGs) are created through two mechanisms. At the base level, they arise from the agent's

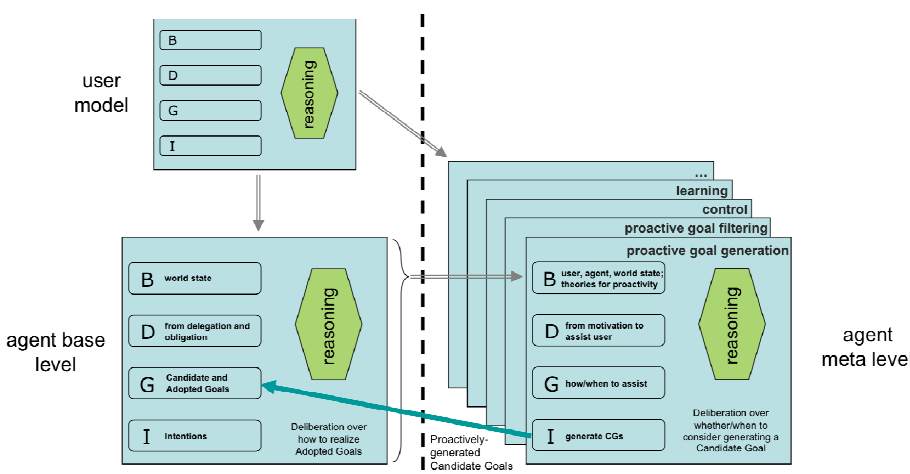

Fig. 3. Extended BDI agent architecture for proactive assistance.

motivations to achieve tasks delegated by the user. At the meta-level, CGs are generated proactively as depicted, as a result of deliberation over a *theory of proactivity*, elements of which are described in Section 4.2. This aspect of meta-cognition, motivated by the high-level desire $\hat{\mathbf{d}}$, reasons over agent beliefs about user, agent, and world states, user and agent capabilities, as well as a theory of helpfulness.

As we have noted, the agent's generation of a CG does not imply that it will necessarily adopt the CG. The control aspect of meta-level cognition chooses how to execute any adopted goal; in particular, the agent might wait before acting, suggest that it acts, ask whether it should act, just act, and so forth.[42,81,27] This filtering of proactively-generated CGs is accomplished by the meta-layer denoted *proactive goal filtering* in Figure 3.

The architecture does not impose the characteristics of the agent's behavior, which will vary from agent to agent. Similarly, it does not specify the mechanism for reasoning to determine which CGs to create for transferal to the base-level portion of the agent. This strategy can be freely specified according to the character of the agent by appropriate instantiation of the theory of proactivity. Section 5 describes our implemented approach to CG generation, adoption, and assistance manifestation, founded on the elements we next describe.

4.2. *Elements of a theory of proactivity*

Section 3 identified three challenges for an agent to support proactive assistance: what form of assistance, what actions to take, and when and how to perform the actions. Recall that our focus is the first two challenges in the context of task management. We now propose principles for a *theory of proactivity* designed to support a proactive agent in meeting these challenges. This theory is composed of subtheories for *user desires*, *helpfulness*, and *safe actions*. With our focus on the operationalization of proactive assistance, we do not develop formally the subtheories.

Theory of User Desires. A theory of user desires is necessary to describe the long- and short-term objectives of the user. Such a theory provides the means to assess the situated value of each potential agent action in terms of the user's objectives. The question for the agent is then: when are actions of varying degrees of safety, utility, and timeliness to be considered? If a task has many safe actions and high perceived benefit, should it be barred because one action is potentially unsafe, e.g., accepting a meeting on the user's behalf?

While application- and task-focused proactivity seek to provide assistance to the user with immediate, tangible goals, utility-focused proactivity by contrast addresses more general objectives of the user. Utility-focused proactivity requires a representation of the user's unstated *interest goals* (in the terminology of the cognitive model of Ortony, Clore and Collins (OCC)[73]) as well as explicitly stated *achieve* and *replenishment* goals.[k] Since interest goals differ between individuals, a helpful assistive agent requires such a model

[k]We might say that OCC interest goals correspond to desires in the BDGI model; OCC achieve goals map directly to goals of achievement, while OCC replenishment goals can be seen as a form of maintenance goal.[91]

(perhaps learned) of its user, in order to assess the value of agent actions. Kamali *et al.*[47] and Oh *et al.*[72] are among those who recognize the value of a model of user desires in providing the most helpful assistance over an extended period.

Further, meta-reasoning over the sufficiency of the model (i.e., how complete, current, and accurate the agent believes is its knowledge of the user's desires) guides the agent's goal deliberation and also guides its consideration of learning goals.

Theory of Helpfulness. A theory of helpfulness defines the principles that direct the agent's reasoning to determine what actions would (most) aid the user now and in the future. In particular, this theory encodes the logic for selecting among possible intentions and the means to pursue them, given a characterization of current user desires. The various approaches in the literature to operationalize a theory of helpfulness include Bayesian modelling,[42] decision theory,[26,72] and forms of logical rules.[27]

Theory of Safe Actions. A theory of *safe actions* is necessary to define bounds on what an agent is allowed to do when performing tasks proactively. In particular, given that the agent will be acting on its own initiative without user awareness of its actions, it is important that only actions with benign or positive effects be performed, so as not to interfere with user activities or change the world in unexpected ways.

Anticipating the consequence of actions requires suitable models of effects, temporal projection (for instance, with Linear Temporal Logic[40]), and possibly dedicated data structures and reasoning (e.g., Refs. 89 and 58). The definition of a safe action varies from context to context, as a conjunction of factors such as: maintains world state; maintains world state except to increase knowledge; immediately reversible without cost; immediately reversible with negligible cost; reversible without cost; reversible with negligible cost; reversible; without negative consequence on user tasks; without negative consequence on tasks of others in the team; and with limited use of shared (team) resources.

5. Operationalizing Proactivity through Assistance Patterns

As Grosz and Kraus[36] remark, BDI and collaboration theories are less often directly implemented in practical agent designs as much as used to provide a "system specification", or even simply as a source of insight informing the design. For example, COLLAGEN's discourse reasoning algorithms originate from reasoning with the SharedPlans *intend that* construct, but do not implement explicit reasoning over such constructs.

Similarly, our implementation of the meta-level components for *proactive goal generation* and *proactive goal filtering* in Figure 3 avoids reasoning over explicit theories of user desires, helpful activities, and safe actions. Rather, for proactive goal generation, these theories are compiled into a form of meta-knowledge that we term *assistance patterns* (APs). Assistance patterns provide a form of knowledge that bridges the gap from the high-level desire $\hat{\mathbf{d}}$ to help the user to more concrete motivations for the agent. At present APs are compiled by knowledge engineers; it is not within our scope to explore automated AP compilation from high-level specifications of proactive behavior. APs are defined in terms of a set of *trigger conditions* that, when satisfied, identify one or more Candidate Goals

to be considered for passing to the base-level component of the agent. AP triggers are defined in terms of beliefs about the user's mental state (e.g., goals, commitments, focus of attention), the agent's own state, and the world state.

Part of the user's mental state consists of her preferences about interaction and assistance. These include, for instance, her preferred types of assistance, her preferences about interruption (including timing), and her preferences about modality of interaction.[82] CALO combines reasoning over current user, task, and system activity and the context of that activity, and reasoning over the user's preferences, in order to appropriately trigger APs and act on triggered APs, as we will explain. CALO builds a preference model from two sources. First, it obtains explicitly stated preferences, such as through a "wizard" interface for new CALO users. Second, CALO observes the user's activities and responses, and refines and augments its preference model through automatic learning from explicit and implicit feedback (e.g., Ref. 60).

In the remainder of this section we describe our operationalization of proactive assistance. The framework presented in Section 4 is implemented in the CALO agent system, through the Proactive Assistance modules shown in Figure 1. We describe behavior specification through assistance patterns, state estimation through the Workflow Tracker, and Candidate Goal deliberation by the Suggestion Manager.

5.1. *Assistance Patterns describe potentially helpful proactive behaviors*

The assistance patterns of an agent encode the manifestations of proactive behavior given in Figure 2. Four example APs are shown in Figure 4 as pseudocode, with `cue` denoting the triggering conditions. The first AP, **`reduceUserBusynessByDelegatingTask`**, cues from an estimation that the user's busyness is above a threshold; the AP generates a Candidate Goal to suggest that a task be delegated to another agent, as we saw in the earlier example scenario. Second, AP **`commenceNextStepOfSharedWorkflow`** cues from the trigger that there exists a shared task for which the agent can perform the next step; the CG that results is that the agent suggest it perform this step.

The third example is the AP **`remindOthersTheirOverdueTasks`** that identifies tasks assigned to other agents that are behind schedule; the AP generates a CG for CALO to issue appropriate reminders. As a final example, AP **`identifyWeakSynergyBetweenTasks`** cues from the existence of possible synergy between two tasks (contrast Ref. 89); the CG that results is that the agent propose notifying the user accordingly.

We implemented a set of APs within the operational CALO agent, using the BDI-based SPARK system[59] augmented by an explicit representation of Candidate and Adopted Goals. All the activities of Figure 2 are implemented, except those denoted in italic. The pseudocode of APs such as those in Figure 4 is readily translatable into SPARK's procedural representation, leveraging its meta-level events and meta-reasoning capabilities, and its concept of *advice*. Advice is form of guidance and preferences that can be specified by users to exert run-time control on what an agent does (i.e., degrees of autonomy) and how it does it (i.e., selection among alternatives).[62,63]

```
reduceUserBusynessByDelegatingTask
cue:  user busyness > threshold AND
      there exists a user task t AND
      CALO knows that t can be potentially delegated
body: select agent A from set of potential delegatees
      create CG to suggest that user delegates t to A

commenceNextStepOfSharedWorkflow
cue:  there exists shared workflow w AND
      w has a subtask t' not yet commenced AND
      t' is immediate successor of a completed subtask AND
      t' is feasible for CALO to perform AND
      no advice prohibits CALO from commencing t'
body: create CG to suggest that CALO commence t'

remindOthersTheirPendingTasks
cue:  there exists workflow w for task t AND
      expected remaining duration of w will exceed deadline for t AND
         t has a subtask t' being executed by agent A
body: alert user that w could exceed deadline
      create CG to suggest that CALO remind A of t'

identifyWeakSynergyBetweenTasks
cue:  there exist tasks t and t' in different workflows AND
      the post-conditions of t imply the post-conditions of t'
body: create CG to alert user that accomplishing t will accomplish t'
```

Fig. 4. Sample assistance patterns.

We emphasize that many APs are *generic*: they are general capabilities that apply in any circumstance within the domain of the agent. For example, the AP to perform the next step of a shared task may be expected to be relevant as a source of CG generation for all CALO users in all circumstances.[1] While the focus of our implementation has been a set of universally applicable patterns for task management, we envision certain patterns that are specialized to a given context. For example, one user might teach her CALO an assistance pattern that the agent should send her a text message, if there is no response from her secretary over a certain period.

5.2. *Workflow Tracker informs Assistance Patterns*

To assist its user with task management, a personal agent requires an understanding of the user's goals, and knowledge of means by which the user and/or the agent can achieve these goals. As described in Section 3.1, part of the context of task management in the CALO system is the user's list of her tasks in the To-do Manager. Recall that some of these tasks are associated with formal models, i.e., system-understandable descriptions of the user's goal, and that CALO's task library consists of a dual declarative/procedural representation of multi-agent workflows to achieve a given goal.

When a workflow involves user steps, keeping track of progress is challenging. For example, in the workflow `JournalPaperReview`, the user first downloads the paper and the review form attached to the review request email. Next, the user prints the paper

[1]This is not to say that such CGs will always be generated from this AP: it depends on whether the AP's cue is triggered, as well as on the outcome of agent's meta-decision making processes.

and completes the review form. Finally, the user sends the completed review form back as a reply to the request email.[m] It would be burdensome for CALO to require the user to explicitly indicate commencement and completion of every step she undertakes. We call the problem of automatically identifying the workflow and the user's current step *workflow recognition and tracking*. As shown in Figure 1, CALO includes instrumentation for the desktop (Windows Explorer) and common applications such as email clients (Thunderbird), web browsers (Firefox and Internet Explorer), and office applications (Word, PowerPoint, Excel) so that user-performed actions are captured and logged. The *Workflow Tracker* module,[53] identifies whether the stream of captured interaction events matches with any of the workflows in the task library and, if so, tracks its current progress.

Variants of the *Hidden Markov Model* (HMM) have been used for this kind of activity tracking problem. However, in the desktop domain, steps in a workflow are often associated with a particular desktop object, such as an email, file, or webpage, best described as a parameter for the step (e.g., `OpenFile("review.doc")`). To accommodate workflow parameters, CALO uses a *Logical Hierarchical HMM*[66] as its representation of the workflow model.

The Logical HMM extends the HMM state to take the form of a ground atom in a first-order language. State transitions can be written in the form of logical rules, such as $\mathrm{OpenFile}(X) \rightarrow \mathrm{EditDocument}(X) : 0.8$. Here, the variable X of the predicates ranges over the set of documents in the system, and 0.8 represents the transition probability. Similarly, the observation model is $\mathrm{OpenFile}(X) \rightarrow \mathrm{WordOpenEvent}(X) : 1.0$. In order to accommodate irrelevant activities between workflow steps (e.g., the user reads some other emails), a distinguished 'Background' state is included in the model; it can generate any observable event uniformly. The Logical Hierarchical HMM further arranges the HMM states into a hierarchy, in order to successfully scale the workflow tracking.

Workflow recognition can now be viewed formally as a filtering problem on the Logical Hierarchical HMM representing the workflow. We adopt a particle filter approach to avoid the prohibitive cost of exact inference. Given a stream of user interaction events, the algorithm returns a distribution over the possible steps in the workflow (including a 'Background' state). This allows CALO both to identify the most likely step and to identify the most likely parameter values for this step. The algorithm is described further by Duong *et al.*[21]; it has been extended to handle multiple interleaved concurrent workflows.

The state information from the *Workflow Tracker* is provided to the Execution Monitor (Figure 1) and exploited by the assistance patterns to generate situationally relevant Candidate Goals, both task focused (i.e., relating to the task that Workflow Tracker identifies the user is working on) and utility focused (e.g., relating to overall workload).

5.3. *Suggestion Manager filters Candidate Goals generated by APs*

The principle of goal-directed focus in deliberation[6,12] applies to assistance patterns in terms of contextual filtering, as much as it does to other aspects of agent deliberation,

[m] A personal assistant agent could perform some of the steps, such as sending the reply email. Other steps, such as completing the review, are beyond current technology!

such as the agent's regular control loop (compare Figure 3). Accordingly, generation of a proactive CG need not entail its adoption. Consideration of APs and the CGs they generate is informed by the context of the agent's current mental state and its beliefs about the user's state — including its estimate of the user's current task — and about the world. At any point, the agent may deliberate over whether to adopt any CGs the APs may generate. At the other extreme, the agent designer may choose to encode some APs so that their body executes without explicit deliberation nor consultation with the user, as soon as the cues become true.

To support this kind of meta-reasoning, which instantiates the theory of helpfulness, and further to provide contextually sensitive assistance, we implemented a *Suggestion Manager*, as shown in Figure 1. The Suggestion Manager provides CALO with the *proactive goal filtering* meta-layer depicted in Figure 3. This module again leverages meta-level reasoning facilities. It deliberates over whether and how to proactively act, complementing the existing situations where CALO is obliged to act, such as when a subtask is explicitly delegated by the user.

All of the APs currently implemented in CALO result in a CG to create a suggestion to the user; none lead to autonomous action except by the decision of the Suggestion Manager. Thus, we defer towards the interface end of the Interface–Proactivity continuum. This design decision of limited autonomy follows the principles highlighted earlier, pursuing the objectives of transparency, controllability, and safety, so that non-technical users may trust the CALO agent in real-world setting.

The Suggestion Manager reasons over the proactively generated CGs to decide whether to manifest each suggestion (or simply drop it), and if so, when and how. Figure 5 shows a suggestion that originates from the AP **`commenceNextStepOfSharedWorkflow`**, unobtrusively manifest in a peripheral sidebar (top right). If the user accepts a suggestion, then CALO acts on the body of the suggestion. For example, it commences the next step of the workflow: here, to open the attachment. In this way we design CALO's default behavior to be safe and deferential, minimizing the cost to the user of unwanted or inappropriate proactive activity. The Suggestion Manager may, however, decide not to display the suggestion, but simply go ahead and act on its body without explicit instruction from the user. By default, it acts autonomously only when given advice it may do so, such as, "always accept tasks delegated to me by my manager".[62]

We have implemented other interaction modalities besides the peripheral sidebar display depicted in Figure 5. In order of increasing demand of attention (and so disruptive cost if inappropriate), these include 'toast' toolbar notifications, chat messages, and non-modal and modal dialog boxes, in addition to email. CALO emphasises peripheral and contextual modalities so that it rarely interrupts the user in a manner demanding of attention.

The Suggestion Manager's reasoning accounts for the user's interaction preferences, her current activity (to avoid acting or interrupting out of context[51]), the potential consequences of its actions (cost, reversibility), the certainty of its information (e.g., confidence of workflow state), and adjustable autonomy permissions. It performs a cost–benefit computation with adaptive weights and, currently, fixed rules. CALO therefore acts, asks, suggests, or does nothing, in order to assist and (it is hoped) not irritate the user.[41,27]

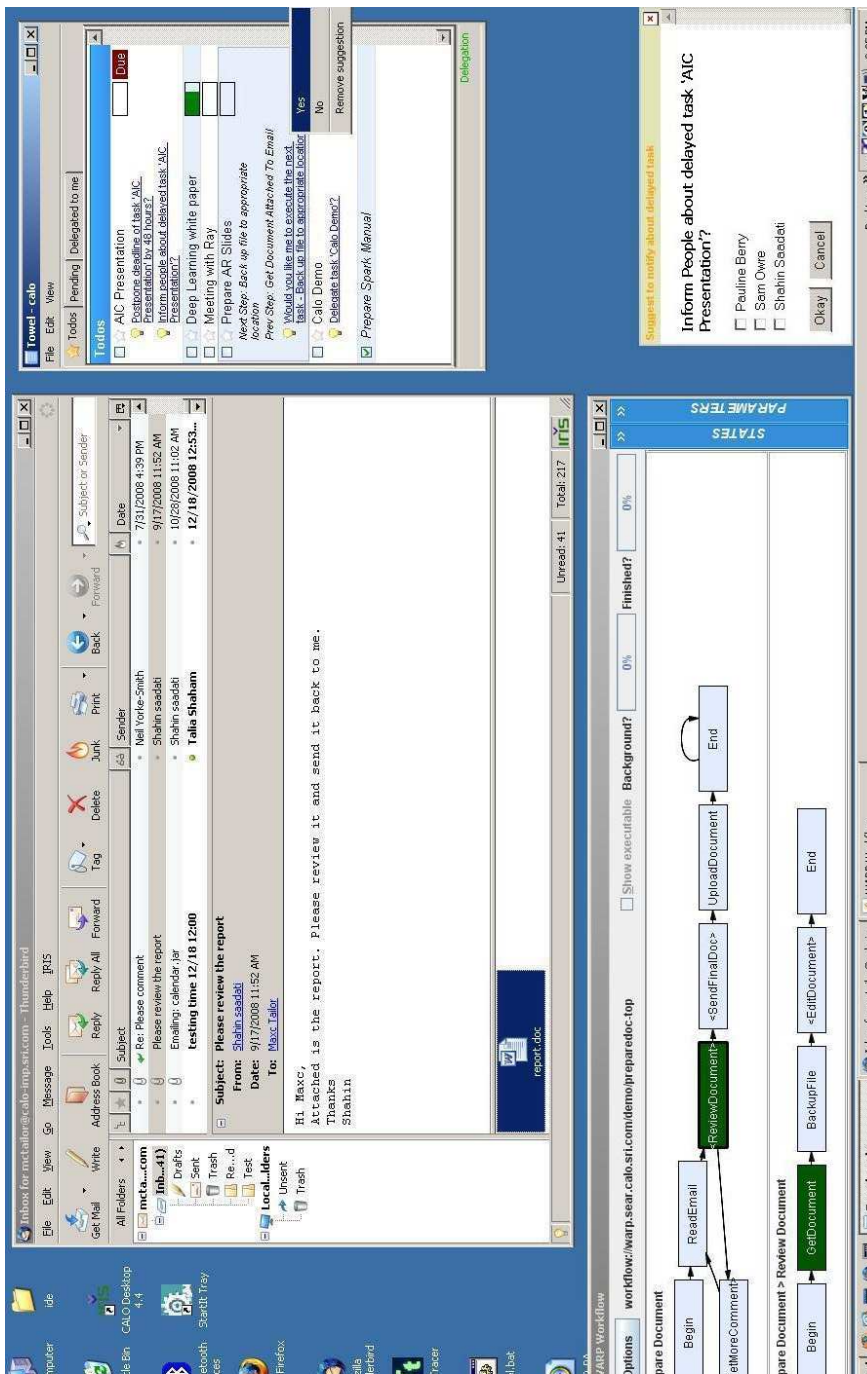

Fig. 5. Task-focused proactive assistance provided by CALO.

The cost–benefit features include estimates of the value of suggestion type, value of suggestion instance, cost of mistake, and cost of interruption; the urgency (time-sensitivity) of the suggestion; and the degree of uncertainty and the system's confidence. Each AP specifies the *type* of each suggestion that it generates. Preconfigured thresholds govern the number of suggestions of each type that can be displayed concurrently, the maximum frequency for suggestions of each type and about any subject, and the permitted or prohibited modalities of each suggestion type. The thresholds for acting, asking, suggesting, or doing nothing are established from a range of default values according to user-stated advice, and elicited initial preferences from the configuration 'wizard'. In the current system the thresholds are static.[54,27] Future work is to enhance the reasoning, drawing on more sophisticated models pioneered in other systems (e.g., Ref. 42).

By adapting to its user's preferred working and communication styles, and her capabilities and experience (e.g., Refs. 82 and 23), an agent becomes a more helpful and thus valuable assistant over time. In pursuit of this objective, the Suggestion Manager observes and learns from user feedback about its suggestions. The feedback is optional and may be explicit or implicit. Explicitly, in some views of the user interface the user may click on 'thumbs up/thumbs down' icons, or deliberately dismiss a suggestion. Figure 5 shows the latter form of feedback via a context menu. (Suggestions that are not accepted but never explicitly dismissed are, in due course, removed automatically, according to their relevance, temporal expiry, priority, and the available space in the relevant modality.) Implicit feedback comes in the form of suggestions that the user accepts or ignores. Based on the observed feedback, the Suggestion Manager adapts the weights used in its deliberation; the adaptation follows a damped exponential scheme.[n] Future work is to explore transfer learning of suggestion importance, timing, and modality between classes of users, e.g., according to the role or the level of expertise of the user.[74]

6. Conclusion and Future Work

Proactive behavior promises to increase the value of an assistive agent by enabling it to take on a larger role in helping a user complete tasks and manage complex information spaces. Proactive behavior encompasses more than acting directly to achieve an assigned task. We have characterized the properties desired of such behavior, and presented an extended agent architectural model that features a meta-level layer charged with identifying potentially helpful actions and determining when it is appropriate to perform them. The meta-reasoning that answers these questions draws on a theory of proactivity that describes user desires, a model of helpfulness, and conditions under which it is safe to perform actions. *Assistance patterns* represent an engineered form of this knowledge that instantiates this meta-reasoning over the agent's beliefs about the user's mental state and actions as well as over world state. We have implemented the resulting generic framework for proactive goal generation as part of the CALO personalized assistant agent. The implemented frame-

[n]Consider a weight of w normalized into $[0, 1]$. A positive feedback instance increases the value to $w + \frac{1}{2}(1-w)$ while a negative feedback instance decreases it to $\frac{1}{2}w$. Explicit feedback is ascribed higher significance than implicit.

work draws on a broad range of services in the CALO system, including sophisticated workflow recognition that informs the agent's beliefs about the user's state and action.

This article has described the rationale, basis, and approach in designing CALO's proactive capabilities for task management assistance. Aspects of the system have been specialized to several transition domains.[53,83,74,61] Notably, the Towel to-do list manager evolved into *Task Assistant*, in which to-do management is combined with collaborative multi-user task execution and proactive suggestions. These suggestions include parameter completion, delegation suggestions, task matching into an ontology, and system task execution. Deployment of Task Assistant to multiple organizations within the US government is underway.

Helpful proactive assistance beholds significant technical challenges. Besides the theoretical underpinnings for a principled approach, agent behavior must most importantly be within context.[70,51,69,28] The vision of agents that operate like intuitive and courteous butlers hinges on an understanding of the user and the world — a combination of cognitive modelling and recognition of task activity — and of how and when to assist, lest the agent be proactive but anything but courteous. Our implementation already draws on activity recognition technology. Our ongoing work is to strengthen the cost–benefit, timing, and modality reasoning of the CALO Suggestion Manager, and to adapt the agent's behavior more subtly to explicit and implicit user feedback.

Directions for the future include developing a logical formalism in order to make guarantees about agent behavior by reasoning over APs within an instantiation of the theory of helpfulness; a more sophisticated user model including estimation of the user's emotive state; and the potential of building CALO towards an affective agent.[76] Another direction, although not within our plans for CALO, is user–agent and team collaborations that involves complex and interleaved actions and subplans.[26,48] Here, reasoning about uncertainty must take account that some actions (e.g., subactions of a collaborative activity and supportive actions) may fail unexpectedly, and that beliefs between the agents may be divergent.

The principles of proactivity presented in Section 3.4 stem from the vision of an assistive agent acting as a "courteous butler". In other settings, an assistive agent may be endowed with a greater level of autonomy.[24] Here, key heightened challenges are maintaining trust and a greater sensitivity to managing the risk–utility tradeoff of acting.[26]

The lessons learned from our ongoing work — and that of other efforts to build personal cognitive assistants — emphasize the interdisciplinary task if helpful personalized assistant agents are to become a reality in day-to-day life.[38,57] These include intention recognition (as a broader challenge besides activity recognition), reasoning about actions (such as the theories we have described) and uncertainty, planning and scheduling, social network analysis, and knowledge capture, together informed and in collaboration with cognitive science and human-computer interaction.

Acknowledgments

The authors thank the anonymous reviewers for their constructive suggestions, and the reviewers of an earlier version of this article that was presented at AAMAS'09. We ex-

press gratitude to H. Bui and the CALO Activity Recognition team, to J. Carpenter, A. Muruganujan, and A. Rodriguez for additional engineering development, to A. Spaulding, and to the CALO testing and support team. This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-07-D-0185/0004. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or the Air Force Research Laboratory.

References

1. V. Bellotti, B. Begole, E. H. Chi, N. Ducheneaut, J. Fang, E. Isaacs, T. King, M. W. Newman, K. Partridge, B. Price, P. Rasmussen, M. Roberts, D. J. Schiano, and A. Walendowski. Activity-based serendipitous recommendations with the Magitti mobile leisure guide. In *Proc. of CHI'08*, pages 1157–1166, Florence, Italy, 2008.
2. V. Bellotti, B. Dalal, N. Good, P. Flynn, D. G. Bobrow, and N. Ducheneaut. What a To-Do: Studies of task management towards the design of a personal task list manager. In *Proc. of CHI'04*, pages 735–742, Vienna, Austria, 2004.
3. P. Berry, K. Conley, M. Gervasio, B. Peintner, T. Uribe, and N. Yorke-Smith. Deploying a personalized time management agent. In *Proc. of AAMAS'06* (*Industrial Track*), pages 1564–1571, Hakodate, Japan, 2006.
4. P. Berry, M. Gervasio, B. Peintner, and N. Yorke-Smith. PTIME: Personalized assistance for calendaring. *ACM Transactions on Intelligent Systems and Technology*, 2(4):40:1–40:22, 2011.
5. G. A. Boy. *Intelligent Assistant Systems*. Academic Press, San Diego, CA, 1991.
6. M. E. Bratman, D. J. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 14:349–355, 1988.
7. C. Castelfranchi and Y.-H. Tan. The role of trust and deception in virtual societies. *Intl. J. Electronic Commerce*, 6:55–70, 2002.
8. H. Chalupsky, Y. Gil, C. Knoblock, K. Lerman, J. Oh, D. Pynadath, T. Russ, and M. Tambe. Electric Elves. *AI Magazine*, 23(2):11–24, 2002.
9. K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou. Developing a context-aware electronic tourist guide: some issues and experiences. In *Proc. of CHI'00*, pages 17–24, The Hague, The Netherlands, 2000.
10. A. Cheyer, J. Park, and R. Giuli. IRIS: Integrate. relate. infer. share. *Proc. of 4th Intl. Semantic Web Conf. Workshop on The Semantic Desktop*, 2005.
11. A. K. Chopra, F. Dalpiaz, P. Giorgini, and J. Mylopoulos. Reasoning about agents and protocols via goals and commitments. In *Proc. of AAMAS'10*, pages 457–464, Toronto, Canada, 2010.
12. P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
13. K. Conley and J. Carpenter. Towel: Towards an intelligent to-do list. In *Proc. of AAAI 2007 Spring Symposium on Interaction Challenges for Intelligent Assistants*, pages 26–32, Stanford, CA, 2007.
14. M. T. Cox and A. Raja, editors. *Metareasoning: Thinking about Thinking*. MIT Press, Cambridge, MA, 2011.
15. H. S. Cramer, V. Evers, M. W. van Someren, and B. J. Wielinga. Awareness, training and trust in interaction with adaptive spam filters. In *Proc. of CHI'09*, pages 909–912, Boston, MA, 2009.
16. M. Czerwinski, E. Horvitz, and S. Wilhite. A diary study of task switching and interruptions. In *Proc. of CHI'04*, pages 175–182, Vienna, Austria, 2004.
17. C. da Costa Pereira and A. G. Tettamanzi. A belief-desire framework for goal revision. In *Proc. of 11th Intl. Conf. on Knowledge-Based Intelligent Information and Engineering Systems*

(*KES'07*) and *17th Italian Workshop on Neural Networks* (*WIRN'07*), pages 164–171, Vietri sul Mare, Italy, 2007.
18. M. Dastani and L. van der Torre. Programming BOID-plan agents: Deliberating about conflicts among defeasible mental attitudes and plans. In *Proc. of AAMAS'04*, pages 706–713, New York, NY, 2004.
19. J. Davitz, J. Yu, S. Basu, D. Gutelius, and A. Harris. iLink: Search and routing in social networks. In *Proc. of 13th Intl. Conf. on Knowledge Discovery and Data Mining* (*KDD'07*), pages 931–940, San Jose, CA, 2007.
20. A. N. Dragunov, T. G. Dietterich, K. Johnsrude, M. McLaughlin, L. Li, and J. L. Herlocker. TaskTracer: A desktop environment to support multi-tasking knowledge workers. In *Proc. of IUI'05*, pages 75–82, San Diego, CA, 2005.
21. T. Duong, D. Phung, H. Bui, and S. Venkatesh. Efficient duration and hierarchical modeling for human activity recognition. *Artificial Intelligence*, 173(7–8):830–856, 2009.
22. M. T. Dzindolet, S. A. Peterson, R. A. Pomranky, L. G. Pierce, and H. P. Beck. The role of trust in automation reliance. *Intl. J. Human-Computer Studies*, 58:697–718, June 2003.
23. J. L. Edwards and G. Scott. LOCATE intelligent systems demonstration: Adapting help to the cognitive styles of users. In *Proc. of AAAI'06*, pages 1937–1938, Boston, MA, 2006.
24. R. Falcone and C. Castelfranchi. Levels of delegation and levels of adoption as the basis for adjustable autonomy. In *Proc. of AI*IA'99*, pages 273–284, Bologna, Italy, 1999.
25. A. Faulring, B. Myers, K. Mohnkern, B. Schmerl, A. Steinfeld, J. Zimmerman, A. Smailagic, J. Hansen, and D. Siewiorek. Agent-assisted task management that reduces email overload. In *Proc. of IUI'10*, pages 61–70, Hong Kong, 2010.
26. A. Fern, S. Natarajan, K. Judah, and P. Tadepalli. A decision-theoretic model of assistance. In *Proc. of IJCAI'07*, pages 1879–1884, Hyderabad, India, 2007.
27. M. Fleming and R. Cohen. User modeling and the design of more interactive interfaces. In *Proc. of UM'99*, pages 67–76, Banff, Canada, 1999.
28. D. Fogli, N. Gelfi, M. Giacomin, and G. Guida. A computational model for adapting presentation to content in web interfaces. *Intl. J. on Artificial Intelligence Tools*, 19(6):783–818, 2010.
29. M. Freed, J. Carbonell, G. Gordon, J. Hayes, B. Myers, D. Siewiorek, S. Smith, A. Steinfeld, and A. Tomasic. RADAR: A personal assistant that learns to reduce email overload. In *Proc. of AAAI'08*, pages 1287–1293, Chicago, IL, 2008.
30. K. Gajos and D. S. Weld. SUPPLE: automatically generating user interfaces. In *Proc. of IUI'04*, pages 93–100, Funchal, Portugal, 2004.
31. D. Garlan and B. Schmerl. The RADAR architecture for personal cognitive assistance. *Intl. J. Software Engineering and Knowledge Engineering*, 1(2):171–190, 2007.
32. M. P. Georgeff and F. F. Ingrand. Decision-making in an embedded reasoning system. In *Proc. of IJCAI'89*, pages 972–978, Detroit, MI, 1989.
33. M. Gervasio, T. J. Lee, and S. Eker. Learning email procedures for the desktop. In *Proc. of AAAI 2008 Workshop on Enhanced Messaging*, Chicago, IL, 2008.
34. Y. Gil and V. Ratnakar. Towards intelligent assistance for to-do lists. In *Proc. of IUI'08*, pages 329–332, Gran Canaria, Spain, 2008.
35. A. Glass, D. L. McGuinness, and M. Wolverton. Toward establishing trust in adaptive agents. In *Proc. of IUI'08*, pages 227–236, Gran Canaria, Spain, 2008.
36. B. J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.
37. B. J. Grosz, S. Kraus, S. Talman, B. Stossel, and M. Havlin. The influence of social dependencies on decision-making. In *Proc. of AAMAS'04*, pages 782–789, New York, NY, 2004.
38. K. Haigh, L. Kiff, J. Myers, V. Guralnik, C. Geib, J. Phelops, and T. Wagner. The independent LifeStyle assistant: AI lessons learned. In *Proc. of IAAI'04*, pages 852–857, San Jose, CA, 2004.

39. H. Hexmoor, C. Castelfranchi, and R. Falcone, editors. *Agent Autonomy*. Kluwer, Boston, MA, 2003.
40. K. V. Hindriks, W. van der Hoek, and M. B. van Riemsdijk. Agent programming with temporally extended goals. In *Proc. of AAMAS'09*, pages 137–144, Budapest, Hungary, 2009.
41. E. Horvitz. Principles of mixed-initiative user interfaces. In *Proc. of CHI'99*, pages 159–166, Pittsburgh, PA, 1999.
42. E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The Lumière project: Bayesian user modeling for inferring the goals and needs of software users. In *Proc of UAI'98*, pages 256–266, Madison, WI, 1998.
43. E. Horvitz, C. M. Kadie, T. Paek, and D. Hovel. Models of attention in computing and communications: From principles to applications. *Comm. ACM*, 46(3):52–59, 2003.
44. W. Iba. When is assistance really helpful? In *Proc. of AAAI 2007 Spring Symposium on Interaction Challenges for Intelligent Assistants*, pages 62–63, Stanford, CA, 2007.
45. C. L. Isbell and J. S. Pierce. An IP continuum for adaptive interface design. In *Proc. of HCI International 2005*, Las Vegas, NV, 2005.
46. N. R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75(2):195–240, 1995.
47. K. Kamali, X. Fan, and J. Yen. Towards a theory for multiparty proactive communication in agent teams. *Intl. J. Cooperative Information Systems*, 16(2):271–298, 2007.
48. E. Kamar, Y. Gal, and B. J. Grosz. Incorporating helpful behavior into collaborative planning. In *Proc. of AAMAS'09*, pages 875–882, Budapest, Hungary, 2009.
49. D. Kinny, M. Ljungberg, A. S. Rao, L. Sonenberg, G. Tidhar, and E. Werner. Planned team activity. In *Artificial Social Systems*, LNAI 850, London, UK, 1994. Springer–Verlag.
50. J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Comm. ACM*, 40:77–87, 1997.
51. H. Lieberman and T. Selker. Out of context: computer systems that adapt to, and learn from, context. *IBM Systems Journal*, 39(3–4):617–632, 2000.
52. K. E. Lochbaum. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4):525–572, 1998.
53. O. Madani, H. Bui, and E. Yeh. Efficient online learning and prediction of users' desktop actions. In *Proc. of IJCAI'09*, pages 1457–1462, Pasadena, CA, 2009.
54. P. Maes. Agents that reduce work and information overload. *J. ACM*, 37(7):30–40, 1994.
55. R. T. Maheswaran, M. Tambe, P. Varakantham, and K. Myers. Adjustable autonomy challenges in personal assistant agents: A position paper. In *Proc. of AAMAS'03 Workshop on Agents and Computational Autonomy* (*AUTONOMY'03*), pages 187–194, Melbourne, Australia, 2003.
56. D. C. McFarlane and K. A. Latorella. The scope and importance of human interruption in human-computer interaction design. *Human-Computer Interaction*, 17(1):1–61, 2002.
57. F. Meneguzzi and J. Oh, editors. *Proactive Assistant Agents: Papers from the 2010 AAAI Fall Symposium*, Menlo Park, CA, 2010. The AAAI Press. Technical Report FS-10-07.
58. F. R. Meneguzzi, A. F. Zorzo, and M. da CostaMóra. Propositional planning in BDI agents. In *Proc. of 2004 ACM Symposium on Applied Computing* (*SAC'04*), pages 58–63, Nicosia, Cyprus, 2004.
59. D. Morley and K. Myers. The SPARK agent framework. In *Proc. of AAMAS'04*, pages 714–721, New York, NY, 2004.
60. K. Myers, P. Berry, J. Blythe, K. Conley, M. Gervasio, D. McGuinness, D. Morley, A. Pfeffer, M. Pollack, and M. Tambe. An intelligent personal assistant for task and time management. *AI Magazine*, 28(2):47–61, 2007.
61. K. Myers, J. Kolojejchick, C. Angiolillo, T. Cummings, T. Garvey, M. Gervasio, W. Haines, C. Jones, J. Knittel, D. Morley, W. Ommert, and S. Potter. Learning by demonstration technol-

ogy for military planning and decision making: A deployment story. In *Proc. of IAAI'11*, San Francisco, CA, 2011.
62. K. L. Myers and D. N. Morley. Human directability of agents. In *Proc. of 1st Intl. Conf. on Knowledge Capture* (*K-CAP'01*), pages 108–115, Victoria, Canada, 2001.
63. K. L. Myers and D. N. Morley. Policy-based agent directability. In H. Hexmoor, L. Castelfranchi, and R. Falcone, editors, *Agent Autonomy*. Kluwer, Boston, MA, 2003.
64. K. L. Myers and N. Yorke-Smith. A cognitive framework for delegation to an assistive user agent. In *Proc. of AAAI 2005 Fall Symposium on Mixed-Initiative Problem-Solving Assistants*, pages 94–99, Arlington, VA, 2005.
65. C. Nass, C. with Yen. *The Man who Lied to his Laptop: What machines teach us about human relationships*. Penguin, 2010.
66. S. Natarajan, H. Bui, P. Tadepalli, K. Kersting, and W. Wong. Logical Hierarchical Hidden Markov Models for modeling user activities. In *Proc. of 18th Intl. Conf. on Inductive Logic Programming* (*ILP'08*), pages 192–209, Prague, Czech Republic, 2008.
67. N. Negroponte. Agents: From direct manipulation to delegation. In J. M. Bradshaw, editor, *Software Agents*, pages 57–66. AAAI Press/The MIT Press, Menlo Park, CA, 1997.
68. A. Nguyen and W. Wobcke. An agent-based approach to dialogue management in personal assistants. In *Proc. of IUI'05*, pages 137–144, San Diego, CA, 2005.
69. A. Nguyen and W. Wobcke. An adaptive plan-based dialogue agent: integrating learning into a BDI architecture. In *Proc. of AAMAS'06*, pages 786–788, Hakodate, Japan, 2006.
70. D. A. Norman. How might people interact with agents. *Comm. ACM*, 37(7):68–71, 1994.
71. J. Oh, F. Meneguzzi, and K. Sycara. ANTIPA: an agent architecture for intelligent information assistance. In *Proc. of ECAI'10*, pages 1053–1054, Lisbon, Portugal, 2010.
72. J. Oh, F. Meneguzzi, and K. Sycara. Probabilistic plan recognition for intelligent information agents: Towards proactive software assistant agents. In *Proc. of 3rd Intl. Conf. on Agents and Artificial Intelligence*, 2011.
73. A. Ortony, G. L. Clore, and A. Collins. *The Cognitive Structure of Emotions*. Cambridge University Press, Cambridge, UK, 1988.
74. B. Peintner, J. Dinger, A. Rodriguez, and K. Myers. Task Assistant: Personalized task management for military environments. In *Proc. of IAAI'09*, Pasadena, CA, 2009.
75. P. Penna and C. Ventre. Optimal collusion-resistant mechanisms with verification. In *Proc. of 10th ACM Conference on Electronic Commerce* (*EC'09*), pages 147–156, Stanford, CA, 2009.
76. R. Picard. Affective computing: Challenges. *Intl. J. Human-Computer Studies*, 59(1–2):55–64, 2003.
77. A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: A BDI reasoning engine. In *Multi-Agent Programming*, pages 149–174, New York, 2005. Springer.
78. M. Pollack. Intelligent technology for an aging population: The use of AI to assist elders with cognitive impairment. *AI Magazine*, 26(2):9–24, 2005.
79. A. S. Rao and M. P. Georgeff. Modeling agents within a BDI-architecture. In *Proc. of KR'91*, pages 473–484, Cambridge, MA, 1991.
80. C. Rich and C. Sidner. COLLAGEN: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, 8(3/4):315–350, 1998.
81. D. Sarne, B. J. Grosz, and P. Owotoki. Effective information value calculation for interruption management in multi-agent scheduling. In *Proc. of ICAPS'08*, pages 313–321, Sydney, Australia, 2008.
82. S. Schiaffino and A. Amandi. User-interface agent interaction: personalization issues. *Intl. J. Human-Computer Studies*, 60(1):129–148, 2004.
83. A. Spaulding, J. Blythe, W. Haines, and M. Gervasio. From geek to sleek: Integrating task learning tools to support end users in real-world applications. In *Proc. of IUI'09*, pages 389–394, Sanibel Island, FL, 2009.

84. SRI International. CALO: Cognitive Assistant that Learns and Organizes. `pal.sri.com`, 2003–2009.
85. A. Stoica and N. Avouris. An architecture to support personalized interaction across multiple digitally augmented spaces. *Intl. J. on Artificial Intelligence Tools*, 19(2):137–158, 2010.
86. R. A. Subramanian, S. Kumar, and P. R. Cohen. Integrating joint intention theory, belief reasoning, and communicative action for generating team-oriented dialogue. In *Proc. of AAAI'06*, pages 1501–1507, Boston, MA, 2006.
87. M. Tambe. Towards flexible teamwork. *J. Artificial Intelligence Research*, 7:83–124, 1997.
88. A. G. Tesler. Networked computers in the 1990s. *Scientific American*, pages 86–93, Sept. 1991.
89. J. Thangarajah, L. Padgham, and M. Winikoff. Detecting and avoiding interference between goals in intelligent agents. In *Proc. of IJCAI'03*, pages 721–726, Acapulco, Mexico, 2003.
90. G. Tur, A. Stolcke, L. Voss, S. Peters, D. Hakkani-Tür, J. Dowding, B. Favre, R. Fernndez, M. Frampton, M. Frandsen, C. Frederickson, M. Graciarena, D. Kintzing, K. Leveque, S. Mason, J. Niekrasz, M. Purver, K. Riedhammer, E. Shriberg, J. Tien, D. Vergyri, and F. Yang. The CALO Meeting Assistant system. *IEEE Transactions on Audio, Speech, and Language Processing*, 18:1601–1611, 2010.
91. B. van Riemsdijk, M. Dastani, and M. Winikoff. Goals in agent systems: A unifying framework. In *Proc. of AAMAS'08*, pages 713–720, Sydney, Australia, 2008.
92. M. Winikoff. JACK intelligent agents: An industrial strength platform. In *Multi-Agent Programming*, pages 175–193. Springer, New York, 2005.