Tracking and Learning Acceptance Critera for Delegated Tasks*

Timothy W. Rauenbusch and Kenneth Conley
Artificial Intelligence Center
SRI International
Menlo Park, CA 94025
rauenbusch,conley@ai.sri.com

Abstract

This paper describes a software system that enables people to both (1) track tasks on their to-do lists and (2) delegate responsibility for performing tasks to others. Combining these two processes provides several benefits, including: (1) automated tracking; and (2) simplified communication and organization of information relevant to a task. On top of these core capabilities, the system supports automatic learning of whether a delegated task should be accepted. A user study investigated whether machine learning methods are effective in predicting whether a delegated task will be accepted. The results indicate that off-the-shelf machine learning techniques can help predict whether a task will be accepted based on earlier examples of task acceptance but that they are prone two types of errors. Task acceptance learning capability may improve performance because the system can take on responsibilities for task management that would otherwise be performed manually.

Introduction

Recent work has investigated systems that assist people in performing their activities in an office, at home, in hospitals, and elsewhere (Varakantham, Maheswaran, & Tambe 2005; Scerri, Pynadath, & Tambe 2002; Fraser & Hauskrecht 2000; Pollack *et al.* 2003). One key function of such systems is the monitoring and allocation of tasks (Varakantham, Maheswaran, & Tambe 2005; Hunsberger & Grosz 2000). While the majority of this work has focused on tasks performed by computer agents, other work has been primarily concerned with tasks performed by people (Bellotti *et al.* 2005; Whittaker, Bellotti, & Gwizdka 2006). Whether tasks are performed by people or software agents, effective assistance software must support (1) tracking tasks and (2) an ability to allocate tasks to those in the best position to perform them. Systems designed for tasks performed by computer agents typically focus on allocation, while systems designed for tasks performed by people have focused on tracking. Therefore, previous systems for task management have generally focused on only one or the other of the two processes.

This paper describes a system that combines the two processes. It has several desirable features:

1. tasks are specified using natural language phrases;
2. additional information relevant to a task can be easily associated with it throughout its life cycle via drag and drop;
3. tasks can be delegated to others and monitored by interested parties after delegation.

In the basic version of the system, when a person receives a request to take on responsibility for a task, she must decide whether to accept or reject that responsibility. This paper describes an enhancement: a learning component that learns to accept or reject tasks on a person's behalf. Okamoto et al (2006) have shown that agent-assisted contingency management could potentially result in large gains in organizational performance and small gains for increased communication speed. We aim to achieve gains both from improved organizational performance and increased communication speed through the learning component. Based on organizational relationships, current workload, task type, and other features, this component learns whether or not a user will accept or reject a task delegated by another person. Based on the learned model, a person can have tasks automatically accepted or rejected by the system, which can reduce interruptions during a period of high workload and improve response times to the person delegating responsibility for the task.

The learning component uses off-the-shelf classification algorithms to determine whether a task request should be accepted or rejected. We analyzed the error rate of the learning component by gathering data during a study that involved sixteen users generating 126 delegation requests. The results indicate that off-the-shelf machine learning techniques can help predict whether a task will be accepted based on earlier examples of task acceptance but that they are prone to two types of errors.

The architecture of the system described in this paper is as follows. Tasks are managed through the Towel to-do list application (Conley & Carpenter 2007). A separate delegation learning agent monitors events from Towel, saves labeled examples of task delegation requests, and learns to predict

*This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010. 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 the Defense Advanced Research Projects Agency (DARPA), or the Department of Interior-National Business Center (DOI-NBC).

whether a delegation request should be accepted. A setting in Towel controls whether the system can automatically accept or reject tasks based on the decisions made by the delegation learning agent.

The contributions of this paper are:

1. a system that combines tracking tasks and allocating responsibility for tasks to other agents in a novel way;
2. analysis of the impact of automatic acceptance of task delegation requests;
3. a study of the effectiveness of machine learning methods for making acceptance and rejection decisions.

This paper is organized as follows. In the next section, we describe the Towel system for tracking and delegating tasks. We then describe the component that learns whether a task delegation request should be accepted, and provide results of a study that tested its effectiveness. Next, we situate our system in the context of related work and then provide conclusions and suggest areas for future work.

Monitoring and Delegating Tasks

The user interface for the monitoring and delegating of tasks is the Towel to-do list application (Conley & Carpenter 2007). The Towel to-do list is a narrow window designed to sit in the peripheral part of a user's screen where it can be easily monitored. To-dos within the Towel are associated with textual reminders: "walk the dog," "groceries," "meeting with Bill." Towel enables the user to organize these to-dos by grouping, tagging, and hiding. It also enables users to perform modifications on individual to-dos such as checking (completing), delegating, setting deadlines, and starring.

The Towel to-do list extends beyond its reminder and organizational role by enabling users to drag resources, such as files and URLs, onto an item in the to-do list (Bellotti *et al.* 2005). These resources allows users to accumulate information about a task, such as where or how a task can be done, or even define the task itself. For example, a to-do labeled "finish paper" might have the paper and citations attached. Instead of searching for the paper when the work needs to be done, the user can use the Towel window as a launch pad for performing and updating the task.

Our system uses a delegative model of task management to augment these task management capabilities and provide means for exchanging tasks directly from the to-do list. A user may choose to delegate a to-do either to another person or to her personal software assistant (Myers & Yorke-Smith 2005). We focus on delegation to people, specifically *teammates* (for example, a coworker working on the same project in an office environment). When a user (the *sender*) wishes to send a delegation request to a teammate, she is given the option of providing a message and additional temporal details about the task (a deadline and duration estimate) that her teammate may find useful. The teammate (the *receiver*) then chooses to accept or reject the request. Changes made to a to-do by the receiver after accepting a delegation request are communicated back to the sender. This feature enables automated tracking of a task: the sender can glance at her to-do list to monitor the progress of a task throughout its life cycle, regardless of who is currently responsible for performing it.

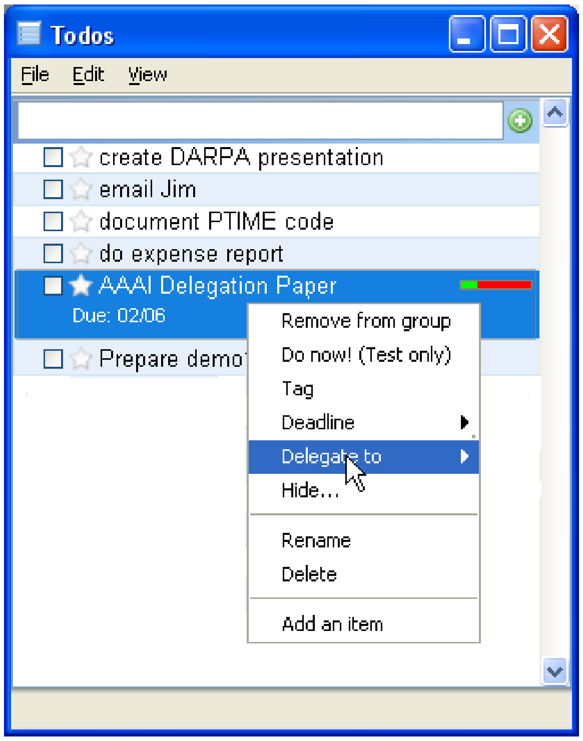


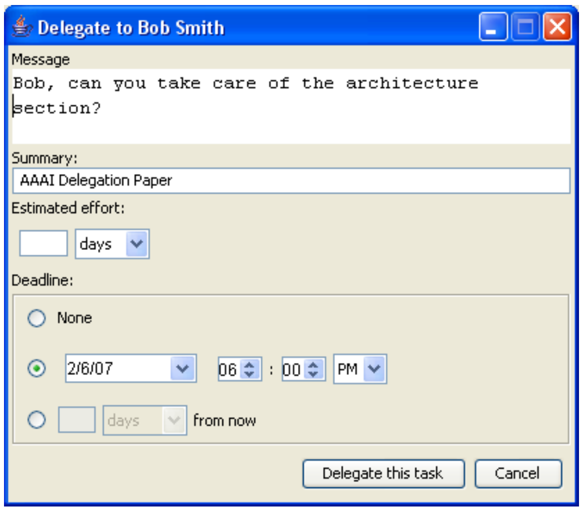
Figure 1: Towel To-do List: Context Menu

Example

The remainder of this section describes the task delegation process using screenshots from Towel. Figure 1 shows Helen Troy's to-do list. The figure shows the context menu that supports delegation. Helen has opened the context menu for the "AAAI Delegation Paper" item, and after she chooses the "Delegate to" menu item, she is presented with a list of people and chooses one.

After choosing Bob Smith as the receiver for the delegation request, Helen is presented with the delegation details form shown in Figure 2. On this form, Helen specifies (1) a message and summary; and (2) temporal information. Both the message and summary are free text fields in which anything might be specified. The summary field is the name used when the task is shown on the to-do list. Two types of temporal information may be specified: estimated effort and deadline. All of the fields on this form are optional; the intent of the form is to allow further specification of task that might be relevant to the receiver. Helen completes the form by clicking on the button labeled "Delegate this task."

Figure 3 shows the accept/reject dialog. This dialog appears on the receiver's screen appears if task acceptance is set to *manual mode*; that is, when the learning component described in the next section is not making the acceptance decisions automatically. The dialog presents the information specified on the delegation details form, and includes



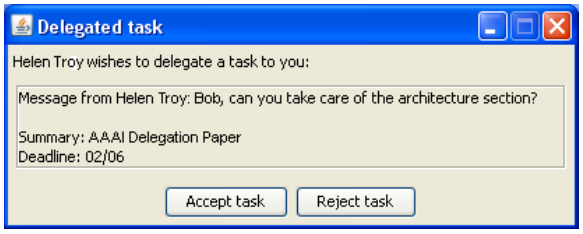
Message
Bob, can you take care of the architecture section?

Summary:
AAAI Delegation Paper

Estimated effort: days

Deadline:
None
[✓ 2/6/07 06 : 00 PM]
days from now

Figure 2: Towel: Delegation Details Form

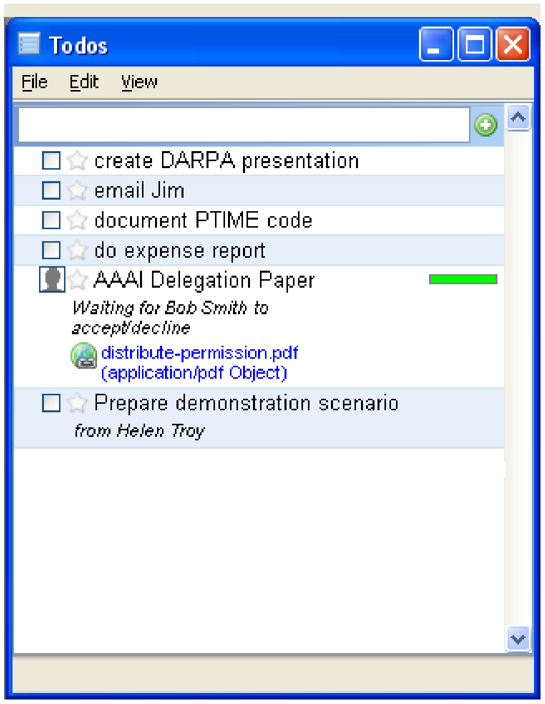


Helen Troy wishes to delegate a task to you:

Message from Helen Troy: Bob, can you take care of the architecture section?

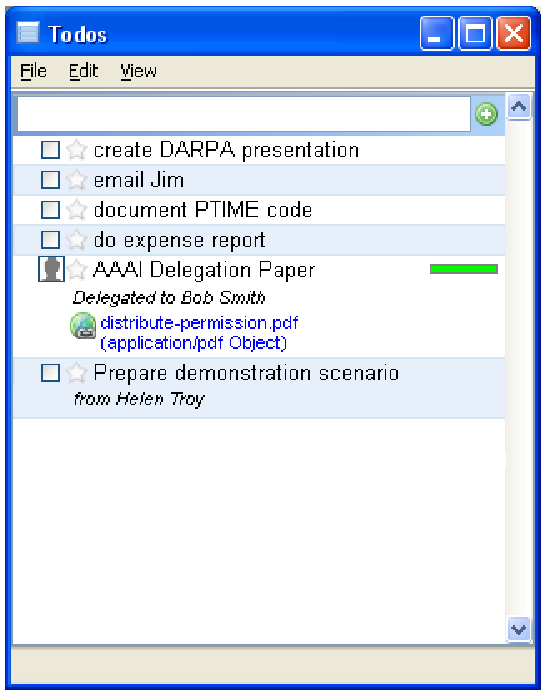
Summary: AAAI Delegation Paper
Deadline: 02/06

Figure 3: Towel (Manual Mode): Acceptance Dialog



- create DARPA presentation
- email Jim
- document PTIME code
- do expense report
- AAAI Delegation Paper
  Waiting for Bob Smith to accept/decline
  distribute-permission.pdf (application/pdf Object)
- Prepare demonstration scenario
  from Helen Troy

Figure 4: Towel To-do List: Waiting for Decision



- create DARPA presentation
- email Jim
- document PTIME code
- do expense report
- AAAI Delegation Paper
  Delegated to Bob Smith
  distribute-permission.pdf (application/pdf Object)
- Prepare demonstration scenario
  from Helen Troy

Figure 5: Towel To-do List: Delegated

the name of the sender of the request. The receiver of the task request may accept or reject the task by selecting the appropriate button. (If task acceptance was set to *automatic mode*, no dialog would appear, and the decision would be made immediately according to the classifier described in the next section.)

When task acceptance is in manual mode and the receiver has not yet chosen to accept or reject the request, the sender's to-do list displays the phrase "Waiting for [receiver] to accept/decline" (Figure 4).

If the receiver chooses to accept the request, the item is automatically placed on his to-do list. The item remains on the sender's to-do list and is updated automatically when any changes to its status are made (Figure 5). If the receiver rejects the request, a dialog notifies the sender, and the phrase "Waiting for [receiver] to accept/decline" is removed from the item. To attempt to alleviate awkwardness that may arise as a result of a task rejection, when the system is in automatic mode, the sender is notified that the decision was made automatically.

Task Acceptance Learning

The learning component of the system automatically responds to task delegation requests when the system is in automatic mode. The problem addressed by the component is that of *task acceptance learning*, which is to automatically determine whether a task should be classified as accepted or rejected, given a set of labeled training examples. A labeled set of training examples consists of a set features that describe each task delegation request, along with a label to indicate whether the request was accepted or rejected.

	Task accepted	Task rejected
Task should have been accepted	Correct	**Under-commitment**
Task should have been rejected	**Over-commitment**	Correct

Table 1: Errors in acceptance learning

There are advantages and disadvantages to pursuing automatic acceptance or rejection of task delegation requests. The main advantages are time savings for both the sender and receiver of the request. For the receiver, automatic classification means that she need not spend time considering and responding to each request she receives; her time can be spent on more important activities. For the sender, automatic classification means immediate feedback. The lag time between sending a request and receiving a response is eliminated. This decreased lag time is especially useful if the request is rejected—a decision can be made immediately about how to proceed.

The main disadvantages of automatic classification are classification errors. Table 1 plots the two types of errors: undercommitment and overcommitment. *Undercommitment* occurs when a task delegation request that should have been accepted is rejected. The costs of undercommitment is the extra time that the sender incurs in finding an alternative way to proceed (for example, delegate to someone else, abandon the task, perform it herself, or try to resend the delegation request). *Overcommitment* occurs when a task delegation request that should have been rejected is accepted. The receiver of a task that is accepted in error may not have the knowledge or resources required to perform it effectively.

Both types of errors introduce unnecessary costs and neither type of error lends itself to domain- and task-independent quantification. For example, erroneously rejecting a request from a manager for the task of preparing a financial report may have a higher cost than erroneously rejecting a peer's request to bring food to the potluck. However, quantifying the costs of each error may be difficult. For now, we are concerned with studying the error rates of task acceptance learning, but do not explicitly quantify the costs of the errors or the benefits of automatic acceptance of task delegation requests.

Method

To test the efficacy of automated machine learning of task acceptance, we conducted a study in which we collected examples of task delegation requests along with a labels of whether the requests were accepted or rejected. We then performed a ten-fold cross-validation of two off-the-shelf machine learning algorithms on the data.

We identified three main categories of factors that people use to determine whether they will accept or reject a task delegation request:

Relationship — the relationship between the sender and the receiver of the request (for example, whether they are peers, or whether one is in a position of authority over the other)

Context of Receiver — factors that affect the ability of the receiver to take on new commitments (for example, the number of current commitments on the receiver's schedule)

Task Specification — features that describe the task (for example, deadlines or expected difficulty).

Category	Feature
Relationship	managerOf managementHierarchy
Context of Receiver	timeAvailable shiftTasksCost
Task Specification	deadlineDistance latestStartTimeDistance taskType

Table 2: Features used to describe examples

We collected task acceptance and rejection data as part of a six-day study in which sixteen participants interacted with a personal assistance software application while doing real or simulated office work. Users were given instructions on how to delegate tasks and were asked to use the delegation component of the system. However, the particular to-do items, and the acceptance decisions were left up to the participants and were not specified in advance.

During the study, we collected labeled data for a total of 126 task delegation requests. Of these 126 requests, 97 were accepted and 29 were rejected.

Table 2 summarizes several features of each task delegation request that was recorded along with a notation of whether the task was accepted or rejected. The managerOf and managementHierarchy features aim to describe the relationship between the sender and receiver of the task delegation request. managerOf is a boolean value that is true if and only if the sender of the task is a manager of the receiver. managementHierarchy is an integer that enumerates the distance between the sender and receiver in an organizational hierarchy.

The features timeAvailable and shiftTasksCost aim to describe the context of the task delegation request receiver and are a function of the receiver's schedule and the expected duration and deadline given by the task delegation request. timeAvailable is a boolean value that is true if and only if the user has sufficient time in its current schedule to accommodate the task's expected duration by the deadline. shiftTasksCost is an integer that enumerates the receiver's current commitments that would need to be abandoned or moved if the requested task's expected duration was to be incorporated into the schedule by the given deadline.

The features deadlineDistance, latestStartTimeDistance, and taskType aim to describe features of the task in the request. deadlineDistance denotes the total number of minutes between the deadline and the time the task delegation request was sent. latestStartTimeDistance is the deadlineDistance minus the expected duration given in the task delega-

Classifier	Undercommitment	Overcommitment
Naïve Bayes	89 (71%)	4 (3%)
Decision Tree	0 (0%)	29 (23%)

Table 3: Error rate in classification

tion request. taskType is an approximation of the sort of task in the request, as computed from the natural-language task description. We used the BEAM system (Chklovski & Gil 2005; Gil & Chklovski 2007) to provide an estimate of the task type. In the initial experiment, we supported task types "Scheduling," "Reviewing," and "Other." The BEAM system intelligently relates natural-language to-do descriptions to types. For example, BEAM categorizes the tasks "Agree on meeting time" and "Schedule meeting with Paul" as "Scheduling" tasks. The three initial task types were selected because the system's initial test domain was hiring a job candidate that involved reviewing resumes and scheduling interviews.

For many of the training examples, not all of the features were specified. For example, if a sender did not specify a deadline for a task, deadlineDistance and latestStartTimeDistance cannot be computed. In such cases, a default value was used as the value for the feature in the example.

Two standard machine learning algorithms were applied to the data: (1) naïve Bayes classifier and (2) C4.5 (J48) decision tree classifier. A cross-validation study was used to generate error rates. For each algorithm, the 126 data points was divided into ten folds. Nine of the folds were used as learning examples, and one was held out to test the data. The process was repeated ten times, so that each fold was held out for testing once. The two algorithms were chosen because they are relatively well-known and access to their implementations was readily available. Our goal was to test the feasibility of machine learning techniques for task acceptance—not to determine which learning technique were best for the problem.

Results

Table 3 shows the error rates for learning on each of the two classifiers. The results show that decision tree learning was more effective in minimizing undercommitment, while naïve Bayes learning was more effective in minimizing overcommitment. The naïve Bayes classifier is conservative with respect to accepting a task request. Therefore, it minimizes its overcommitment rate (3%), while incurring a relatively high rate of undercommitment: it labeled as rejected 89 of the 97 tasks that were actually accepted.

The decision tree classifier is aggressive in accepting tasks. It never rejected a task that should have been accepted. However, it accepted all 29 tasks that should have been rejected.

Both the naïve Bayes and the decision tree classifiers made a significant number of errors. With the data generated from our study, it would appear that the naïve Bayes classifier is preferred in domains in which the cost of overcommitment is high, and the decision tree classifier is preferred in domains in which the cost of undercommitment is high. However, while the results of this study indicate that machine learning may be useful for automatic acceptance of tasks, further study is required before recommendations should be made.

Related Work

As mentioned earlier, recent work has investigated systems that assist people in performing their activities in a variety of settings (Varakantham, Maheswaran, & Tambe 2005; Scerri, Pynadath, & Tambe 2002; Fraser & Hauskrecht 2000; Pollack *et al.* 2003). Personal assistant software systems have been used to address problems in decision support (Li, Giampapa, & Sycara 2006), scheduling and coordination (Emami *et al.* 2006; Berry *et al.* 2006; Weber & Pollack 2007) and monitoring task progress (Chalupsky *et al.* 2001). The to-do manager and delegation learning capability presented in this paper aim to provide support for coordination and progress monitoring in a flexible way. The system provides communication and data entry infrastructure. It can learn to make task acceptance decisions on a person's behalf but can also operate in a manual mode, where the person remains the decision maker.

Okamoto et al (2006) aim to quantify the impact of personal assistant software on productivity in organizations. Their results indicate that such software can improve performance, that the improvement is typically not dramatic, and that the areas in which such software is most useful is not well understood. Our work aims to understand the effectiveness of learning for task acceptance but we have not performed a systematic evaluation of the impact of our assistance system on personal or group efficiency.

For many people, email is the primary application for task management (Bellotti *et al.* 2005). Towel's to-do-list-based approach provides additional capabilities that we believe make it more desirable than email for certain types of tasks: users get immediate feedback on the status of a task, information about the task like deadlines are explicitly modeled, and it simplifies task delegation by allowing the user to pass along information about the task that has been accumulated.

Conclusion and Future Work

We described a personal assistance software system that combines the Towel to-do list management system that enables a person to delegate a task to others and a component for automatic acceptance learning. There were several advantages for the receiver and sender to using the system to automatically accept or reject delegation requests. On the other hand, automatic acceptance and rejection risks incurring costs of overcommitment and undercommitment errors. We conducted a study in which people generated 126 task delegation requests. We applied two off-the-shelf classifiers to the data and generated error-rate predictions using the data. While the results indicate that naive Bayes and decision tree classifiers may be effective in making acceptance decisions of a user's behalf, significant error rates occurred.

In the future, we hope to decrease the rate of overcommitment and undercommitment. We plan to experiment with performing learning on a per-user basis rather than across all users. To do an effective study, we need to collect more data from each user. The results of the experiments on the learning were based on analysis of the 126 data points generated by all sixteen participants in the study. Using training data gathered from a collection of users has the advantage of more training examples but the drawback of possible overgeneralization. We plan to run another study in which we collect more data per participant to investigate whether improvements in error rates can be attained by enabling the system to learn a model of acceptance based on a single user's training examples.

In addition, we plan to test the effectiveness of additional features of task delegation requests. For example, the relationship features managerOf and managementHierarchy are currently only applicable to persons in the same hierarchical organization and requires that the organizational tree be known in advance. We plan to investigate other, more ad hoc, relationship features such as the difference in each person's job title in a hierarchy of job titles generally to approximate seniority or reporting relationships that might affect delegation request acceptance.

We also plan to extend the behavior of the learning component beyond simple acceptance or rejection. For example, we hope to study a system that learns *which* teammate would be a good candidate for delegation of a particular task. We propose to use performance metrics on prior tasks (possibly generated automatically from deadline and actual completion times) in the training.

References

Bellotti, V.; Ducheneaut, N.; Howard, M. A.; Smith, I. E.; and Grinter, R. E. 2005. Quality versus quantity: e-mail-centric task management and its relation with overload. *Human-Computer Interaction* 20(1–2):89–138.

Berry, P.; Peintner, B.; Conley, K.; Gervasio, M.; Uribe, T.; and Yorke-Smith, N. 2006. Deploying a personalized time management agent. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM Press.

Chalupsky, H.; Gil, Y.; Knoblock, C.; Lerman, K.; Oh, J.; Pynadath, D.; Russ, T.; and Tambe, M. 2001. Electric elves: Applying agent technology to support human organizations. In *IAAI01*.

Chklovski, T., and Gil, Y. 2005. An analysis of knowledge collected from volunteer contributors. In *Proc. of AAAI05*.

Conley, K., and Carpenter, J. 2007. Towel: Towards an intelligent to-do list. In *AAAI Spring Symposium on Interaction Challenges for Intelligent Assistants*.

Emami, G.; Cheng, J.; Cornwell, D.; Feldhousen, M.; Long, C.; Malhotra, V.; Starnes, I.; Kerschberg, L.; Brodsky, A.; and Zhang, X. 2006. Active: agile coordinator testbed integrated virtual environment. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 1580–1587. New York, NY, USA: ACM Press.

Fraser, H., and Hauskrecht, M. 2000. Planning treatment of ischemic heart disease with partially observable markov decision processes. *AI in Medicine* 18:221–244.

Gil, Y., and Chklovski, T. 2007. Enhancing interaction with to-do lists using artificial assistants. In *AAAI Spring Symposium on Interaction Challenges for Intelligent Assistants*.

Hunsberger, L., and Grosz, B. J. 2000. A combinatorial auction for collaborative planning. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000)*.

Li, C.; Giampapa, J.; and Sycara, K. 2006. A review of research literature on bilateral negotiations. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Special Issue on Game-theoretic Analysis and Stochastic Simulation of Negotiation Agents* 36(1).

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.; and Sycara, K. 2006. Towards an understanding of the impact of software personal assistants on human organizations. In *AAMAS*.

Pollack, M. E.; Brown, L.; Colbry, D.; McCarthy, C. E.; Orosz, C.; Peintner, B.; Ramakrishnan, S.; and Tsamardinos, I. 2003. Autominder: An intelligent cognitive orthotic system for people with memory impairment. *Robotics and Autonomous Systems* 44:273–282.

Scerri, P.; Pynadath, D.; and Tambe, M. 2002. Towards adjustable autonomy for the real-world. *JAIR* 17:171–228.

Varakantham, P.; Maheswaran, R.; and Tambe, M. 2005. Exploiting belief bounds: Practical pomdps for personal assistant agents. In *AAMAS*.

Weber, J. S., and Pollack, M. E. 2007. Entropy-driven online active learning for interactive calendar management. In *International Conference on Intelligent User Interfaces*.

Whittaker, S.; Bellotti, V.; and Gwizdka, J. 2006. Email as personal information management. *Communications of the ACM* 49(1):68–73.