2008

Successful User Interfaces for RADAR

Andrew Faulring
Carnegie Mellon University

Brad Myers
Carnegie Mellon University

Ken Mohnkern
Carnegie Mellon University

Aaron Steinfeld
Carnegie Mellon University

Successful User Interfaces for RADAR

Andrew Faulring
HCI Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
faulring@cs.cmu.edu

Brad Myers
HCI Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
bam@cs.cmu.edu

Ken Mohnkern
HCI Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
kem@cs.cmu.edu

Aaron Steinfeld
Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
steinfeld@cmu.edu

CHI 2008, April 5 – April 10, 2008, Florence, Italy
Workshops and Courses: Usable Artificial Intelligence

Abstract
We are designing, implementing, and testing the user interface for RADAR (Reflective Agents with Distributed Adaptive Reasoning), which is a large multi-agent system that uses learning to help office workers cope with email overload and to complete routine tasks more efficiently. RADAR provides a mixed-initiative user interface in which artificial intelligence helps users perform the tasks that arrive in email messages. A large-scale user test of RADAR demonstrated the effectiveness of its user interface and AI.

Keywords
Task management, forms, form filling, anchors, intelligent user interfaces, agents, RADAR.

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces---Interaction styles, Graphical user interfaces (GUI).

RADAR's Usable User Interfaces
RADAR includes many AI technologies such as an email classifier, natural language processor, schedule optimizer, webmaster, and briefing assistant. A large-scale user test has evaluated several versions of the RADAR system [3]. The test measures RADAR's performance using quantitative metrics acquired through data logging and qualitative metrics collected

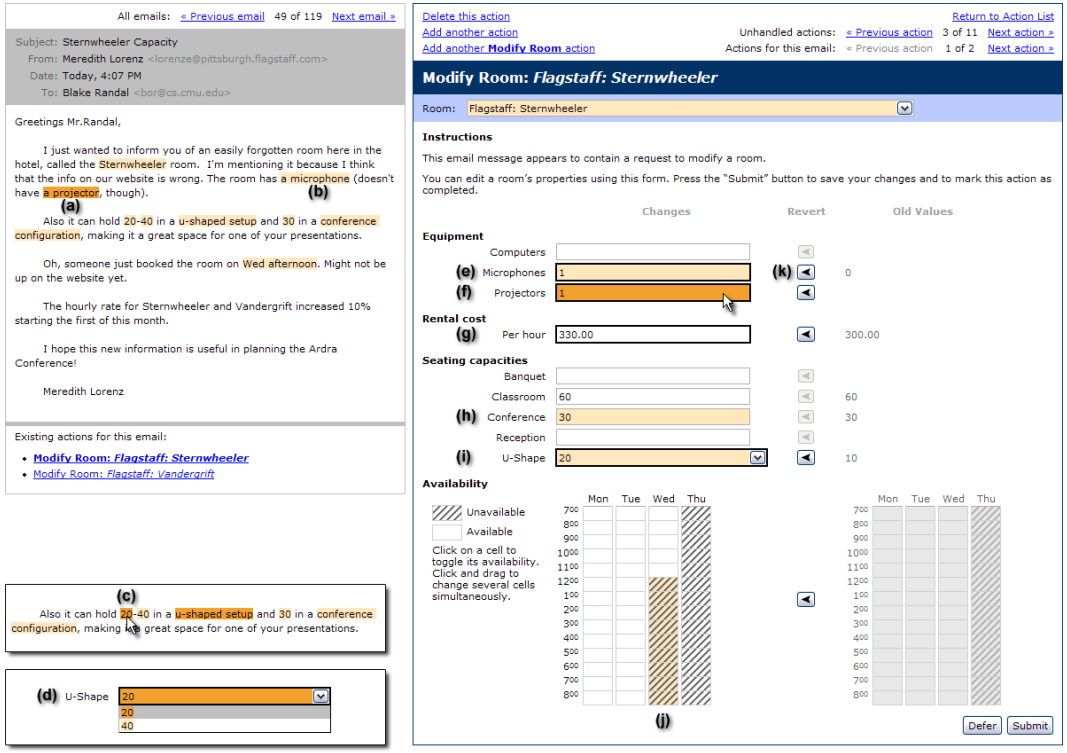

Figure 1: The email on the left contains information about a room. The form on the right is used to edit that room's properties. After analyzing the email, RADAR proposes values for several fields, which are shown with an orange background (e, f, h, i, and j). The text segment within the email used by RADAR to derive the proposed field's values is called an *anchor* and is also 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).

with a post-test user survey [4]. The post-test user survey for last year's RADAR 1.1 test found that RADAR's AI had very little impact on user perceptions of the system. It was hypothesized that the poor user interfaces for many of RADAR's components were masking the potential benefit of the AI [4]. So we undertook a system-wide effort to improve the user interface for RADAR 2.0. The post-test survey for this year's RADAR 2.0 study found that the AI now positively impacted user perceptions of the RADAR system. In particular, participants were more confident that they had done tasks well, had found tasks easier to complete, had been more immersed in the test, and stated that they had completed more of the test. These qualitative metrics were supported by a statistically significant improvement in the RADAR test evaluation score, which summarizes overall performance into a single objective number.

We attribute the improvements, in part, to several user interface innovations. First, we designed a task-focused interface for managing the tasks contained within email messages. This approach follows recent work on task management within email applications [1, 2, 5]. In our approach, the user's inbox is augmented with a separate task list. RADAR's email classifier identifies tasks contained within an email and automatically creates an initial set of tasks. A single email can spawn zero, one, or many tasks. When the classifier errs, the task list interface allows the user to add, delete, or modify tasks. Second, performing many of the tasks requires that the user read an email and then enter its information into a forms-based user interface. RADAR's natural language processing (NLP) component proposes values for the form's fields based upon the content of the email. We developed novel visualizations and interaction techniques for allowing the user to check and, when necessary, correct the proposed field values (see Figure 1). The user study showed that our user interface was successful.

How Aggressive Should AI Be?

The AI in RADAR 2.0 offers to help the user only when it has high confidence that its proposal is correct. For example, when searching for tasks contained in email messages, the email classifier favors minimizing false positives (5.7%, 5 out of 87) at the expense of increasing false negatives (48.2%, 42 out of 87). The email classifier developers made this tradeoff out of concern that users, who were novices with respect to the RADAR system, would not recognize when RADAR made a mistake. More experience users, who are better able to identify mistakes, might prefer the system to favor recall over precision, since for those users a missed task incurs a greater cost than does an incorrect proposal. Perhaps the user interface should display the AI's confidence in each of its decisions to inform the users in what cases the proposal is more likely to be wrong.

In general, how aggressive should the AI be, and how should system designers weigh the benefits of AI automation against its costs? AI automation offers a number of benefits to the user. AI can help users to complete tasks faster and can perform the tedious or repetitive parts of tasks, leaving the user to handle the difficult or special case situations. AI may increase accuracy if it handles the mundane, easy cases, allowing users to focus their limited energy and attention on the more difficult ones. AI can also have a more fundamental impact, for example by supporting a more efficient workflow. The task-centric workflow enabled by RADAR's email classifier would be less effective if the user also had to manually detect all of the tasks contained within the emails.

However, AI automation also entails potential costs. While the cost of an error varies with each task, the costs can be grouped into several categories. First, mistakes made by the AI may be attributed to the user, causing the user to look bad. For example, when a user fails to respond to an email because their spam filter incorrectly flags it as spam, the sender might assume that the receiver chose to ignore the message. In another example, a meeting scheduling agent might respond to a meeting request that the user is unavailable, even though the requestor was important enough that the user would have adjusted their schedule to accommodate the request. Second, the AI might lack information, thereby causing it to take non-optimal action. For example, a meeting scheduling agent, unaware of all of the user's preferences and constraints, might make non-optimal scheduling decisions. Third, the effort required by the user to correct the AI's error may exceed the effort required had the user performed the task without the AI's assistance.

One way to mitigate these costs is to limit the impact of the AI action. The AI might be limited to only performing undoable actions that are not visible to other people. For example, a calendar scheduling agent might just propose possible times for a meeting to the user, rather than automatically replying to the meeting requestor. In RADAR, the NLP proposes values for form fields, but the user has the final say before submitting the form. The AI's role is to propose actions to the user, who then checks and possibly corrects them.

In a mixed-initiative system, errors in the final results have several origins. First, the user might make a mistake regardless of the AI. Second, the user might

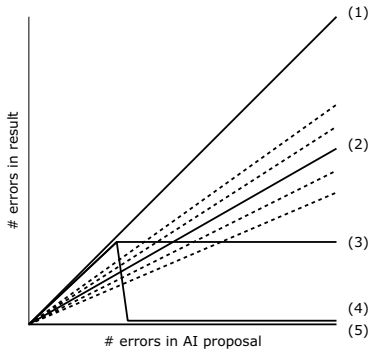

Figure 2: Curves representing the relationship between the number of errors that the AI makes and the number of errors in the final result after the user has checked the AI's proposals.

not notice an incorrect proposal by the AI. Third, the user might not recognize that the AI's proposal is wrong. If the user is unfamiliar with the task domain, the agent's proposal might bias their thinking, causing the user to decide that an incorrect proposal is correct. Even when the user is familiar with the task domain, users are likely to accept the agent's proposal.

Thus, key questions are what is the likelihood that the user notices and corrects errors, what factors affect that likelihood, and how can we design user interfaces that support the process. Figure 2 shows a conceptual graph relating the number of errors in the AI proposal to the number of errors in the final result. Curves (1) and (5) represent the cases where the user corrects zero or all AI errors, respectively. With curve (2), the user corrects some errors while missing others. Curve (2) is an instance of a family of similar curves, each representing the effectiveness of a different visualization and error correction technique for agent proposals. This family of curves need not be linear (3) or even monotonic (4). After repeated exposure to the similar errors, the user may learn to anticipate them (3). Alternatively few errors on the same screen may be missed, but several errors might be noticed (4). An additional factor to consider is how many opportunities the user will have to notice an error. If the error is visible in different contexts, it might be more likely that the user will notice it.

We plan to explore these questions and search for more effective visualization and interaction techniques through our continuing work on the RADAR user interface.

Conclusion

We have extensive experience designing and evaluating user interfaces for AI systems. Our work on the RADAR user interfaces yielded measurable improvements to the system. Hence, we have much to contribute the workshop discussion on Usable AI.

Acknowledgements

The authors thank S. Rachael Bennett, Polo Chau, Jordan Hayes, Pablo-Alejandro Quinones, Bill Scherlis, Bradley Schmerl, and Anthony Tomasic. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010.

References

[1] Bellotti, V., Ducheneaut, N., Howard, M., Smith, I. and Grinter, R.E. Quality Versus Quantity: E-Mail-Centric Task Management and Its Relation With Overload. *Human-Computer Interaction 20*, 1/2 (2005), 89–138.

[2] Gwizdka, J. TaskView: Design and Evaluation of a Task-based Email Interface. *Proc. CASCON*, IBM Press (2002).

[3] 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. Evaluation of an Integrated Multi-Task Machine Learning System with Humans in the Loop. *Proc. PerMIS*, NIST (2007).

[4] Steinfeld, A., Quinones, P.-A., Zimmerman, J., Bennett, S.R. and Siewiorek, D. Survey Measures for Evaluation of Cognitive Assistants. *Proc. NIST Performance Metrics for Intelligent Systems Workshop (PerMIS)*, NIST (2007).

[5] Whittaker, S., Bellotti, V. and Gwizdka, J. Email in Personal Information Management. *CACM 49,* 1 (2006), 68–73.