Towel: Towards an Intelligent To-Do List

Kenneth Conley, James Carpenter

SRI International
333 Ravenswood Avenue, Menlo Park, CA, 94025-3493
conley@ai.sri.com, james.carpenter@sri.com

Abstract

In this paper we describe Towel, a task management application that couples a user's to-do list with a software personal assistant. This to-do list provides a unified environment for managing personal tasks, delegating tasks to the software personal assistant, and collaborating with other users. We use to-do list and instant messaging metaphors to enable the user to initiate, manage, and modify complex agent-executed tasks. Through simple operations on to-do items and direct-manipulation chat, we envision many seamless interactions between the user and various AI technologies that ultimately result in saving the user work, reducing cognitive load, and improving task performance.

Introduction

For the past three years we have been developing a Project Execution Assistant (PExA) that combines various AI technologies to assist a user in task and time management (Myers *et al.* 2006). These AI technologies include a plan execution agent, a time management agent, task explanation, multiple task learning systems, procedure modification, concept learning, and more. While these components are too varied to expose entirely through one interface, we sought an end-user application metaphor that would allow us to present the core capabilities and integrate many of the peripheral capabilities as well.

For guidance on how to build a user interface for our assistant, we looked to applications designed for human task management, e.g. to-do lists. We wanted to create a fluid relationship between tasks a user is doing and tasks the assistant is doing on the user's behalf. Bellotti et al. have done extensive anthropological research on to-dos have built multiple software prototypes, including e-mail-based solutions (Bellotti *et al.* 2003) and a to-do list application called TaskVista (Bellotti *et al.* 2004). A to-do list application like TaskVista was appealing, as the to-do list could be placed in the peripheral part of the user's display and provide status information on our assistant. We easily envisioned dispatching tasks to our assistant by simply adding an item to a to-do list. But to-do lists in general do not have any affordances for communication: our assistant has to communicate with the user to solicit parameters for tasks, convey status, support interactions, and display results.

The communication requirements for our task execution are varied. At times tasks will be short in duration, such as finding hotels for a user, where the assistant solicits parameters immediately and quickly returns results to the user. Our assistant is also capable of tasks larger and longer in scope, such as arranging a client visit, that require it to solicit parameters and guidance from the user over time as the process becomes more defined; it might ask the user a question hours, days, or evens weeks after the task was initiated. We needed to select a style of digital communication that would have the proper types of affordances for these tasks.

We explored several digital communication metaphors, including e-mail, which handles long-term communication well but makes rapid communication cumbersome, Web pages, which are well-suited to short tasks but poor for long-term monitoring, and spoken interfaces, which we felt were too intrusive in a workplace. Instant messaging (IM) seemed ideal: in normal human-to-human IM communication, users regularly engage in rapid conversations, but the other participants can take an arbitrary length of time to respond. IM also provides a model for interruption (opening a chat window and short sounds), status information about contacts, and a peripheral list similar to a to-do list application. We also liked that users can hide interactions that aren't currently active by closing the chat window.

We have developed a task management application called Towel that builds on principles from to-do management and instant-messaging communication to provide a unified environment for both human- and agent-based task management. Towel's to-do list window is isomorphic to a contact list window for Instant Messaging: each contact is instead a to-do and double-clicking on a to-do opens up a chat window about that to-do. The to-do list window is a central view in which the user can enter her own to-dos, delegate tasks to our assistant, and monitor execution of delegated tasks. The chat windows are task-specific views where the user and assistant can communicate to refine, investigate and manipulate a task. We are also integrating Towel with various AI technologies to enable Towel to react to information about the user's current workload, such as reducing visual

overload, suggesting tasks our assistant can perform, and providing intelligent notifications.

The rest of this paper is organized as follows. We first introduce the architecture of Towel and selected portions of the underlying CALO system. We then discuss the to-do list user interface and the chat user interface. This is followed by a discussion of our delegation model and the deployment of Towel. We conclude with a discussion of future work.

Architecture

Towel is part of the CALO system (caloproject.sri.com/), which is a large-scale effort to build a personalized software assistant for the office domain. The CALO system is very large with many changing components, which has an important effect on the design of Towel: rather than tightly integrate with the CALO system as a whole, we have decided to do lightweight integrations on a one-to-one basis with components. Features are selectively enabled or disabled as other components change availability. With no other components present, Towel is just a to-do list application, but as more components are made available to Towel, it increases in effectiveness. The full subset of CALO components related to task execution is called the Project Execution Assistant (PExA), which is described in Myers et al. 2006.

The most important component for Towel is the Task Manager, which performs tasks within the CALO system. The Task Manager regularly sends Towel information on tasks being performed, which are then presented within the Towel user interface. The Task Manager and process models are built on top of SPARK (Morley & Myers 2004), which is a Belief-Desire-Intention framework similar to PRS (Georgeff & Ingrand 1989).

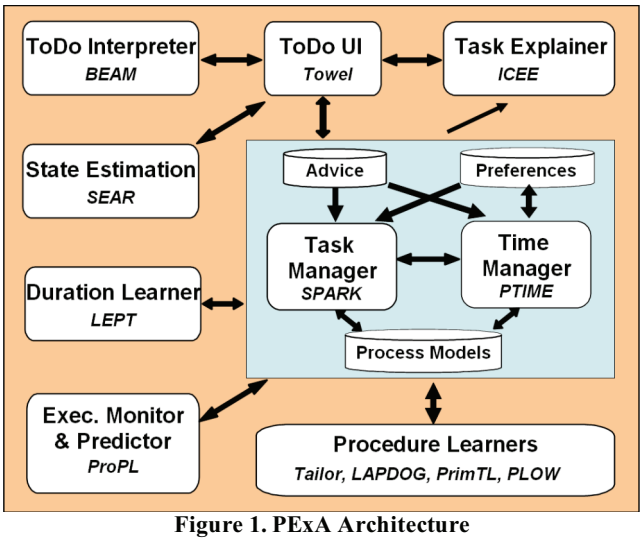

Figure 1. PExA Architecture

To-do lists and Tasks

To-dos within the Towel are nothing more than textual reminders: "walk the dog," "groceries," "meeting with Bill." On the Towel to-do list, users can perform modifications on to-dos such as grouping, tagging, checking (completing), delegating, setting deadlines, hiding, star-ing, and adding attachments. *Tasks* represent an action, something being done. There are also *CALO tasks*, which are tasks that the CALO system can perform. Although to-dos are a reminder of work and tasks are the work being done, we represent both using an iCalendar-VTODO-based ontology. To-dos and tasks share the same properties such as deadlines, completion status, and delegation history, but tasks are also linked to a task ontology. The identical representations allow for to-dos to become tasks and vice-versa.

The user can go beyond simple text to-dos by dragging files and URLs onto the to-do list, where they can be clicked on to open the resource. This allows the to-do to be the thing that needs to be done or a pointer to how or where it can be done. We also wish to add the ability to drag any semantic object from the CALO application environment (project, e-mail, person, etc…) onto the to-do list, either as a resource for a to-do or as the to-do itself.

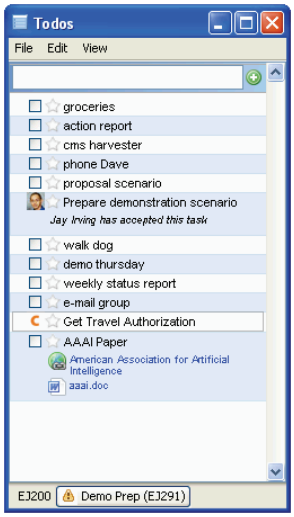

Figure 2. Towel To-Do List

Given the large number of to-dos a user might have, one of our design goals for the to-do list was to enable the user to hide to-dos. Contact lists for IM clients allow users to group contacts together and also hide contacts that are offline, and we similarly allow the user to organize to-dos into collapsible groups and explicitly hide to-dos for a period of time. We also allow the user to add tags and deadline information that can be used to filter to-dos by keyword or due date.

We are currently implementing a "Do it Now/Timecard" view of the to-do list, in which the user would select a to-do item that she is currently performing and track time spent on it. In this enhanced view, more information about

the selected to-do would be displayed and less immediate to-do items would be hidden. Data about which task the user has selected is sent to the SEAR task estimator, which can then make future predictions about which to-do the user is performing. We hope to use this data from SEAR to improve this view over time. For example, Towel could recommend different CALO tasks to assist the user in the completion of the task she is working on or it could delay the display of less important chat messages if she is busy.

The perfect, telepathically enabled to-do list would show you just one item: the next thing you are going to do. We can't implement such a system, but we hope to observe the user's behavior in hiding and performing to-dos, apply learning components such as SEAR to reduce the visual overload of a lengthy to-do list, and thus help the user focus on what needs to be done.

Converting to-dos into tasks

One of the main goals of the CALO personal software assistant is to save the user time and effort. Thus, a principal goal of Towel is to convert user to-dos into CALO tasks, i.e. to complete work on the user's behalf. Our model for converting user to-dos into CALO tasks is a *delegation* model (Myers & Yorke-Smith 2005): the user must tell CALO to take over a to-do. We provide a variety of unobtrusive mechanisms for delegation in order to allow delegation to occur at various points in a to-do's lifecycle.

The first opportunity a user has to delegate is while the to-do is being entered. At the top of the Towel to-do list is a textbox in which the user can type in new to-dos as well as search their to-do list. We have combined the operations of adding and searching in order to reinforce the reminder nature of to-do lists: a typical user may have as many as 70 electronic to-dos (Bellotti *et al*. 2004), so a to-do being added may already be on the to-do list, or it may be similar to one previously completed and reusable. Below the list of matching to-dos, Towel also suggests possible CALO tasks using a simple keyword-matching algorithm: the "Schedule a Meeting" task will be listed if the user types "Schedule" or "Meeting"; it will also match partial substrings like "sch." If she clicks on the suggestion, the new task is delegated to the Task Manager. We believe that this sort of opportunistic mechanism helps the user discover CALO tasks, especially with task learning extending the capabilities of the system.

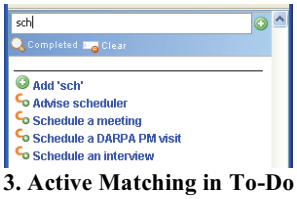

Figure 3. Active Matching in To-Do List

The keyword-matching algorithm can provide quick feedback for partial to-do phrases, but it is not very robust as it misses synonyms and alternate phrasings. We pass the complete text of new to-dos to the BEAM concept learning component. BEAM learns alternate phrasings for tasks based on volunteer-entered paraphrases (Chklovski & Gil 2005) and can find task parameters in to-do text, though we have not yet integrated the latter functionality. If BEAM is able to map the to-do text to the task ontology, Towel will annotate the to-do with a suggestion to invoke the CALO task. This suggestion is passive and the user may delete it.

The keyword- and concept-learning-based approaches will often fail as to-dos are often semantically opaque — even to other people — so we also enable the user to delegate to-dos to CALO using a chat dialog. If the user double-clicks on one of her to-dos or if she selects "Delegate to->CALO" on the right-click menu, Towel opens a chat window that starts with CALO asking, "How can I help you with 'name of task'?" The user can then select a CALO task from a menu to have it delegated to CALO.

We are exploring other approaches for improving the ease with which tasks can be delegated to CALO. One hope is that users will enter sufficient to-do metadata to narrow down a list of CALO tasks to suggest to a user. For example, if the user drags a PowerPoint presentation onto a to-do item, Towel can assume that the to-do is presentation-related. Similarly, if the user drags multiple people onto a to-do item, Towel might offer to schedule a meeting with those people or send them a file that is also attached to the to-do.

Chat-based Task Communication

We implemented our own chat system as a direct-manipulation, instant-message client, where each chat window represents a to-do item or CALO task. Direct-manipulation and contextualizing each chat window went hand-in-hand: the contextualized chat windows limited the number of possible operations, which favors a direct-manipulation approach; using direct-manipulation instead of natural language prevented the user from entering commands outside the context of the task, such as attempting to purchase a laptop in a chat window about scheduling a meeting.

Although the previous versions of our CALO communication architecture were chat-based, this was the first version to explicitly emulate IM clients and their semantics: the chat windows are designed to look similar to iChat (www.apple.com/ichat/), with alternating chat balloons emanating from a user icon; notification of new messages opens a new chat window or the task bar icon for the window; chat windows can be closed without losing the state of the chat. The main difference between our chat and an actual IM client is that we do not allow free-form text to be entered.

In designing our direct-manipulation IM client, we were heavily influenced by MERL's Diamond Help (Rich *et al.* 2006). Diamond Help is a direct-manipulation interface for managing home appliances. The Diamond Help screen is divided into two halves: the top half resembles an iChat instant messaging application, but with direct-manipulation input instead of free-form text input. The bottom half of the screen displays an appliance-specific control panel that allows entering of more complex input, such as scheduling settings for a VCR. Towel's chat windows are generally a single window, with input forms appearing inside of chat balloons, but we decided to adopt the DiamondHelp's split-screen approach for input forms that were too complex to fit in a chat balloon.

Task explanation and modification

We also adopted Diamond Help's approach of including several standard buttons within each active chat balloon. For our system, we chose "what are you doing right now?" and "why?" as our buttons, which we hope will improve the user's trust in the agent by allowing the user to investigate the agent's actions.

The "what" button prompts the agent for status on what it is currently doing; we found in our previous dialogs, the process models were was sending status updates too often and caused frequent interruption. The "why" button integrates with ICEE, a task explanation component based on Inference Web (McGuinness and Pinheiro da Silva 2004; McGuinness et al. 2006). ICEE explains both provenance and execution information associated with SPARK process models. The "why" button is a natural extension to the "what" button: for example, the user asks what is being done about the task, the agent responds with what it is doing, and then the user asks "why" because she is uncertain why the agent has chosen to take that particular course of action.

In the future, we wish to push the interaction one step further by coupling task explanation with procedure modification. For example, if the explanation component reports that a procedure is blocking on a condition, the user can automatically be offered the option of removing or modifying that condition.

Task duration prediction

One of the simplest, most useful, and most problematic features we implemented is an animated thought balloon to indicate that their agent will be sending more messages about that task to the user. This is also an indicator of whether or not a task is ongoing or completed. While this feature was initially appreciated by users, we found that an unintended dependence formed: people used this to tell whether or not the underlying CALO system had crashed (many of our users were developers or testers working on unstable versions). Extra frustration resulted if Towel failed to detect a crash and continued to display the thought balloon. A more important problem for all of our users was that the thought balloon didn't provide any indication when the next message would arrive, which could be seconds, minutes, or even longer away.

These problems have arisen in part because there are gaps in our IM metaphor: an agent is always present and IM chats don't complete like tasks; it is more appropriate to say that they are suspended until a later time. Instant messaging has several conventions, provided by both the chat application and people chatting, to help build expectations as to when the next message will arrive. IM applications generally inform you when the other person is typing and the iChat application, in particular, uses a thought balloon to convey this. IM clients can also set a contact to "away" status if the person's computer is idle. A person in a chat can suspend the conversation themselves by saying "be right back" or "in a meeting." An agent doesn't type, never leaves its computer, and is never interrupted, so those IM conventions do not map easily.

We are working on refining our feedback to provide better expectations as to when the agent will respond. We now use a task and to-do duration predictor to inform the user how long the overall task is expected to take. We hope to make this prediction more specific in the future and provide the user an estimate of how long until CALO next communicates about that task. If it is expected to be a long time, the user would know that she can close the chat window and work on other things.

Forms, Custom Forms, and Automatic Forms

In order to solicit input for tasks within a chat dialog, we needed to support both pre-engineered and learned tasks. To best support both, we developed a generic form-rendering component called SwingFormRenderer but also allow pre-engineered tasks to provide custom form renderers. Most forms are displayed inline within a chat balloon, though custom forms and forms too large for a chat balloon are displayed in the bottom half of a split window.

When a task is learned by a CALO component and added to the task ontology, we automatically generate forms to solicit input parameters and display results. Thus, task learning components are not required to specifically implement or encode user interactions.

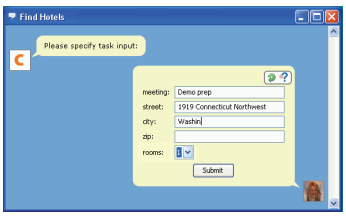

Figure 4. Auto-generated Form for Learned Task

Some tasks have more challenging interaction requirements and can override these default forms. For example, two custom input forms have been developed for

PTIME (Berry *et al*. 2006), which provides the scheduling component of CALO. These forms make it easier to handle the many optional parameters that scheduling can use and also help the user visualize scheduling alternatives and other participants' schedules.

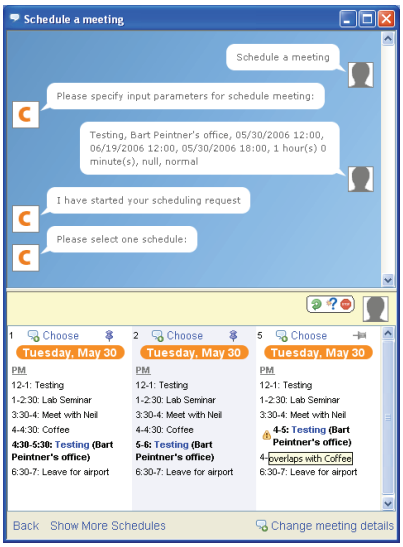

Figure 5. PTIME Schedule Selection Form

Delegation

Our delegation-based model enables the user to delegate tasks to her CALO or to other users. Although the underlying technology for these two types of delegation are different, we present them identically to the user as we wish for the transition between human-executed and agent-executed tasks to be seamless. A user may delegate a task to another person, who then delegates that task to her CALO. From the perspective of the user, all that changes is what or who is executing the task, which we indicate with an icon next to the to-do item: a checkbox for the user's items, a 'C' icon for CALO, or another user's personal icon. We also change the type of chat that opens when the user double-clicks on the item:

- *User's to-do*: opens the "How may I help you with…" delegation chat described in the "Converting to-dos into tasks" section.
- *CALO task*: opens a task-contextualized chat as described in the "Chat" section.
- *Delegated to another person*: in the future this will open an IM chat with the other person.

User-to-user delegation. Delegating a to-do to another user transports the entire metadata for the to-do to the other user; changes to the metadata by the delegatee are then fed back to the delegator. Thus, when the delegatee marks the to-do as complete, the delegator immediately sees the to-do on her own to-do list marked as complete. We don't expect a to-do-list-based delegation mechanism will supplant e-mail as the primary mechanism for users delegating tasks to other users, but the to-do list mechanism does have its advantages: users get immediate feedback on changes to the state of the to-do, the delegation is directly linked to the delegator's own to-do list, a user can use their to-do list to plan a task prior to delegation (e.g. a user might plan a trip by creating to-dos for "book flight", "book hotel", "pack luggage", and then delegate those as appropriate), and to-do metadata is explicitly transported to the delegatee.

Research by Okamoto showed that agent-assisted contingency management could potentially result in large gains in organizational performance; there were also small gains for increased communication speed (Okamoto *et al*. 2006). We are exploring both of these potentials by with a task delegation learning component. Based on organizational relationships, current workload, task type, and other features, this component can learn whether or not a user will accept or reject a task delegated by another person. We are currently testing a feature that enables a user to have tasks automatically accepted or rejected on her behalf, which can reduce interruptions during a period of high workload and improve response times to the person delegating the task. In the future we might also use delegation learning to proactively help a user find someone who can perform the task.

User-to-CALO delegation. User-to-CALO delegation is one of the primary goals of Towel and in the "Converting to-dos into tasks" section we discussed the several to-do-list-based and chat-based mechanisms we have implemented.

CALO-to-User delegation. We have not modeled this type of delegation, but, in the future, we envision that it will be necessary for the user and CALO working together towards a shared goal (Allen, Blaylock & Ferguson 2002). In a reimbursement process, for example, a CALO can perform many of the tedious tasks of filling out and submitting forms, but it cannot fulfill legal requirements such as signing a document. We also envision that CALO could perform many of the preparatory actions for a task, such as gathering information for the user to review and attaching this information to a to-do.

CALO-to-CALO delegation. We also wish to expand the capabilities of our Task Manager to better support CALO-to-CALO delegation. Instead of one CALO requesting that another CALO commit to a meeting on behalf of its user, we wish to model it as one CALO delegating the "commit to meeting" task to another CALO. This difference, while subtle, would enable several improvements to our overall system: Towel could allow the user to mediate whether or not the delegated task is accepted, the user would gain additional privacy, and we could also integrated in adjustable autonomy technology to control the level of interruption.

Deployment

Towel has been successfully deployed on several users' desktops for several months as a to-do list application. It was also deployed in September 2006 as part of a CALO evaluation effort lasting two weeks with sixteen test users. During this deployment, users have been able to use Towel to manager their to-dos, delegate scheduling and purchasing tasks to CALO, and run many new tasks learned during the data collection period. Preliminary interviews with users have offered some insights into how to improve Towel, such as providing better feedback than the thought balloon and rounding out our to-do list features, and we expect to gain further insight once we are able to interview these participants after the conclusion of the data collection effort.

Conclusion and Future Work

We have described Towel, a user interface for task management that seamlessly incorporates agent-executed tasks with a user's own to-do management and leverages instant-messaging metaphors for communication. We have described how we have modeled to-dos and tasks to enable to the user to easily delegate their work to a software personal assistant and we have shown how the flexibility of the to-list and instant messaging metaphor has allowed for the integration of many AI-based components: task manager, task ontology, task state estimation, calendar scheduling, task learning, concept learning, task explanation, duration predication, and task delegation learning. Although Towel still requires further development, it has already been deployed for use within the CALO project.

We have already mentioned several directions of the future directions for Towel and our task-execution platform, including more specific task duration predications, procedure modification within chats, user-to-user instant messaging, and allowing a broader range of to-do types (e-mail messages, calendar events, semantic objects). There are also several other directions based on current research within the CALO project as well as feedback we have received from users:

- *Sharing knowledge via delegation*: We would like be able to transport any data from the user's knowledge base when delegating a to-do. For example, if Alice wishes for Bob to go over some interview candidates, all the data about those candidates, such as e-mail addresses and resumes, could be transferred to Bob's knowledge base.
- *Understanding to-dos*: Using BEAM to map to-do text into the task ontology has already been useful for task delegation learning and task state estimation, and it makes it easier for users to convert their to-dos into tasks. We hope to enhance this in the future by using BEAM to parse parameters within the free-form text so that they are included when the to-do is delegated to CALO.
- *Task cancellation*: We wish to add a "pause" or "cancel" button as one of our standard chat buttons for tasks. This will first require implementing more robust task-cancellation mechanisms in our Task Manager.
- *Location-based to-do lists*: We are looking into implementing location-aware to-do lists using wireless signal tracking. Some to-dos can only be performed in a particular location, so Towel would be able to hide to-dos that are not relevant to a particular location, provide map-based views of to-dos, or suggest location-specific tasks. We are already providing some location-based reminders, which have been shown to be useful for mobile phones (Sohn *et al*. 2005).

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010. The authors would like to thank Karen Myers, Aaron Spaulding, Gus Prevas, Neil Yorke-Smith, David Morley, Victoria Bellotti, Jim Thornton, Pauline Berry, Bart Peintner, Thomas Lee, Chris Brigham, Shahin Saadati, Tim Rauenbusch, Tim Chklovski, Alyssa Glass, Michael Wolverton, Mei Marker, Josh Levy, Hung Bui, Bill Deans and Cory Albright.

References

Allen, J.; Blaylock, N.; and Ferguson, G. 2002. A Problem Solving Model for Collaborative Agents, In *Proc. of AAMAS'03*.

Bellotti, V.; Dalal, B.; Good, N., Bobrow, D.; and Ducheneaut, N. 2004. What a to-do: studies of task management towards the design of a personal task list manager, In *Proc. of CHI 2004*, 735-742.

Bellotti, V.; Ducheneaut, N.; Howard, M. A.; and Smith, I. E. 2003. Taking email to task: the design and evaluation of a task management centered email tool, In *Proc. of CHI 2003*, 345-352.

Berry, P.; Conley, K.; Gervasio, M.; Peintner, B.; Uribe, T.; Yorke-Smith, N. 2006. Deploying a Personalized Time Management Agent, *Proc. of AAMAS'06*.

Berry, P.; Gervasio, M.; Uribe, T.; Pollack, M.; and Moffitt, M. 2005. A Personalized Time Management Assistant, *AAAI Symposium on Distributed and Schedule Management*.

Blythe, J. 2005. Task Learning by Instruction in Tailor, In *Proc. of the Intl. Conf. on Intelligent User Interfaces*.

Chklovski, T. and Gil, Y. 2005. An Analysis of Knowledge Collected from Volunteer Contributors, In *Proc. of AAAI-05*.

Georgeff, M., Ingrand, F. 1989. Decision-Making in an Embedded Reasoning System, In *Proc. of IJCAI-89*.

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

McGuinness, D.L., Ding, L., Glass, A., Chang, C., Zeng, H., and Furtado, V. 2006. Explanation Interfaces for the Semantic Web: Issues and Models. *Workshop on Semantic Web User Interactions, International Semantic Web Conference*. to appear.

Morley, D., and Myers, K. 2004. The SPARK Agent Framework, In *Proc. of AAMAS'04*.

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

Okamoto, S.; Scerri, P.; Sycara, K. 2006. Toward an Understanding of the Impact of Software Personal Assistants on Human Organizations, In *Proc. of AAMAS'06*.

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

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

Sohn, T.; Li, K.; Lee, G.; Smith, I; Scott, J.; Griswold, W. 2005. Place-Its: A Study of Location-Based Reminders on Mobile Phones, In *Proc. of UbiComp 2005*.