Like an Intuitive and Courteous Butler: A Proactive Personal Agent for Task Management

Neil Yorke-Smith, Shahin Saadati, Karen L. Myers, David N. Morley*
Artificial Intelligence Center, SRI International, U.S.A.
{nysmith,saadati,myers,morley}@ai.sri.com

ABSTRACT

The ability to proactively offer assistance promises to make personal agents more helpful to their users. We characterize the properties desired of proactive behaviour by a personal assistant agent in the realm of task management, and present an extended agent cognition 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-cognition over the agent's beliefs about its user's activities as well as over world state. We have implemented the resulting generic framework for proactive goal generation and deliberation as part of a personal assistant agent in the desktop domain.

Categories and Subject Descriptors

I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence—*Intelligent agents*

General Terms

Design, Human Factors, Theory

Keywords

Assistive agents, proactivity, task management, CALO

1. INTRODUCTION

Proactive behaviour is seen as an essential characteristic of autonomous and semi-autonomous agents [28]. We are interested in developing intelligent personal assistant agents that can aid a human in managing and performing complex tasks in an office desktop setting. Our overall goal is to reduce the amount of effort required by the human to complete the tasks she intends. Effort here encompasses both the activities necessary to perform the tasks, and the cognitive load in managing and monitoring them. Thus, a personal assistant agent may aid its user directly by performing tasks on her behalf or in conjunction with her [14], and indirectly through actions such as providing context for her work, minimizing interruptions, and offering suggestions and reminders [7].

We are exploring these ideas within a system for intelligent personalized assistance called *Cognitive Assistant that Learns and Organizes* (CALO) [35]. The focus for a CALO agent is to support a busy knowledge worker in dealing with the twin problems of information and task overload [23]. Specifically, CALO's task management capabilities are grounded in a module called *Project Execution Assistant* (PExA) [26]; in this paper we will refer simply to CALO. CALO's current task-related capabilities are grounded in *delegative behaviour*, in which the system adopts intentions only in response to being explicitly assigned goals by its user. In this fashion, CALO can 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.

An obvious limitation within the current delegative model is the lack of a *proactive* capability that would enable the agent to anticipate needs, opportunities, and problems, and then act on its own initiative to address them. We are interested in developing proactive behaviours along these lines, to increase the overall effectiveness of the system as a personal assistant for task management.

This paper describes our approach to operationalizing proactive behaviour within an assistive agent. We augment a Belief-Desire-Intention (BDI) [31] framework with a meta-cognition 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*: 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.

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 about how to assist with task management. 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). Thus, we do not focus on this aspect, instead drawing on the techniques studied.

Our primary contribution is to address the combined questions of when to take initiative for task management and what actions to manifest it. We provide an approach that is theoretically grounded while being practically realizable. Our work provides the basis for proactive assistance in the CALO system by means of context-sensitive suggestions. After situating our work in Sect. 2, we characterize the properties desired of proactive behaviour in Sect. 3. In Sect. 4 we present an extended BDI framework with a meta-level layer designed to support proactive goal generation, along with requirements on a theory of proactivity to drive this behaviour. In Sect. 5 we describe our operationalization of proactive assistance, implementing the framework in the CALO agent system.

*Authors listed in reverse alphabetical order.

Cite as: Like an Intuitive and Courteous Butler: A Proactive Personal Agent for Task Management, Yorke-Smith, Saadati, Myers, Morley, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

2. RELATED WORK

Efforts to design personal assistants often span several points on the continuum from zero to full automation. The Office Assistant in Microsoft Office was based "in spirit" on the Lumière project [19]. Although more limited in scope than CALO, Lumière shares an ambition to help a user. The domain of the project was user tasks in an office software package; Lumière deliberately considered proactive behaviour. In expressing the rationale, Horvitz et al. [19] 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.

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 [23] and, although not often construed as an agent, adaptive user interfaces While this type of proactivity is certainly within the scope of our work, we are interested in broader forms of proactive behaviour that extend beyond any single application. We will introduce two other forms of proactivity.

The Electric Elves project [4] 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. Electric Elves was focused on shared activities within a team setting.

There has been much recent work on assistive technologies to aid people with cognitive disabilities in managing their daily activities, in which proactivity plays an important role (e.g., [16]). 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.

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 [31] to incorporate notions of joint beliefs and commitments. For example, Joint Intention theory [5] formalizes the communication acts between agents to establish and maintain joint belief and intention. SharedPlans theory [14] specifies collaborative refinement of a partial plan by multiple agents, handling hierarchical action decomposition and partial knowledge of belief and intention.

COLLAGEN [32] 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 Joint Intentions have focused on dialogue management in agent collaboration (e.g., [21]). Applications of SharedPlans have included also task allocation and analysis of helpful behaviour [15], but are limited in terms of proactive scope. Proactive sharing of information, in an extended SharedPlans formulation, has been studied in the multi-agent team setting [22].

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. 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.

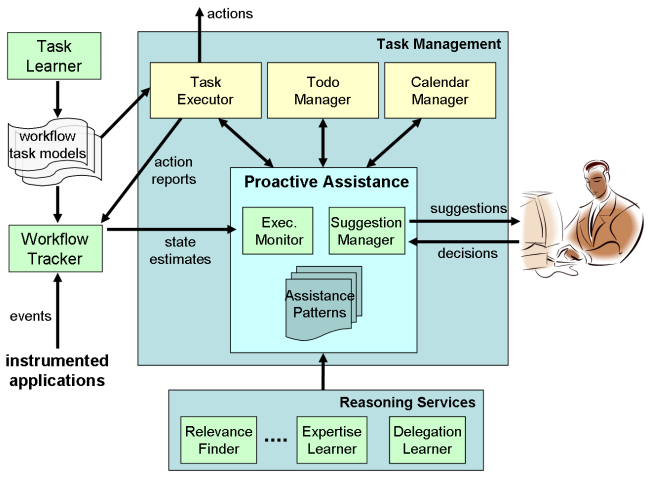

Figure 1: Task management in the CALO system

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 speak to how an agent weighs the cost and benefits of potential goals and the plans to achieve them to assess which are appropriate to perform [34, 9]. 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 is capable of performing. The requirements for and organization of these latter reasoning functions are our topic for this paper.

3. TOWARD HELPFUL PROACTIVE ASSISTANCE FOR TASK MANAGEMENT

Ethnographic studies of human work habits and task management (e.g., [1, 7]) 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. It is these areas where assistance technology can thus provide greatest utility.

Our starting point is that the user has entered a description of her tasks and tasks assigned to her agent into an electronic to-do list [1, 6], thus obviating the need to infer user intent [19, 9]. We take these 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.

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 *Todo Man-*

ager, and a *Calendar Manager*. They are shown in Fig. 1, together with the modules that operationalize the proactive behaviour we describe in the coming sections.

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* 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. 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 library might be executable only by the user, only by the agent, by either, or by both together. The Task Executor is built as a SPARK agent [24], with the procedural aspects of workflows encoded in the SPARK procedure representation language. Workflows can be specified manually or taught to the system through a learning by demonstration framework [11].

The Todo Manager provides a framework to aid the user in organizing and tracking tasks. The CALO Todo Manager, called *Towel* [6], 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 [12]. 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). Further, 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). The Calendar Manager supports the user in scheduling meetings and managing temporal commitments [2]. Individual calendar entries are defined by their start and end times, the organizer, the participants, category, importance, and location.

These task management tools provide a range of services and information that can be used to determine user state to inform proactive reasoning. One such capability is an estimation of user 'busyness', 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 Todo Manager.

Complementary services within the CALO system can further inform reasoning for proactive assistance. An *Expertise Finder* identifies people within an organization who may have expertise on particular topics. A *Relevance Engine* identifies documents and emails that are potentially related to tasks. A *Delegation Learner* develops models that recommend specific individuals to whom tasks might be assigned, based on prior delegation events. Finally, and significantly for the context of proactive behaviour, a *Workflow Tracker* estimates the workflow the user is currently pursuing from her desktop activities. We describe it in detail in Sect. 5.

3.2 Characterizing Proactive Behaviour

Within such a setting, three challenges must be addressed in order to develop effective personal assistant agents. 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 merchant to purchase an item on its user's behalf). 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 aiding 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 when to take initiative for task management and what actions to manifest it. Although we will not address the subject in detail, it is important to recognize that the agent should weight whether the actions it is considering will be helpful to the user. Moreover, since there will be a measure of uncertainty about the current situation, the user's goals and focus of attention, and the effects of its actions, the agent should be careful, safe, and deferent to the user. As will be explained, our approach is to have the agent by default offer suggestions that the agent acts, rather than make decisions or take action in a fully autonomous fashion.

When and 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 [20, 10]. 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?

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 [28, 2].

3.3 Examples of Proactive Behaviour for Task Management

Fig. 2 provides a selected list 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*. All the items in Fig. 2 except those in italic are currently implemented in the CALO system. The following scenario illustrates how a proactive behaviour 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 people, and your commitments for the week ahead. CALO assesses that your workload is likely to be uncomfortably 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 behaviour 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

- **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*

Figure 2: Possible proactive activities in task management

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 are designed for task-focused operation, although not task-focused proactivity.

The second type of proactive behaviour, 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 assist with any individual task on the user's to-do list, but rather in response to a higher-level motivation (namely, workload balancing).

3.4 Principles for Proactive Behaviour

To guide the development of proactive agent behaviour, we set out nine principles, akin to the principles for intelligent mixed-initiative user interfaces [18]: **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; **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.

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 behaviour sought in the Lumière project: "The sensibility of an intuitive, courteous butler ... potentially valuable suggestions from time to time ... genuine value ... minimal disturbance." [19]

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 well as working entirely without it [4, 16, 18, 13]. 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 [28, 13], 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.

4. A BDI FRAMEWORK FOR PROACTIVE ASSISTANCE

Having characterized helpful proactive assistive behaviour, we now introduce — more concretely — an extended BDI model of agency designed to support such behaviour. 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.

4.1 Background

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) [31]. 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 [30], PRS, and SPARK [24] implement these decision-making processes as a combination of base- and meta-level reasoning: 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 behaviour by an intelligent assistant.

The reasoning that enables CALO's current (non-proactive) task-related capabilities draws on a *delegative BDI model* [25]. This 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 [3, 8]) that are simply assumed of goals in many implemented BDI systems.[1] The delegative BDI model also incorporates forms of user-specified guidance and preferences on

[1] 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 [30].

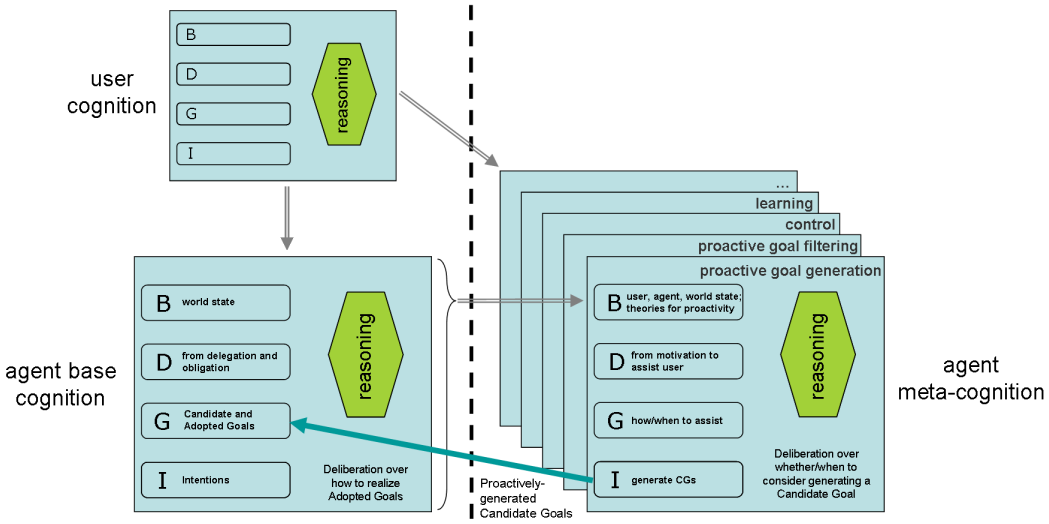

Figure 3: Extended BDI agent architecture for proactive assistance

the execution of these tasks, and on the agent's cognition, called *advice*. However, the framework in the various aspects of its meta-reasoning does not encompass deliberation in order to manifest proactive assistance by generating Candidate Goals.

4.2 Architecture for Proactive Assistance

Fig. 3 depicts proactive goal generation in an extension of the delegative BDI agent architecture. As usual in a BDI formulation, 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.

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.

Candidate Goals (CGs) are created through two mechanisms. At the base level, they arise from the agent's 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 Sect. 4.3. 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 helpful activity.

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 [19, 33, 10]. This filtering of proactively-generated CGs is accomplished by the meta-layer denoted *proactive goal filtering* in Fig. 3.

The architecture does not impose the characteristics of the agent's behaviour, 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. Sect. 5 describes our implemented approach to CG generation, adoption, and assistance manifestation.

4.3 Elements of Proactivity

Sect. 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 requirements on a *theory of proactivity* designed to support a proactive agent in meeting these challenges. This theory is composed of subtheories for *user desires*, *helpful activity*, 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 OCC cognitive model [29]) as well as explicitly stated *achieve* and *replenishment* goals.[2] Since interest goals differ between individuals, a helpful assistive agent requires such a model (perhaps learned) of its user, in order to assess the value of agent actions.

Further, meta-reasoning over the sufficiency of the model (i.e.,

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

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 Helpful Activity. A theory of helpful activity defines the principles to direct the agent's reasoning to determine what actions would most help 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 helpful activity include bayesian modelling [19] and logical rules [10].

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 [17]), and possibly dedicated data structures and reasoning (e.g., [36]). 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: ASSISTANCE PATTERNS

As Grosz and Kraus [14] note, 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 reasoning over such constructs.

Similarly, our implementation of the meta-level components for *proactive goal generation* and *filtering* in Fig. 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 representation that bridges the gap from the high-level desire $\hat{\mathbf{d}}$ to help the user to more concrete motivations for the agent. 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, focus of attention), the agent's own state, and the world state.

The assistance patterns of an agent encode the manifestations of proactive behaviour given in Fig. 2. Two example APs are shown in Fig. 4 as pseudocode, with `cue` denoting the triggering conditions. 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. 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

```
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'
```

Figure 4: Sample assistance patterns

it perform this step. A third example is an AP that cues from a belief that the user has overlooked possible synergy between two tasks (contrast [36]); 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 [24] augmented by an explicit representation of Candidate and Adopted Goals. All the activities of Fig. 2 are implemented, except those denoted in italic. The pseudocode of APs such as those in Fig. 4 is readily translatable into SPARK's procedural representation, leveraging its meta-level events and meta-reasoning capabilities.

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. 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.

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 Sect. 3.1, part of the context of task management in the CALO system is the user's list of her tasks in the Todo 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 one workflow, the user first downloads the paper and the review form attached to the review request email. Next, the paper is printed, and the review form is filled out. Finally, the completed review form is sent back as a reply to the request email. 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 Fig. 1, we instrumented 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 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.

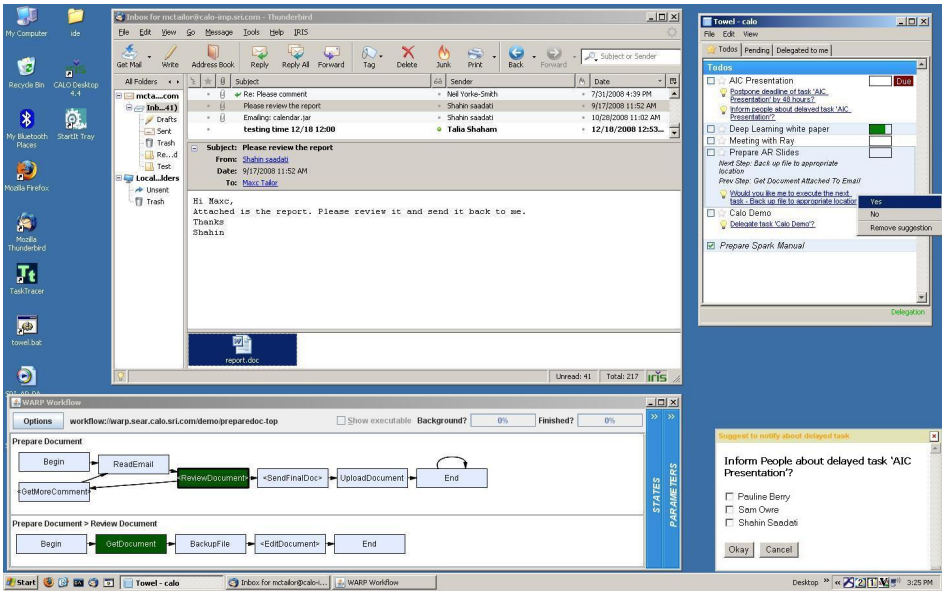

Figure 5: Task-focused proactive assistance provided by CALO

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* [27] as its representation of the workflow model.[3] Workflow recognition can then 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. This information from the *Workflow Tracker* is provided to the Execution Monitor (Fig. 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.

Suggestion Manager filters CG generated by APs.

The principle of goal-directed focus in deliberation [3, 5] applies to APs as much as it does to other aspects of agent deliberation, such as the agent's regular control loop (compare Fig. 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 it should consider checking the AP triggers at all, and over whether to adopt any CGs the APs may generate.

To support this kind of meta-reasoning, which instantiates the theory of helpful activity, and further to provide contextually sensitive assistance, we implemented a *Suggestion Manager*, as shown in Fig. 1. The Suggestion Manager provides CALO with the *proactive goal filtering* meta-layer depicted in Fig. 3. 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 our APs result in a CG to create a suggestion to the user; none lead to autonomous action. Thus, we defer towards the interface end of the Interface–Proactivity continuum. 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. Fig. 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 behaviour to be safe and deferential, minimizing the cost to the user of unwanted or inappropriate proactive activity. However, Suggestion Manager may 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."

We have implemented other interaction modalities besides the peripheral sidebar display depicted in Fig. 5. In order of increasing demand of attention (and so disruptive cost if inappropriate), these include 'toast' pop-up notifications, 'CALO Chat' instant messages, and non-modal and modal dialog boxes, in addition to email.

The Suggestion Manager's reasoning accounts for the user's interaction preferences, her current activity (to avoid acting or interrupting out of context), 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 [18, 10]. Future work is to enhance the reasoning, drawing on more sophisticated models pioneered in other systems (e.g., [19]).

[3] 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 $\text{OpenFile}(X) \rightarrow \text{EditDocument}(X) : 0.8$. Here, variable X ranges over the set of documents in the system, and 0.8 represents the transition probability. Similarly, the observation model is $\text{OpenFile}(X) \rightarrow \text{WordOpenEvent}(X) : 1.0$. 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.

6. CONCLUSION

Proactive behaviour by an assistive agent — which encompasses more than acting directly to achieve an assigned task — promises to make such an agent more helpful to its user. We have characterized the properties desired of such behaviour, and presented an extended agent cognition 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 a compiled 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 framework is informed by 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.

Helpful proactive assistance beholds significant technical challenges. Besides the theoretical underpinnings for a principled approach, agent behaviour must most importantly be within context [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 behaviour more subtly to explicit and implicit user feedback. Directions for the future include a providing a logical formalism such that guarantees can be made about agent behaviour by reasoning over APs within an instantiation of the theory of helpful activity; a more sophisticated user model including estimation of the user's emotive state; and the potential of building CALO towards an affective agent.

Acknowledgements. We thank the anonymous reviewers for their comments, and H. Bui and the CALO Activity Recognition 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.

7. REFERENCES

[1] 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*, 2004.

[2] 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*, 2006.

[3] M. E. Bratman, D. J. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 14:349–355, 1988.

[4] H. Chalupsky, Y. Gil, C. Knoblock, K. Lerman, J. Oh, D. Pynadath, T. Russ, and M. Tambe. Electric Elves. *AI Magazine*, 23(2), 2002.

[5] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[6] 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*, 2007.

[7] M. Czerwinski, E. Horvitz, and S. Wilhite. A diary study of task switching and interruptions. In *Proc. of CHI'04*, 2004.

[8] 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*, 2004.

[9] A. Fern, S. Natarajan, K. Judah, and P. Tadepalli. A decision-theoretic model of assistance. In *Proc. of IJCAI'07*, 2007.

[10] M. Fleming and R. Cohen. User modeling and the design of more interactive interfaces. In *Proc. of UM'99*, 1999.

[11] M. Gervasio, T. J. Lee, and S. Eker. Learning email procedures for the desktop. In *Proc. of AAAI 2008 Workshop on Enhanced Messaging*, 2008.

[12] Y. Gil and V. Ratnakar. Towards intelligent assistance for to-do lists. In *Proc. of IUI'08*, 2008.

[13] A. Glass, D. L. McGuinness, and M. Wolverton. Toward establishing trust in adaptive agents. In *Proc. of IUI'08*, 2008.

[14] B. J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[15] 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*, 2004.

[16] 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*, 2004.

[17] K. V. Hindriks and M. B. van Riemsdijk. Using temporal logic to integrate goals and qualitative preferences into agent programming. In *Proc. of DALT'08*, 2008.

[18] E. Horvitz. Principles of mixed-initiative user interfaces. In *Proc. of CHI-99*, 1999.

[19] 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*, 1998.

[20] E. Horvitz, C. M. Kadie, T. Paek, and D. Hovel. Models of attention in computing and communications: From principles to applications. *Communications of ACM*, 46(3):52–59, 2003.

[21] N. R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75(2):195–240, 1995.

[22] K. Kamali, X. Fan, and J. Yen. Towards a theory for multiparty proactive communication in agent teams. *Intl. J. of Cooperative Information Systems*, 16(2):271–298, 2007.

[23] P. Maes. Agents that reduce work and information overload. *J. ACM*, 37(7):30–40, 1994.

[24] D. Morley and K. Myers. The SPARK agent framework. In *Proc. of AAMAS'04*, 2004.

[25] 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*, 2005.

[26] K. Myers, et al. An intelligent personal assistant for task and time management. *AI Magazine*, 28(2):47–61, 2007.

[27] S. Natarajan, H. Bui, P. Tadepalli, K. Kersting, and W. Wong. Logical Hierarchical Hidden Markov Models for modeling user activities. In *Proc. of ILP'08*, 2008.

[28] D. A. Norman. How might people interact with agents. *Communications of ACM*, 37(7):68–71, 1994.

[29] A. Ortony, G. L. Clore, and A. Collins. *The Cognitive Structure of Emotions*. Cambridge University Press, Cambridge, 1988.

[30] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: A BDI reasoning engine. In *Multi-Agent Programming*. Springer, 2005.

[31] A. S. Rao and M. P. Georgeff. Modeling agents within a BDI-architecture. In *Proc. of KR'91*, 1991.

[32] 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.

[33] D. Sarne, B. J. Grosz, and P. Owotoki. Effective information value calculation for interruption management in multi-agent scheduling. In *Proc. of ICAPS'08*, 2008.

[34] S. Schiaffino and A. Amandi. User-interface agent interaction: personalization issues. *Intl. J. of Human-Computer Studies*, 60(1):129–148, 2004.

[35] SRI International. CALO: Cognitive Assistant that Learns and Organizes. `caloproject.sri.com`, 2003–2009.

[36] J. Thangarajah, L. Padgham, and M. Winikoff. Detecting and avoiding interference between goals in intelligent agents. In *Proc. of IJCAI'03*, 2003.