Carnegie Mellon University
Research Showcase

Institute for Software Research | School of Computer Science

1-1-2009

Design and Evaluation of User Interfaces for the RADAR Learning Personal Assistant

Andrew Faulring

Ken Mohnkern

Aaron Steinfeld

Brad Myers
Carnegie Mellon University, bam@cs.cmu.edu

Follow this and additional works at: http://repository.cmu.edu/isr

Recommended Citation
Faulring, Andrew; Mohnkern, Ken; Steinfeld, Aaron; and Myers, Brad, "Design and Evaluation of User Interfaces for the RADAR Learning Personal Assistant" (2009). *Institute for Software Research*. Paper 737.
http://repository.cmu.edu/isr/737

This Article is brought to you for free and open access by the School of Computer Science at Research Showcase. It has been accepted for inclusion in Institute for Software Research by an authorized administrator of Research Showcase. For more information, please contact research-showcase@andrew.cmu.edu.

The Design and Evaluation of User Interfaces for the RADAR Learning Personal Assistant

Andrew Faulring, Ken Mohnkern, Aaron Steinfeld, and Brad A. Myers

■ *The RADAR project developed a large multiagent system with a mixed-initiative user interface designed to help office workers cope with e-mail overload. Most RADAR agents observe experts performing tasks and then assist other users who are performing similar tasks. The interaction design for RADAR focused on developing user interfaces that allowed the intelligent functionality to improve the user's workflow without frustrating the user when the system's suggestions were either unhelpful or simply incorrect. For example, with regard to autonomy, the RADAR agents were allowed much flexibility in selecting ways to assist the user but were restricted from taking actions that would be visible to other people. This policy ensured that the user remained in control and mitigated the negative effects of mistakes. A large evaluation of RADAR demonstrated that novice users confronted with an e-mail overload test performed significantly better, achieving a 37 percent better overall score when assisted by RADAR. The evaluation showed that AI technologies can help users accomplish their goals.*

E-mail plays a central role in many people's work. Unfortunately e-mail client software is poorly suited to support the "collaborative quality of e-mail task and project management," which results in people suffering from "e-mail overload" (Bellotti et al. 2005). The e-mail overload occurs in part because e-mail client software does not support efficient strategies for tracking and processing all the tasks contained within a person's e-mails. For example, switching from one task type to another can incur a significant overhead. Many coping strategies, such as asking that people use a specific subject line for each task type, do not scale well and simply shift the burden to others.

It is difficult to create an efficient order when sorting one's inbox using e-mail-centric properties, such as sender, subject, or date. In general, the resulting order will not group similar tasks together nor will it account for intertask dependencies. Several research projects have experimented with adding task-management features to e-mail clients (Bellotti et al. 2005; Gwizdka 2002; Whittaker, Bellotti, and Gwizdka 2006), and some e-mail clients do provide features that try to facilitate task management such as tagging and separate to-do lists. The primary drawback of this approach is that users resist doing the required additional work to use them (Whittaker and Sidner 1996), and it forces them to read each e-mail at least twice: once to create the task, then again when actually doing the task.[1]

We developed a mixed-initiative e-mail system, modeled after a personalized assistant, which uses AI learning techniques to reduce e-mail overload. RADAR (reflective agents with distributed adaptive reasoning) was a five-year interdisciplinary proj-

Copyright © 2009, Association for the Advancement of Artificial Intelligence. All rights reserved. ISSN 0738-4602

ect involving more than 100 people, which built a suite of intelligent agents to help office workers complete routine tasks more efficiently. RADAR's AI technologies supported a more efficient task-centric workflow as compared with the traditional e-mail-centric workflow.

This article discusses usability issues encountered while designing and evaluating RADAR. It focuses on providing concrete examples of many of the ideas discussed in the theme articles in this special issue, rather than providing a complete description of RADAR. Freed provides a more detailed description of the RADAR approach, architecture, and agents (Freed et al. 2008). Steinfeld describes the extensive evaluation used to measure how well RADAR succeeded in reducing e-mail overload (Steinfeld et al. 2007).

A key design decision for RADAR was that its agents would generally not be allowed to take autonomous action when such actions would be visible to other people or systems. Instead, RADAR assisted a user by completing parts of a task and then would ask the user for approval when the next step would be externally visible. This approach gave RADAR considerable flexibility to work on the user's behalf without risking costly mistakes. In a real-world situation, such mistakes could embarrass the user or leak private information, which are concerns that might negatively affect adoption and trust. The user remains in control, has an understanding of what the system will do, and maintains the desired amount of privacy, all challenges that are discussed in the usability side-effects theme article (Jameson 2009) in this issue.

The RADAR architecture consists of three groups of agents, which together help the user to handle their incoming e-mails (see figure 1). First, the message-task linking (METAL) agents identify the tasks contained within each e-mail (b). The action list, the primary user interface for METAL, provides a task-centric view of the user's inbox (see figure 2). Second, the multitask coordination assistant (MCA) agents suggest an order for handling the tasks (c). The progress bar displays a suggested schedule along with a visualization of the user's completed work (see figure 3). Third, the PowerTools agents provide assistance with performing each task (d). Each PowerTools agent supports a particular kind of task. PowerTools agents include the briefing assistant (Kumar, Das, and Rudnicky 2007), schedule optimizer (Fink et al. 2006), and webmaster (Zimmerman et al. 2007). This article focuses on usability issues for the METAL and MCA agents; the PowerTools agents were designed using the same process.

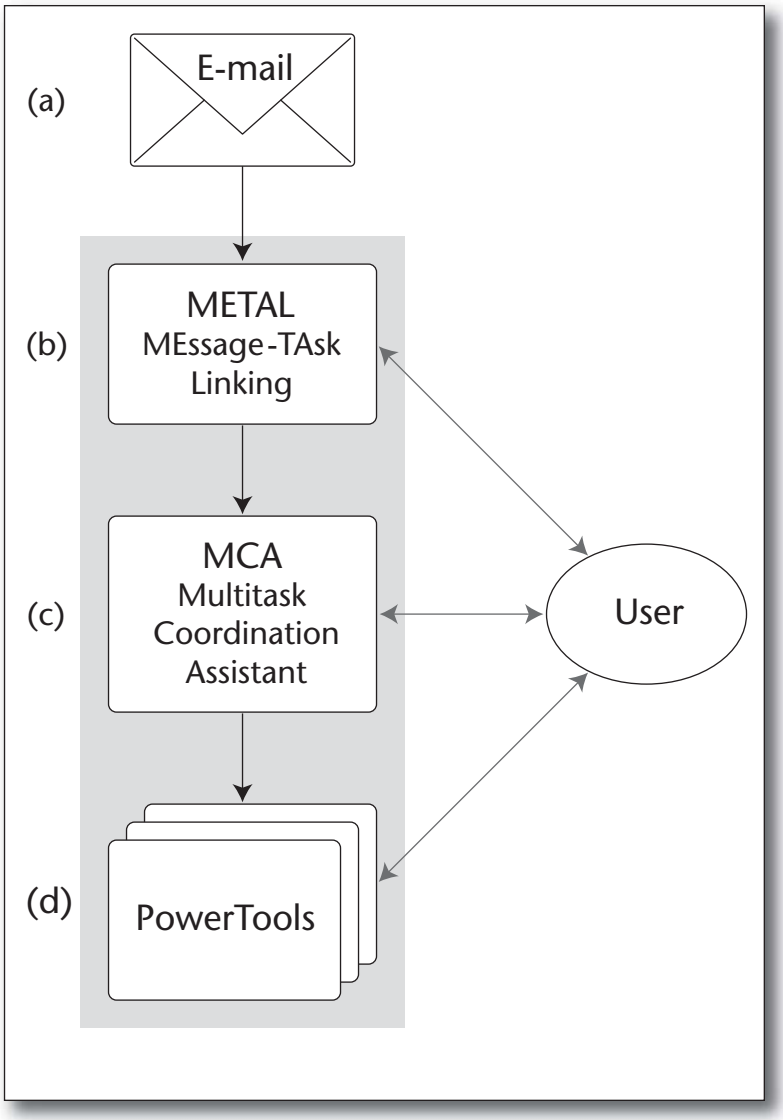

Figure 1. The RADAR Architecture.

The architecture consists of three groups of agents (b, c, and d), which together help users to handle their incoming e-mails (a).

Evaluation Challenges

A key challenge was designing an effective evaluation that measured how well the system reduced e-mail overload. We wanted to conduct carefully controlled, repeatable user studies to measure how well RADAR helped people. Since the project was planned to run for five years, repeatability was an important requirement that would allow us to measure progress over time. We initially considered designing an evaluation that used people's actual e-mail, but eventually decided against such a design. Such an evaluation would make comparisons difficult because of the vast differences among people's workloads. Additionally, any study would have to carefully protect the privacy of both

(a) Incomplete Actions (11)

Order ▼	Description	Subject	Sender	Created	Modified	Creator
1	Modify Event: *Demo M1: Driver Monitoring Systems*	Attendance figures and new #	Amy Lim <lim12@ardra.org>	Today, 3:32 PM		RADAR
2	Modify Event	note schedule chagnes	Spence Pierro <spierro@ardra.org>	Today, 3:54 PM		RADAR
3	Modify Room: *Flagstaff: Sternwheeler*	Sternwheeler Capacity	Meredith Lorenz <lorenze@pittsburgh.flagstaff.com>	Today, 4:07 PM		RADAR
4	Modify Room: *Flagstaff: Vandergrift*	Sternwheeler Capacity	Meredith Lorenz <lorenze@pittsburgh.flagstaff.com>	Today, 4:10 PM		USER
5	Optimize the Schedule	*no email*		Today, 3:45 PM		RADAR
6	Website Update (VIO): Modify Person: *Austin Parton*	Webpage	Austin Parton <aparton@ardra.org>	Today, 3:37 PM		RADAR
7	Website Update (VIO): Modify Person	Attendance figures and new #	Amy Lim <lim12@ardra.org>	Today, 3:32 PM		RADAR
8	Website Update (VIO)	Organization Wrong	Sonal Malhotra <smalh@ardra.org>	Today, 4:32 PM		RADAR
9	Website Update (WbE)	change phone numbers	Emily Halwizer <halwizer@ardra.org>	Today, 4:47 PM		RADAR
10	Place a Vendor Order	Tech. Request - flip charts	Maggie Foxenreiter <mfox@ardra.org>	Today, 3:33 PM		RADAR
11	Send a Briefing	Brief me, please	Jonathon Robertson <jrobertson@ardra.org>	Today, 4:42 PM		RADAR

(a.i) (a.ii) (a.iii) (a.iv)

(b) Overflow Actions (1)

Order	Description ▼	Subject	Sender	Created	Modified	Creator
	Reply to Question	Vegetarian options?	Sandra Nubanks <snubanks@ardra.org>	Today, 4:02 PM		RADAR

(c) Completed Actions (1)

Order	Description	Subject	Sender	Created	Modified ▼	Creator
	Modify Event: *Workshop 1a: Intermodal Passenger Screening*	Attendance figures	Amy Lim <lim12@ardra.org>	Today, 3:21 PM	Today, 3:45 PM	RADAR

(d) Deleted Actions (1)

Order	Description	Subject	Sender	Created	Modified ▼	Creator
	Modify Speaker's Availability	Planning for History Week	Michelle Randal <mich-randal@gmail.com>	Today, 4:28 PM	Today, 4:34 PM	RADAR

(e) Possibly Conference-Related Emails (1)

Read	Subject	Sender	Date ▼	
•	for my presentation	Laura Timdale <laurat2@ardra.org>	Today, 3:24 PM	Add an Action

Blake, I didnt know who to contact about making sure to have a laptop available, and connected to teh AV equipment - ie projector. I want all that ready on the ...

(f) Other Emails (1)

Read	Subject	Sender	Date ▼	
•	car arrangements	Angie Randal <angiednacer6@gmail.com>	Today, 3:23 PM	Add an Action

Ms K is counting on me to help out with the kids' dance class. The car is still in the shp. Can you drop me off over there? thanks :-)

(g) Deleted Emails (1)

Read	Subject	Sender	Date ▼	
	Precipitation Update	Weather Alerts <weather@weather.gov>	Today, 3:56 PM	Add an Action

There is a 70% probability for thunderstorms with heavy rain in ALLEGHENY COUNTY this evening through tomorrow. Plan accordingly and be safe! Go to www.weather.gov ...

Figure 2. The RADAR Action List.

The action list provides a task-centric view of an e-mail inbox. The "Incomplete Actions" (a), "Overflow Actions" (b), and "Completed Actions" (c) tables list the tasks contained within e-mail messages, allowing the user to sort by task-centric properties. The three e-mail tables contain e-mails for which no tasks have been created (e, f, and g).

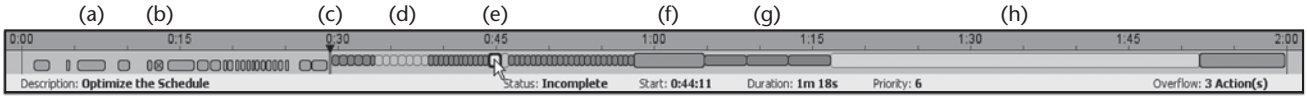

Figure 3. The RADAR Progress Bar.

The progress bar shows completed (a) and deleted (b) tasks to the left of the current time (c), and the suggested schedule to the right. Non-critical tasks are blue (a, b, and g), critical tasks are orange (f), and expected tasks are gray (d and h). Details about the highlighted task (e) are shown in the status bar at the bottom. The progress bar appears in its own window, which is always visible.

the participants and those with whom they communicated by e-mail.

We still wanted to create the most realistic, generalized evaluation possible. Project members, in conjunction with external evaluators, spent significant effort developing a rich systemwide user test to evaluate how well RADAR assists a user (Steinfeld et al. 2007).[2] The evaluation presented users with a simulated conference-planning scenario in which each user is confronted with a challenging e-mail overload workload. First, we required that dependencies exist between some tasks, and doing those tasks out of order should result in wasted work or incorrect results. Second, e-mail-centric properties such as subject lines should not be very helpful for grouping similar tasks or discerning effi-

cient task orders. Third, the test should be hard and users should not be expected to finish within the allotted time, so as to eliminate ceiling effects from performance results.

In the test, users assumed the role of a conference planner, filling in for the regular planner, Blake, who is indisposed. The simulated four-day, multitrack conference had keynotes, plenary talks, banquets, paper sessions, workshops, and so forth. Participants needed to handle the outstanding conference planning tasks that had arrived in e-mail, including many requests from the conference attendees. Blake's inbox contained these e-mails, which can be categorized as follows. Scheduling e-mails told the organizer about updated event constraints or room properties, which determine how well each room satisfies an event's requirements. Website e-mails contained requests from attendees to correct their contact information on the conference website. Informational e-mails contained questions about the conference, especially the schedule, from attendees. Vendor e-mails specified attendee meal preferences and A/V requirements. The briefing e-mail from the conference chair requested a summary of the user's progress, which was due at the end of the test. Participants also had to deal with a conference crisis, which involved the loss of use of a significant number of rooms in which conference events had already been scheduled.

Over three years, we ran four major studies using this test to evaluate RADAR's ability to assist a user. Each study used a unique set of e-mails that followed the pattern described above but varied so that the agent developers would not have seen the exact e-mails prior to each test. Each e-mail set contained about 120 e-mails. Approximately one third of those messages were unrelated to the conference, serving as "noise" messages.

Each test session lasted up to 4.25 hours and included up to 15 users, who worked independently on the same e-mail set. With such long test sessions, the ability to test multiple users concurrently allowed us to run many tests on different versions of the system. In the first phase of each test session, users learned about the conference-planning scenario and participated in hands-on training with the software. The hands-on training ensured that all users would be familiar with the software that they would use. During training, RADAR did not provide expert advice; instead users were given examples of the types of advice that they would receive. Following a break, users started the two-hour testing session, which included another break after one hour.

Over the course of three years, approximately 400 people participated in the four major evaluations; 300 more people participated in additional interim evaluations. The test scenario created a realistic environment in which to evaluate how a personalized assistant can help overcome e-mail overload. The results of the repeated evaluation of RADAR with novice users gave confidence that its technologies help reduce e-mail overload.

Design and Evaluation Methods for Improving Usability

Common user interface design techniques worked well when designing RADAR's user interfaces. For example, early in the project some RADAR researchers performed a type of field study called a "contextual inquiry" (Beyer and Holtzblatt 1998). The researchers observed office workers who regularly receive e-mail requests that are performed by filling out forms. The observed work was very helpful in informing the design of the RADAR system because it helped us understand actual user needs.[3] These workers would regularly save a small set of related tasks and then perform them together as a group. They claimed that this saved them time by reducing the number of times that they needed to connect to different IT systems. Each connection entailed overhead costs of navigating to different interfaces and logging in to the individual systems. These observations inspired us to focus on creating a task-centric workflow for processing e-mail requests, which fit these users' desired work practices better than the traditional e-mail-centric workflow. In conjunction with the underlying AI technologies, which we will discuss later, we were able to create a system that matched users' desired work practices (see the usability benefits theme article [Lieberman 2009] in this issue).

We employed an iterative design process, another commonly user interface design technique (also discussed in the usability engineering methods theme article [Spaulding and Weber 2009] in this issue). Early in the process we built paper prototypes of the screen layouts, which we showed to potential users. We could construct paper prototypes, allowing us to iterate over many design ideas and thereby rapidly improve the user interface. One important discovery from the usability testing was that the term *action* works better than *task*, so we used the former term in the user interface. However, we will continue to use *task* throughout this article except when referring to specific user interface elements.

We held weekly development meetings attended by both agent developers and user interface designers. These meetings increased team integration and facilitated an ongoing discussion about possible ways in which AI technologies might be employed to assist users through usable interfaces.

Prior to major tests, we ran multiple practice tests over the course of several months. These prac-

tice tests, which employed both RADAR developers and novice users as testers, revealed many problems. For some of the tests, we used RADAR developers, who were effectively novice users with respect to the other system components. These people saw firsthand how developers can have difficulty seeing usability problems with their own component, and hence understood the usefulness of having "fresh eyes" test a system. It made them more aware of the challenges of designing user interfaces for other people and more open to suggested improvements to their components based upon usability testing.[4] Usability testing allowed us to find and fix problems, which significantly increased the usability of the overall system.

A Task-Centric Interface for E-Mail Processing

The e-mail classifier, a METAL agent, examines the content of each e-mail for evidence that it contains any of the eight task types that it was trained to recognize (Yang et al. 2005). The e-mail classifier uses a regularized logistic regression algorithm that scales to thousands of labels (Yang, Zhang, and Kisiel 2003). When it finds sufficient evidence for a given task type, it labels the e-mail with the task type. The classifier evaluates the evidence that supports each task type independently, and so it can apply zero, one, or more different task labels to each e-mail. However, it cannot determine whether an e-mail contains multiple tasks of the same type. To improve classification performance, Scone (Fahlman 2006), which is RADAR's knowledge base, provides additional ontological information that is not contained within an e-mail's content. Examples of this information include basic facts, "the Connan room is in the University Center," and higher-level concepts, "a peanut is a kind of food that people might be allergic to."[5]

The novice users in the RADAR evaluation might have had difficulty effectively judging whether the e-mail classifier's labels were correct, because these users were not experts with respect to the different tasks. We were concerned that too many false positives might have confused them, causing them to waste time.[6] So we tuned the classifier to favor precision over recall. An examination of the classifier's behavior showed that it did perform as desired. The most recent evaluation used an e-mail set with 123 e-mails, which contained 102 task labels.[7] The classifier correctly found 47 task labels and incorrectly suggested 6 other task labels (false positives): precision = 0.887 (47/53) and recall = 0.461 (47/102). In a longer running test or a real work environment in which the users are less likely to be misled by errors, it may be desirable to adjust the precision/recall balance. In such cases, increasing the recall, which may lower the precision, may lead to better user performance, since experts may be able to quickly discard false positives. Even so, too many false positives may undermine a user's confidence in the system.

For each task label applied to an e-mail, RADAR creates a task object, which is managed by the task manager database (Garlan and Schmerl 2007). Information stored includes whether there is a web form for handling that kind of task. Early in the system design, contextual inquiries had revealed that many tasks required filling out web-based forms. RADAR's natural language processor (Nyberg et al. 2008) attempts to specify task-specific parameters in the form, including the database record that the form should modify. If RADAR can identify the record, then it will also try to fill in the fields of the form.

The resulting tasks are displayed in the action list, which provides a task-centric view of the user's e-mail inbox (see figure 2). The action list design provides a mixed-initiative interaction style for creating and completing tasks contained within e-mails. The action list allows a user to inspect the tasks that RADAR created, add ones that were missed, delete ones that should not have been created, and launch web pages to perform some of the tasks. This novel integration of a to-do list with the forms for completing the tasks removes unnecessary steps from the process of performing a task.

The action list contains seven tables divided into two groups: the first for tasks, and the second for e-mails (see figure 2). The task group contains four tables that list "incomplete" (a), "overflow" (b), "completed" (c), and "deleted" (d) tasks. Tasks that the user has yet to perform are split between the incomplete and overflow tables, with the latter table containing the tasks that the multitask coordination assistant recommends that the user should skip due to time constraints. An e-mail message can be associated with multiple tasks of the same type (see a.ii and a.iii in figure 2), multiple tasks of different types (a.i and a.iv), or both. Note that in these examples, the same e-mail appears multiple times in the action list. Tasks completed by the user appear in the completed table, which provides users with a record of their progress and allows them to go back and revisit previous tasks, for example to do another task in the same way. The deleted table is intended to contain tasks that RADAR created erroneously and the user subsequently deleted.

The tabular task display allows users to sort their tasks with respect to task-centric properties such as "description" or "order" (by clicking the appropriate column title), in addition to standard e-mail-centric properties such as sender, subject, and date. When an e-mail contains multiple tasks of different types, each of those types will be grouped with

the other tasks of the same type when the user sorts the table by the "description" column. The other columns in each table use a standard sort order based upon their data type.

The second set of tables (e, f, and g) display e-mails that are not associated with any nondeleted tasks. These tables provide similar features to a traditional e-mail inbox, such as columns for the subject, sender, and date. The entry for each e-mail also includes an excerpt from the beginning of the e-mail body to aid the user in determining whether an e-mail contains a task without requiring the user to open the e-mail. Clicking either the subject text or the "Add an Action" link navigates to a page that displays the standard header and body sections along with the list of tasks that the user can add to the e-mail.

The e-mails are divided among the three tables. The "possibly conference-related e-mails" table contains e-mails that RADAR thinks may contain tasks but could not confidently identify the exact task type (e). This partial classification focuses the user's attention on e-mails likely to contain tasks without risking errors that might result if RADAR incorrectly classified the task as being of a particular type. The second table contains other e-mails that RADAR did not identify as task-related (f). The third table contains e-mails that the user deleted (g).[8]

The Multitask Coordination Assistant: Providing Task Ordering Advice

METAL's ability to automatically locate tasks within e-mails helps the user at the individual task or e-mail level. However, the user still has to select an order in which to work on the tasks. A novice user will likely lack metaknowledge about tasks that would be helpful in picking an efficient order. Useful metaknowledge includes relative task importance, expected task duration, task ordering dependencies, and which new tasks may become necessary after completing a task. An expert user with that knowledge should be able to select an efficient order and would know which tasks to skip when there is not enough time to complete all the tasks.

RADAR's multitask coordination assistant (MCA) learns task metaknowledge by passively observing expert users. The MCA provides novice users with guidance, including a continuously updated schedule. The MCA is designed to support near-term deadlines on the order of 1–8 hours, which would be encountered during a typical user's workday, and it can handle situations in which the amount of work exceeds the time allotted.

The primary advice provided by the MCA is the suggested schedule, which specifies an order in which to perform tasks. For example, the MCA recommends updating the event constraints and room properties before running the schedule optimizer. The optimizer can incur both monetary and time costs, so running it multiple times is best avoided, which is strategy knowledge that a novice user is not likely to possess. The MCA learns what tasks may become necessary after a task is completed, and so it also adds such "expected" tasks to the schedule. For example, the user will need to reschedule existing vendor orders after the schedule optimizer creates a new schedule; this process might require 20–30 minutes of work, which is a significant portion of the two hour test. Showing the expected tasks in the schedule provides the user with a more realistic understanding of upcoming work and eliminates major changes to the schedule that would otherwise occur when an expected task actually becomes necessary. Additionally, the MCA identifies "critical" tasks, which are particularly important tasks that the user should definitely complete.

The MCA adapts to the user's actual behavior rather than force the user to follow the schedule rigidly. The suggested schedule specifies a total ordering of the tasks even though some tasks are not strictly ordered with respect to each other. While we considered visualizing the schedule as a partially ordered treelike diagram, we ultimately decided that the additional information would be more confusing than helpful. When users choose to deviate from the schedule, the MCA ignores deviations that are inconsequential. However, the MCA displays popup warning dialogues when the user significantly deviates from the suggested schedule. In particular, the warnings are issued if the user works on a critical task much earlier than experts did, if the users had not yet started working on a critical task by the time that most experts had, or if the user starts working on a critical task that is not the next critical task on the suggested schedule.[9] Additionally, the MCA adapts the schedule to the user's observed performance. For example, the MCA observes the user's pace relative to experts and then scales the planned task durations accordingly. When the MCA calculates that there is not enough time remaining to perform all incomplete tasks, it recommends specific tasks to skip (Varakantham and Smith 2008). These behaviors are examples of how technology can adapt to the user, not the other way around (see the usability benefits theme article [Lieberman 2009] in this issue). The user is ultimately in control; the system never prevents them from doing anything.

The primary user interfaces for displaying the suggested schedule are the "order" column in the action list (see figure 2) and the progress bar (see figure 3). The progress bar appears at the bottom of

the screen and always remains visible without obscuring other windows. Time is represented on the horizontal axis, which in this case spans two hours. An inverted black triangle and vertical red line represent the current time (c), which increases from left to right. Each box represents a task. Task boxes to the left of the current time represent completed (a) or deleted (b) tasks, providing a record of the user's progress so far. The width of each of these task boxes represents the time that the user spent working on the task.

The suggested schedule is visualized by the tasks to the right of the current time. The width of each of these task boxes represents the amount of time that the MCA expects the tasks to require. Blue task boxes represent noncritical tasks (g). Orange task boxes represent critical tasks and are designed to have higher visual salience (f). Gray boxes represent "expected" tasks (d); expected tasks that are critical appear as taller gray boxes (h).

The user can quickly inspect any task by moving the mouse over its task box (e), which updates the status bar at the bottom to show the highlighted task's description, status, actual/planned start time, actual/planned duration, and priority. The highlighted task, along with all other tasks of the same type, is drawn with a thicker border to allow the user to see where that type of task is distributed throughout the schedule. Double-clicking a box opens up the corresponding task. The number of overflow tasks, which are the ones the MCA proposes to skip due to time constraints, appears at the bottom right.

The MCA's advice also appears in other parts of the RADAR user interface. First, the action list's "order" column shows the position of each future task within the suggested schedule (see figure 2). Only tasks in the "incomplete actions" table are included in the suggested schedule. Sorting the "incomplete actions" table by the "order" column shows the schedule as an ordered to-do list. Second, the specific tasks that the MCA suggests that the user skip are shown in the "overflow actions" table; the progress bar only shows the number of such tasks. Third, after the user completes or deletes a task, RADAR provides a link to the next suggested task, which allows the user to move to it without having to return to the action list.

The MCA learns models of expert behavior by passively observing experts performing tasks using the same user interfaces that test participants will later use. To train the system, experts did the two-hour study using a version of the system for which the MCA learning components were watching rather than recommending. Other AI components operated normally. For example, the e-mail classifier had already analyzed the e-mails and identified tasks. The training used three different sets of e-mails (none of which was the test e-mail set), which provided variability to prevent overtraining. In a real deployment, the MCA would use online learning, so it would continuously adapt to the user's choices.

Agent-Assisted Form Filling

Many of the conference evaluation tasks are completed by entering information from an e-mail into a form. As our contextual inquires showed, form filling is a common activity since people often receive structured information as unstructured, plain text. Examples of structured information include calendar appointments, contacts, addresses, purchase orders, bibliographies, and so forth. The receiver needs to enter the information into a forms-based user interface, a process that can be tedious and error prone.

A number of natural language processing systems have demonstrated the feasibility of parsing natural language into structured formats (Culotta et al. 2007; Stylos, Myers, and Faulring 2004; Zimmerman et al. 2007). Such systems offer the potential to significantly reduce the tedious task of entering the information in each field. However, such systems can make errors, so the form-filling user interface needs to help the user to identify and correct the errors. We designed novel visualizations and interaction techniques that help the user to do that efficiently.

The user can access a task's form several ways: clicking the action's description link in the action list (for example, "Modify Room: Flagstaff: Sternwheeler" in figure 2.a.ii), clicking the next suggested action link after completing another action, or double-clicking a task box in the progress bar. As seen in figure 4, the e-mail appears on the left, and the task's form on the right. Meredith, who sent this e-mail, works at the Flagstaff hotel, which is one of the places where a RADAR test user, acting as the conference planner, can schedule conference events. Meredith's e-mail describes some inaccuracies and out-of-date information for a particular room on the hotel's website. The user's task is to enter the updated information into the RADAR schedule optimizer's room property form for the Sternwheeler room. Updating the room's properties provides the schedule optimizer with the most accurate information about the state of the simulated conference world so that it can produce the best conference schedule.

Figure 4 shows the forms-based interface after RADAR has already selected the correct task, Modify Room, and record, Sternwheeler. This form-filling interface includes visualizations and interaction techniques that allow the user to see the changes that RADAR has proposed[10] and to correct any errors. RADAR has proposed new values for fields that already had values (e, f, and j), assigned

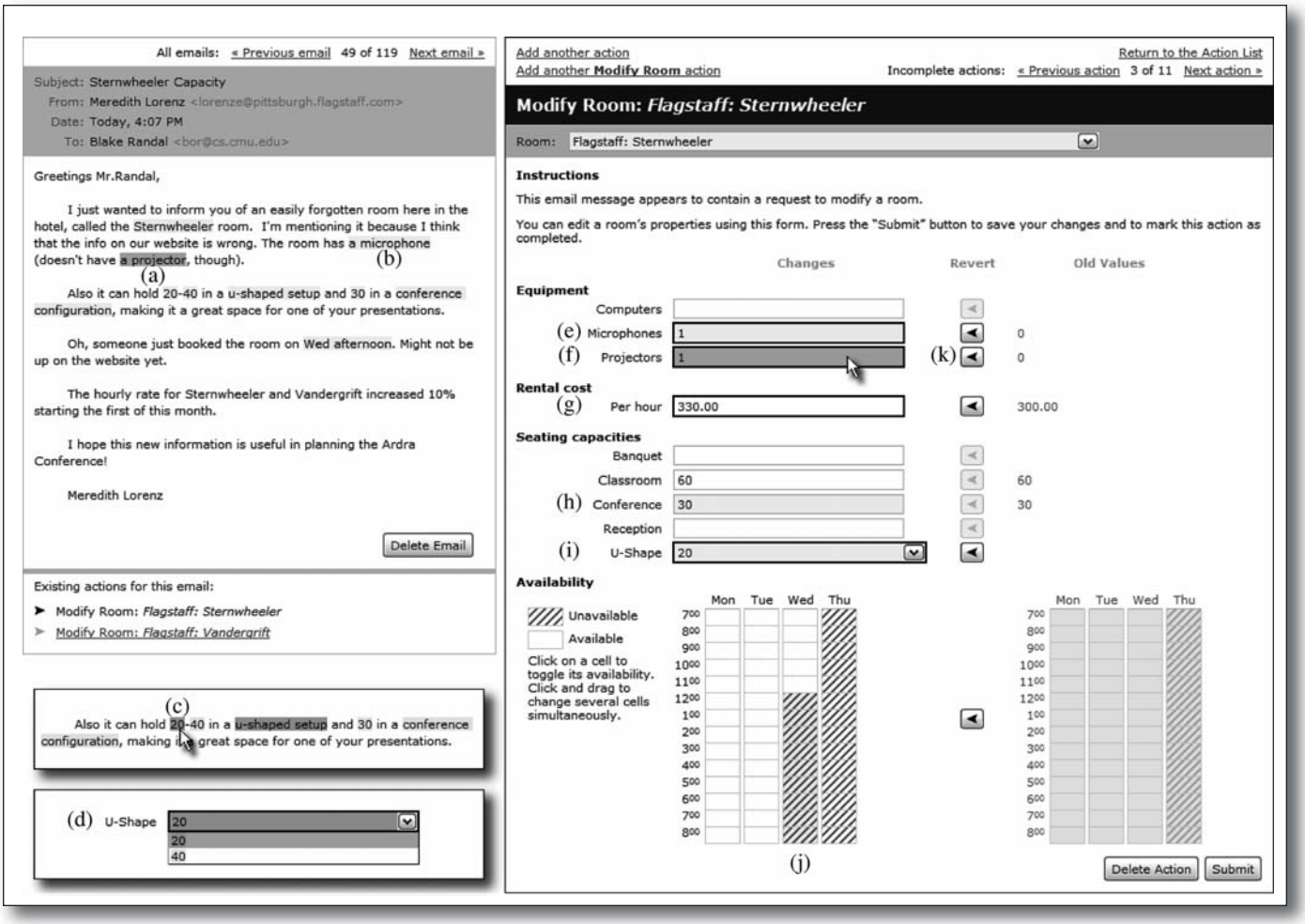

Figure 4. Forms-Based Interface.

The e-mail on the left contains information about a room. The form on the right is used to edit that room's properties. After analyzing the e-mail, RADAR proposes values for several fields (e, f, h, i, and j). The text segment within the e-mail used by RADAR to derive the proposed field's values is called an anchor and is drawn with an orange background (a and b). Moving the mouse or cursor over a proposed field value (f) highlights the associated anchor (a). Additionally, moving the mouse over an anchor (c) will open the menu of proposed field values (d)

a value to a field that was previously empty (i), and decided that an existing value was correct (h). We also designed and implemented two user interface features that the current RADAR agents do not fill in. First, the user interface allows an agent to propose multiple possible values for a field, rather than discard all but its top choice. All of the proposed values appear in a menu, which allows the user to select the correct value when it is not the top choice (d and i). Second, a special availability grid allows the user to specify availability constraints, such as when an event could be scheduled (j). While RADAR agents do not propose changes to the availability constraints, the schedule optimizer does use the availability constraints entered by the user when generating the conference schedule.

The user interface helps the user to identify the changes that RADAR has proposed and to fix any incorrect ones.[11] The original value for each field appears in the "old values" column on the right. When a field's current value (left column in the form) differs from its old value, the border of the field changes to a thick black border and the "revert" button becomes enabled (k). In the example, RADAR has incorrectly proposed a value of "1" for the projectors field (f). The user can undo this erroneous change by pressing the revert button (k), which restores the field to its old value. The field-modified visualizations are independent of whether or not RADAR proposed a value for the field. For example, the conference seating capacity field has a proposed value that equals the old value, so the field-modified visualization is not enabled (h). For another example, consider when the user edits the per hour rental-cost field, which RADAR had not proposed changing (g). After the

Condition	N	Score Mean (Std. Dev.)
Without Learning	23	0.550 (0.072)
Without MCA	28	0.706 (0.063)
With MCA	28	0.754 (0.035)

Table 1. Evaluation Scores (Mean and Standard Deviation) for the August 2008 Test.

"N" denotes the number of participants in each condition.

user changes the field's value, the field-modified visualizations are enabled.

The user interface allows RADAR to specify the text regions from which it derived a proposed field value. Such a text segment is called an anchor since the proposed field value in the form links back to the source text. The anchor text within the source e-mail and the proposed field values (e, f, h, i, and j) are both drawn with an orange background. A proposed field value can link to multiple noncontinuous segments of text. For example, the "U-Shape" seating capacity field's proposed value of "20" links to both the "20" and "u-shaped setup" text segment in the source text. By highlighting the text segments within the e-mail, the user can tell whether or not RADAR has identified a given text segment as relevant to any field. If RADAR misses some relevant text, the lack of an anchor should alert the user to the omission. When the user moves the mouse cursor over an anchor, the anchor (a) and associated field (f) are drawn with a darker orange background to visually associate the proposed field value with its anchor text. The same visual change appears when the user moves the mouse or text cursor to a field (f). Additionally, moving the mouse over an anchor (c) will open the menu of proposed field values (d). This technique provides more information than VIO, which uses a single static highlight color (Zimmerman et al. 2007), and Culotta and colleagues' system, which uses different colors for each group of related fields (Culotta et al. 2007).

Evaluation Results

We evaluated RADAR using the conference planning test to determine how effective it is at assisting novice users based upon learned models of expert performance.[12] The test compared participant performance among three conditions: Without Learning, which had no learned models; Without MCA, which had all the learned models except for the MCA ones; and With MCA, which had all AI systems active. We report here the results of the fourth evaluation, which was conducted in August 2008.

An evaluation score, designed by external program evaluators, summarized overall performance into a single objective score ranging from 0.000 to 1.000, with higher scores reflecting better performance on the conference planning test (for full details, see Steinfeld et al. [2007]). It was important that this score be tied to objective conference planning performance rather than a technology-specific algorithm (for example, F1 for classification). This technology-agnostic approach allowed us to compare performance across conditions given any technology. The single objective score was calculated from points earned as a result of satisfying certain conditions, coupled with costs and penalties. These included the quality of the conference schedule (for example, constraints met, special requests handled), adequate briefing to the conference chair, accurate adjustment of the website (for example, contact information changes, updating the schedule on the website), costs for the rooms, food, and equipment for the conference, and penalties for requesting that others give up their existing room reservations. The score coefficients were two-thirds for the schedule, one-sixth for website updating, and one-sixth for briefing quality.

On this measure, With MCA participants clearly outperformed Without MCA participants, who in turn outperformed Without Learning participants (ANOVA, $F(2,76) = 83.7$, $p < 0.0001$) (see figure 5 and table 1). A subsequent Tukey post-hoc test found that the three conditions were significantly different from each other. All but 3 of the 28 With MCA participants earned higher scores than the average score of the Without MCA participants (see figure 5).

Figure 5 shows that several participants in the Without MCA condition earned much lower scores than others earned, which we call the long tail of performance. Earlier pre-MCA RADAR evaluations had yielded similar results. One of the motivations for building the MCA was to improve the performance of the low-performing participants. The standard deviation of the evaluation score dropped 44 percent from the Without MCA to With MCA condition, and the long tail of poor performance in the Without MCA condition did not occur in the With MCA condition.

In summary, the evaluation of RADAR with and without the MCA has shown that it improves the mean performance and reduces performance variance by helping users who struggle the most. Such results highlight how AI technologies can provide functionality that helps users efficiently accomplish their goals and can adapt to different users' needs.

Conclusions

The five-year RADAR project has concluded with great success. A formal evaluation of the RADAR system showed that users, who were confronted with an e-mail overload test, performed much better when assisted by learning technologies. The HCI methods and principles described above were crucial in building a mixed-initiative system that was both usable and useful, making the intelligent outputs understandable and controllable by users. Out of the hundred people who worked on RADAR, only a few had extensive HCI training. Based upon this experience, we believe that AI projects can produce significantly more usable systems if they include even a few HCI specialists, as long as they are involved with the project from the beginning and integrated with the development members of the team. We hope that this case study has made HCI methods and principles more accessible to AI researchers by showing how they apply to AI systems. Additionally, our iterative design approach with close cooperation among the developers of the AI and user interface components provides a good model for future intelligent system development.

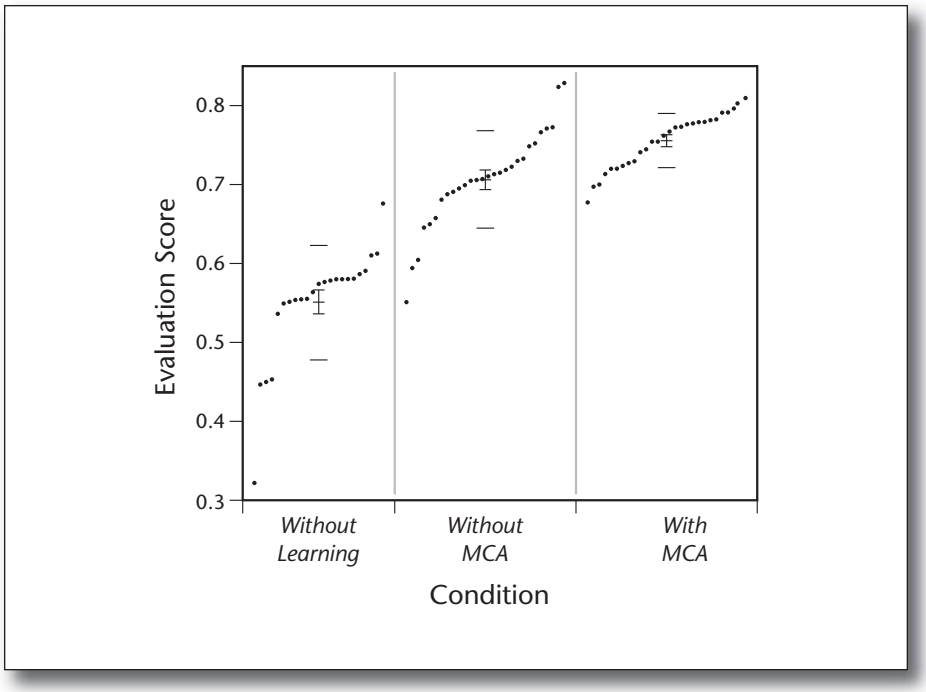

Figure 5. Evaluation Scores

The evaluation scores show that the MCA advice in the With MCA condition significantly improved performance, reduced the performance variation, and eliminated the long tail of poor performance.

Acknowledgements

The authors thank Michael Freed, Geoff Gordan, Jordan Hayes, Jeffery Hansen, Javier Hernandez, Matt Lahut, Pablo-Alejandro Quinones, Bill Scherlis, Bradley Schmerl, Nicholas Sherman, Daniel Siewiorek, Asim Smailagic, Stephen Smith, Fernando de la Torre, Pradeep Varakantham, Jigar Vora, Yiming Yang, Shinjae Yoo, Gabriel Zenarosa, and John Zimmerman. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010.

Notes.

1. RADAR illustrates how AI can help users accomplish their personal goals in ways that fit into their desired work practices (see the usability benefits theme article [Lieberman 2009] in this issue). RADAR explicitly models tasks and their interrelationships and applies this model to particular cases while the user is working. By contrast, the analyses of users' tasks achieved with HCI usability engineering methods yield an understanding that informs the system's interaction design but is normally not represented or used in the system (see the usability engineering methods theme article [Spaulding and Weber 2009] in this issue).

2. Significant portions of the evaluation materials are available to interested researchers at www.cs.cmu.edu/~airspace.

3. The contextual inquiries conducted in the RADAR project helped to identify the specific ways in which users could benefit the most from the addition of intelligent functionality (see the usability engineering methods theme article [Spaulding and Weber 2009] in this issue).

4. The regular interaction between AI experts and usability specialists in the RADAR team made possible a "binocular" perception of the design problems and possible solutions, so that decisions about technology and interaction design could be made together rather than separately (see the introduction [Jameson et al. 2009] and usability engineering methods theme article [Spaulding and Weber 2009] in this issue).

5. Such commonsense knowledge can improve a system's ability to understand and communicate with users, thereby enhancing the system's usability (see the usability benefits theme article [Lieberman 2009] in this issue).

6. The different types of errors that a system may make (for example, missed opportunities versus false positives) can have qualitatively different consequences for the user's responses to the system, ranging from diminished effectiveness to confusion and loss of trust. Moreover, these consequences can change over time as a function of the user's experience. (For a more general discussion see the usability side-effects theme article [Jameson 2009] in this issue.)

7. There were actually more tasks than task labels since some e-mails contained multiple tasks of the same type.

8. Because the task-centric interface reproduces many features of familiar e-mail clients, it helps to avoid the usability side effect of diminished aesthetic appeal (see the usability side-effects theme article [Jameson 2009] in this issue).

9. The way in which the MCA allows the user to deviate from its suggested schedule, while warning the user about potentially dangerous deviations, represents a novel way of giving the user control over the nature of the interaction with an intelligent system while still retaining the benefits of intelligence (see the usability side-effects theme article [Jameson 2009] in this issue).

10. The e-mail used in this example was written to demonstrate all the features of the user interface, and so it is more complicated than a typical e-mail used during the

test. We did not run the natural language-processing component on this e-mail.

11. This interface is a good example of an effort to mitigate the usability side effects of imperfect system performance by helping users to detect possible errors before they have any consequences (see the usability side effects theme article [Jameson 2009] in this issue).

12. This section describes an especially elaborate evaluation of an intelligent interactive system, and it illustrates some of the points discussed in the corresponding section of the usability engineering theme article (Spaulding and Weber 2009) in this issue.

References

Bellotti, V.; Ducheneaut, N.; Howard, M.; Smith, I.; 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.

Beyer, H., and Holtzblatt, K. 1998. *Contextual Design: Defining Customer-Centered Systems*. San Francisco: Morgan Kaufmann Publishers.

Culotta, A.; Kristjansson, T.; McCallum, A.; and Viola, P. 2007. Corrective Feedback and Persistent Learning for Information Extraction. *Artificial Intelligence* 170(14–15): 1101–1122.

Fahlman, S. E. 2006. Marker-Passing Inference in the Scone Knowledge-Base System. In *Proceedings of First International Conference on Knowledge Science, Engineering and Management,* 114–126. Berlin: Springer.

Fink, E.; Bardak, U.; Rothrock, B.; and Carbonell, J. G. 2006. Scheduling with Uncertain Resources: Collaboration with the User. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics,* 11–17. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Freed, M.; Carbonell, J.; Gordon, G.; Hayes, J.; Myers, B.; Siewiorek, D.; Smith, S.; Steinfeld, A.; and Tomasic, A. 2008. RADAR: A Personal Assistant That Learns to Reduce E-Mail Overload. In *Proceedings of Twenty-Third Conference on Artificial Intelligence,* 1287–1293. Menlo Park, CA: AAAI Press.

Garlan, D., and Schmerl, B. 2007. The RADAR Architecture for Personal Cognitive Assistance. *International Journal of Software Engineering and Knowledge Engineering* 17(2): 171–190.

Gwizdka, J. 2002. TaskView: Design and Evaluation of a Task-based E-Mail Interface. In *Proceedings of Conference of the Centre for Advanced Studies on Collaborative Research,* 4. Toronto, Ontario, Canada: IBM Press.

Jameson, A. 2009. Understanding and Dealing with Usability Side Effects of Intelligent Processing. *AI Magazine* 30(4).

Jameson, A.; Spaulding, A.; and Yorke-Smith, N. 2009. Introduction to the Special Issue on Usable AI. *AI Magazine* 30(4).

Kumar, M.; Das, D.; and Rudnicky, A. I. 2007. Summarizing Nontextual Events with a "Briefing" Focus. In *Proceedings of Recherche d'Information Assistée par Ordinateur (RIAO)*. Paris: Centre De Hautes Etudes Internationales D'Informatique Documentaire.

Lieberman, H. 2009. User Interface Goals, AI Opportunities. *AI Magazine* 30(4).

Nyberg, E.; Riebling, E.; Wang, R. C.; and Frederking, R. 2008. Integrating a Natural Language Message Pre-Processor with UIMA. Paper presented at the Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP Workshop at the Sixth Conference on Language Resources and Evaluation, Marrakech, Morroco, 31 May.

Spaulding, A., and Weber, J. 2009. Usability Engineering Methods for Interactive Intelligent Systems. *AI Magazine* 30(4).

Steinfeld, A.; Bennett, S. R.; Cunningham, K.; Lahut, M.; Quinones, P.-A.; Wexler, D.; Siewiorek, D.; Hayes, J.; Cohen, P.; Fitzgerald, J.; Hansson, O.; Pool, M.; and Drummond, M. 2007. Evaluation of an Integrated Multi-Task Machine Learning System with Humans in the Loop. In *Proceedings of NIST Performance Metrics for Intelligent Systems Workshop (PerMIS)*. Washington, DC: National Institute of Standards and Technology.

Stylos, J.; Myers, B. A.; and Faulring, A. 2004. Citrine: Providing Intelligent Copy and Paste. In *Proceedings of Symposium on User Interface Software and Technology*, 185–188. New York: Association for Computing Machinery.

Varakantham, P., and Smith, S. 2008. Linear Relaxation Techniques for Task Management in Uncertain Settings. In *Proceedings of International Conference on Automated Planning and Scheduling*, 363–371. Menlo Park, CA: AAAI Press.

Whittaker, S.; Bellotti, V.; and Gwizdka, J. 2006. E-Mail in Personal Information Management. *Communications of the ACM* 49(1): 68–73.

Whittaker, S., and Sidner, C. 1996. E-Mail Overload: Exploring Personal Information Management of E-Mail. In *Proceedings of Conference on Human Factors in Computing Systems*, 276–283. New York: Association for Computing Machinery.

Yang, Y.; Yoo, S.; Zhang, J.; and Kisiel, B. 2005. Robustness of Adaptive Filtering Methods in a Cross-Benchmark Evaluation. In *Proceedings of the 28th Annual International ACM SIGIR Conference*, 98–105. New York: Association for Computing Machinery.

Yang, Y.; Zhang, J.; and Kisiel, B. 2003. A Scalability Analysis of Classifiers in Text Categorization. In *Proceedings of 26th Annual International ACM SIGIR Conference*, 96–103. New York: Association for Computing Machinery.

Zimmerman, J.; Tomasic, A.; Simmons, I.; Hargraves, I.; Mohnkern, K.; Cornwell, J.; and McGuire, R. M. 2007. VIO: A Mixed-initiative Approach to Learning and Automating Procedural Update Tasks. In *Proceedings of Conference on Human Factors in Computing Systems*, 1445–1454. New York: Association for Computing Machinery.

Andrew Faulring is a research programmer in the Human-Computer Interaction Institute at Carnegie Mellon University. His research interests include building usable interfaces for AI technologies. He was heavily involved in the design and implementation of many of RADAR's user interfaces. He earned a B.S. in computer science with a second major in human-computer interaction and an M.S. in computer science, both from Carnegie Mellon University.

Ken Mohnkern was an interaction designer on the RADAR project. He is currently a user experience architect at Buzzhoney, a web-marketing firm in Pittsburgh, PA.

Aaron Steinfeld, Ph.D., is a systems scientist in the Robotics Institute at Carnegie Mellon University. He was the lead for RADAR evaluation activities and is currently the codirector of the Rehabilitation Engineering Research Center on Accessible Public Transportation (RERC-APT) and the area lead for transportation-related projects in the Quality of Life Technology Engineering Research Center (QoLT ERC). His research focuses on operator assistance under constraints, that is, how to enable timely and appropriate interaction when technology use is restricted through design, tasks, the environment, time pressures, or user abilities.

Brad A. Myers, Ph.D., is a professor in the Human-Computer Interaction Institute at Carnegie Mellon University. His research interests include user interface development systems, user interfaces, handheld computers, programming environments, programming language design, programming by example, visual programming, interaction techniques, and window management. He is an ACM Fellow, winner of six best paper awards, and a member of the CHI Academy, an honor bestowed on the principal leaders of the field. He is a senior member of the IEEE, and also belongs to SIGCHI, ACM, the IEEE Computer Society, and Computer Professionals for Social Responsibility.