Enabling Rich Human-Agent Interaction for a Calendar Scheduling Agent

Andrew Faulring and Brad A. Myers
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
{faulring, bam}@cs.cmu.edu

ABSTRACT
The RhaiCAL system provides novel visualizations and interaction techniques for interacting with an intelligent agent, with an emphasis on calendar scheduling. After an agent interprets natural language containing meeting information, a user can easily correct mistakes using RhaiCAL's clarification dialogs, which provide the agent with feedback to improve its performance. When an agent proposes actions to take on the user's behalf, it can ask the user to confirm them. RhaiCAL uses novel visualizations to present the proposal to the user and allow them to modify the proposal, and informs the agent of the user's actions in a manner that supports long-term learning of the user's preferences. We have designed a high-level XML-based language that allows an agent to express its questions and proposed actions without mentioning user interface details, and that enables RhaiCAL to generate high-quality user interfaces.

Categories and Subject Descriptors: H.5.2 [Information Interfaces and Presentation]: User Interfaces---Interaction styles, Natural language, Graphical user interfaces (GUI); H.4.1 [Information Systems Applications]: Office Automation---Time management; I.3.6 [Computer Graphics]: Methodology and Techniques---Interaction techniques.
General Terms: Design, Human Factors.
Keywords: Rich human-agent interaction, interface agents, intelligent user interfaces, calendars, meeting scheduling.

INTRODUCTION

We are investigating user interfaces for *R*ich *H*uman-*A*gent *I*nteraction (Rhai, pronounced "ray") as part of Radar (*R*eflective *A*gent with *D*istributed *A*daptive *R*easoning), a larger interdisciplinary project that is building a suite of intelligent agents to help office workers easily complete routine tasks. Researchers are building agents such as a Calendar Manager [7, 9], a Webmaster [13], and a Space Planner.

We present RhaiCAL, which integrates work on scheduling algorithms in artificial intelligence and groupware calendar tools, to help a user who has to schedule meetings, particularly multi-person ones with busy people. The system works as follows. First, an agent watches for emails containing meeting scheduling information. When the agent finds meeting information, RhaiCAL's clarification dialog allows the agent to check its understanding. Then, the agent proposes actions to help the user schedule the meeting. RhaiCAL makes it easy for the agent to present alternative actions that the user can edit. RhaiCAL provides visualizations of the constraints and preferences that help the user make good decisions, rather than just asking the user to confirm a single action.

RhaiCAL helps users with the difficult task of scheduling multi-person meetings. Palen notes that scheduling is more of a "satisficing" rather than an "optimizing" task [10]. The meeting organizer gathers constraints and preferences from each invitee and attempts to find a satisfactory time. With busy professionals such as managers and college faculty, the problem is often over-constrained, thereby forcing the organizer to negotiate with the invitees to determine the constraints or preferences that can be relaxed. The negotiation can take multiple rounds and could grind to a halt should an invitee fail to respond in a timely manner.

When no common free times exist, shared calendaring tools do not make it easy for a meeting organizer to identify possible meetings times. Such tools do not allow invitees to represent many of their implicit, idiosyncratic preferences such as how early in the day they are willing to meet, how many meetings they are willing to have back-to-back, when they prefer to eat lunch, and so forth. Furthermore a user's calendar may not include all their actual time commitments, such meeting preparation and commute time [10]. A meeting organizer browsing the shared calendars of invitees in search of common available times must "assess the quality of what appears to be free time in someone's calendar" [10]. That assessment, in part, relies on the constraints and preferences not represented in the shared calendar.

When shared calendars are not in use, the meeting organizer often resorts to email to gather constraints and preferences. Current calendaring tools provide invitees with little help responding to meeting requests. Manually converting a graphical calendar representation into prose that describes either explicit ("Monday: 10–12") or vague ("Wednesday afternoon") constraints and preferences is a tedious and error prone process.

Current calendaring tools also make it difficult for the meeting organizer to find a satisfactory time. The natural lan-

CHI 2005, April 2–7, 2005, Portland, Oregon, USA.
ACM 1-59593-002-7/05/0004.

guage responses are an especially poor problem representation compared to a graphical calendar representation. Converting from natural language into a calendar representation is tedious and error prone. Furthermore, current calendaring tools poorly support the decision making process because they cannot represent the variety of constraints and preferences that invitees specify. And once an organizer has found a set of possible times, they face the laborious task of composing a natural language proposal to send to the invitees.

To address some of these deficiencies, we have developed the RhaiCAL visualizations and interaction techniques. These innovations provide the core of a direct manipulation interface that better supports the process of scheduling meetings. However, even building such a direct manipulation interface does not eliminate many of the steps. The organizer would still be responsible for finding possible solutions. Invitees would still have to encode their constraint and preference information in the calendar, and to read and respond to emails.

One approach to addressing the remaining problems involves using an intelligent calendar scheduling agent. Several researchers have described systems that successfully handle parts of the process [3, 6, 8]. Our collaborators are developing CMRadar [7, 9], which uses multi-agent negotiation strategies and schedule optimization algorithms based upon user preferences to help the user by (1) negotiating meeting times on the user's behalf, (2) finding possible schedules given external meeting requests, and (3) asking the user which schedules they prefer. Unfortunately, these research systems have focused on artificial intelligence and neglected the user interface that is so important. We believe that user interfaces that show third-party constraints and the agent's proposed actions, and that provide rich and natural interaction techniques for user correction and control can enable agents to be a significant help to users in many situations.

Along with the benefits that agents bring come additional challenges, which can be addressed with good user interfaces for rich interaction between humans and agents. First, agents misunderstand natural language requests and will need to ask the user to check and, if necessary, correct their interpretation. Second, agents may have inaccurate models of the user's preferences, requirements, and constraints, and so will need to check that their proposed actions are acceptable. Third, even if agents become as good as human assistants, the agents will still need to consult their supervisors, just as human assistants do, when requests are underspecified, have ambiguous instructions, deviate from normal, or have changed. Fourth, the user interface should allow the user to understand and control the agent's behavior, which affects the user's trust in the agent [12].

RhaiCAL contributes solutions to these problems. First, it provides clarification dialogs that allow an agent to check its interpretation of natural language or to request additional information. The dialog allows an agent to suggest multiple options, in case its second or third suggestion is correct.

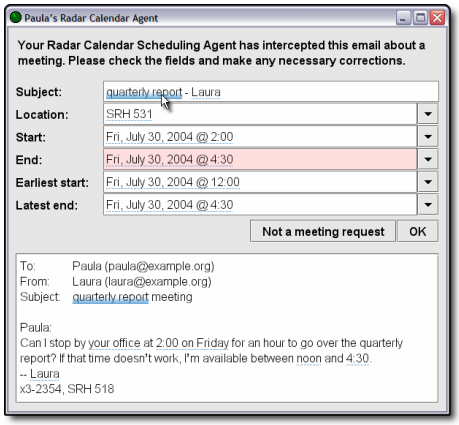


Figure 1: A clarification dialog asks the user to check the agent's understanding of the email shown at the bottom.

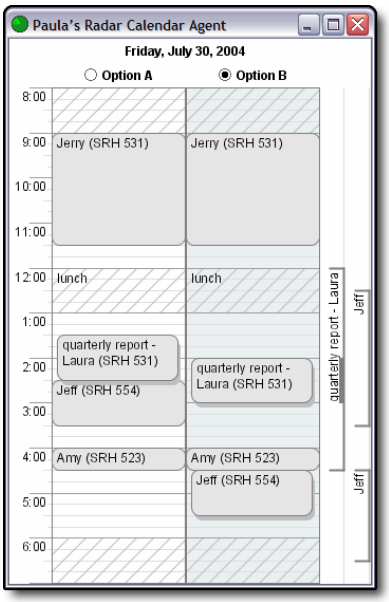
Second, RhaiCAL provides a calendar-based visualization of an agent's actions and proposals that allow the user to see and fix them in an appropriate context. Both of these innovations provide an agent with feedback that it can use to learn how to improve its performance.

NATURAL LANGUAGE CLARIFICATION

Radar provides a natural language processing component for its agents. An agent can ask the user to check its interpretation of natural language with a RhaiCAL clarification dialog box such as the one in Figure 1. To invoke the dialog, an agent generates a message in RhaiCAL's XML-based language that specifies the fields that it has extracted. The agent's instructions appear at the top, the fields of the meeting request in the middle, and text of the original email at the bottom. For example, the agent parses the email at the bottom of Figure 1 and then proposes a subject that contains two phrases extracted from the source text: "quarterly report" and "Laura." RhaiCAL's XML-based language allows the agent to specify the location in the source text where it found a phrase. We call the referenced text within the source an *anchor*. The system visualizes anchors by rendering them with a blue dotted underline, for example "Laura" in Figure 1. To visually connect extracted text with its source, whenever the mouse or text cursor moves over an anchor RhaiCAL highlights the anchor and any referencing anchors by rendering them with a thicker blue underline, for example "quarterly report" in Figure 1.

Repairing Inferences

RhaiCAL offers the user several methods for easily repairing an agent's inferences that also provide the agent with feedback to improve its performance. First, for each field the agent can suggest multiple options and specify its confidence in each option. For example, the calendar agent recognizes three phrases in the email that might specify the meeting's start time: "2:00 on Friday," "noon," and "4:30." A combo box with the options sorted by confidence value lets the user quickly select the correct one, if present. Anchors within the combo box are active, just like any other anchor, so the user can see from where each value origi-




```
<action type="add">
    <parameter name="object" type="meeting">
        <constraint properties="start, end">
            <inclusive-range>
                <min origin="#1288">2004-07-30 12:00:00 -0400</min>
                <max origin="#D8AB">2004-07-30 16:30:00 -0400</max>
            </inclusive-range>
            <preference>
                <inclusive-range>
                    <min origin="#CC89">2004-07-30 14:00:00 -0400</min>
                    <max>2004-07-30 15:00:00 -0400</max>
                </inclusive-range>
            </preference>
        </constraint>
        <object id="mtg-F408" type="meeting" option="a">
            <property name="start">
                <option>2004-07-30 13:30:00 -0400</option>
            </property>
            <property name="end">
                <option>2004-07-30 14:30:00 -0400</option>
            </property>
            <property name="summary">
                <option confidence="0.8">
                    <span origin="#B7C3">quarterly report</span> -
                    <span origin="#A18D">Laura</span>
                </option>
            </property>
            <property name="location">
                <option origin="#CC89">SRH 531</option>
            </property>
        </object>
    </parameter>
</action>
```

(a) (b) (c) (d)

Figure 2: (a, b) RhaiCAL displays two different options for Paula side-by-side within the context of her calendar. (c) Constraints and preferences. (d) Excerpt from the XML that the agent generates to specify "Option A."

nated. When none of the options have a confidence value above minimum threshold, RhaiCAL draws the user's attention to the problematic field by setting the field's background to pink, for example the "End" field of Figure 1. Second, when the user copy-and-pastes or drag-and-drops from the source text into a field, RhaiCAL creates an anchor in the source text and links the copied text to that anchor. When presented with detailed information such as phone numbers, dates, and hard-to-spell names, we believe that the user would prefer to select choices from a popup menu or to use copy-and-paste or drag-and-drop rather than risk mistyping the information. To help the agent improve its performance through learning, RhaiCAL provides the agent with this explicit feedback, such as the location of user-created anchors and the user's combo box selections.

PROPOSING SCHEDULES

Suppose that after the clarification dialog, the agent understands that Laura requested a one-hour meeting at 2:00 but is also available 12:00–4:30. Now the agent can search for schedules that satisfy the request and Paula's preferences. RhaiCAL visualizes the agent's suggestions within the context of the user's calendar[1] since the user would need to see how the proposal relates to existing entries in their calendar, and thereby relieves the user of the difficulty of mapping natural language into calendar slots.

Paula already has three meetings scheduled (see Figure 2a). At 9:00–11:30 Paula meets with Jerry, an important client, whose meeting cannot be moved or canceled. At 2:30–3:30 Paula meets with Jeff, who had indicated that he is available 12:30–3:30 and 4:30–6:30. At 4:00–4:30 Paula meets with Amy, her boss, whose busy schedule makes rescheduling this meeting difficult. Note that few of these constraints can be explicitly represented in a conventional calendar.

If the agent decides to propose scheduling Laura at 2:30–3:30, RhaiCAL will visualize the proposal as seen in Figure 2a. RhaiCAL uses visual layering to group the different types of objects that it adds to the calendar display. The backmost layer contains objects that embody the agent's understanding of some of the user's preferences. RhaiCAL draws these objects with a light gray diagonal pattern. The 8:00–9:00 and 6:00–7:00 slots shows that Paula prefers not to meet early in the morning or late in evening, and the slot at 12:00–1:00 corresponds to her usual lunch time. The agent knows to avoid proposing meetings during either of these times, if possible. The next layer contains existing appointments in Paula's calendar such as the meetings with Jerry, Jeff, and Amy, which the user is free to manipulate. The front most visual layer (objects with shadows) contains objects that the agent is proposing to add or change: in this case the meeting with Laura. The user can manipulate these objects too.

RhaiCAL also allows the agent to visualize scheduling constraints and preferences of other persons. RhaiCAL visualizes these constraints with vertical bars on the right, labeled with the subject for the associated meeting (see Figure 2c). Laura's preference, to meet 2:00–3:00, is drawn with a thicker bar. Since Paula is free to move the meeting suggested by the agent, these constraints and preferences let Paula know which alternative times are okay for Laura.

If the agent chooses to offer Paula multiple schedules, RhaiCAL will visualize them side-by-side (see Figure 2a & b). The first schedule, "Option A," gives Laura the 1:30–2:30 slot. The second schedule, "Option B," gives Laura her preferred time and bumps Jeff's meeting to 4:30–5:30.

[1] We plan to have RhaiCAL read and write data from a user's real calendar, such as Microsoft Outlook.

XML-BASED LANGUAGE

The XML-based language is a general language that was designed to meet the needs of agents in general, including all the Radar agents. Figure 2d shows an excerpt from the XML used to specify the agent's proposal to add the meeting with Laura in Option A. The `<action>` element specifies that the agent is proposing to add an object to Paula's calendar. In general, an action can specify multiple parameters. The `<object>` element specifies the meeting to be added, which consists of a sequence of property-value pairs. RhaiCAL comes with a default schema that defines objects pertinent for calendaring. The RhaiCAL language also allows agents to provide schemas for defining their own objects. Each `<property>` element can contain one or more `<option>` elements. The "start" property has a single option with an `origin` attribute that references an anchor within Laura's email containing the source of the start time value. The value of the attribute is a machine generated identifier that has no semantic meaning. None of the `<property>` elements in this example have multiple `<option>` elements, because the agent has already clarified its interpretation of the message using a dialog such as in Figure 1, but the language allows an agent to specify multiple options for a property. The `<constraint>` element specifies that the start and end properties are constrained. The `<inclusive-range>` element specifies the minimum and maximum value that the agent thinks that the properties should take based upon its understanding of Laura's request. The `<preference>` element similarly specifies bounds on Laura's preferred time.

RELATED WORK

The Visual Scheduler [1] and Time Lattice [5] shared calendars help a meeting organizer identify common free times across multiple people's calendars. However, neither of these systems explicitly represents the constraints and preferences that can help a meeting organizer schedule a meeting when the problem is over-constrained. Begole *et al.* developed a system to monitor computer activity for the purpose of creating visualizations of temporal work patterns [2]. This system could automatically place a user's time-of-day meeting availability within a shared calendar. Recent work by Kristjansson *et at.* has focused on filling-in natural language clarification forms in which as the user selects a value for a field, the system can eliminate choices for other fields due to dependencies amongst the fields [4].

FUTURE DESIGN

Several issues still remain. First, by placing the interaction within a user's application, we allow them to change other appointments that are not related to the agent's proposal. How is the system to know when the user has finished responding? One could add an explicit "OK" button. But what if the user moves on forgetting to explicitly end the dialog? Collagen [11] provides toolkit-level support for determining how user actions fit into an overall plan, which we might incorporate into our system. Second, we plan to design interfaces that let the agent explain why it made certain decisions, though some AI algorithms are more amenable to explanation than others. Finally, we plan a large scale deployment of RhaiCAL as part of Radar that will allow us to test its usability and usefulness in the context of real use.

CONCLUSION

The RhaiCAL system provides novel visualizations and interaction techniques that allow users to interact with calendar scheduling agents. RhaiCAL provides interaction techniques that allow the user to easily correct inference errors made by the agent and returns rich feedback about such corrections to the agent. RhaiCAL allows agents to propose multiple parameters for actions and to specify the constraints and preferences on those parameter values. This support for flexibility helps the user even when the agent is uncertain about the correct suggestion to make. RhaiCAL helps the user more easily see what an agent is doing, which should increase the user's confidence in the agent.

ACKNOWLEDGMENTS

The authors thank Ellen Ayoob, Andrew Ko, Jay Modi, Jeffrey Nichols, Jean Oh, Jeff Pierce, Bill Scherlis, Desney Tan, Aaron Spaulding, Anthony Tomasic, and John Zimmerman. Andrew Faulring is supported by a NSF Graduate Research Fellowship. 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 DARPA or the Department of Interior-National Business Center (DOI-NBC).

REFERENCES

1. Beard, D., *et al.* A Visual Calendar for Scheduling Group Meetings. *Proc. CSCW 1990*. ACM Press (1990). 279–290.
2. Begole, J.B., *et al.* Work Rhythms: Analyzing Visualizations of Awareness Histories of Distributed Groups. *Proc. CSCW 2002*. ACM Press (2002). 334–343.
3. Kozierok, R. and Maes, P. A Learning Interface Agent for Scheduling Meetings. *Proc. IUI 1993*. ACM Press (1993). 81–88.
4. Kristjansson, T., *et al.* Interactive Information Extraction with Constrained Conditional Random Fields. *Proc. NCAI 2004*. AAAI Press (2004). 412–418.
5. Mackinlay, J.D., *et al.* Developing Calendar Visualizers for the Information Visualizer. *Proc. UIST 1994*. ACM Press (1994). 109–118.
6. Mitchell, T.M., *et al.* Experience with a Learning Personal Assistant. *CACM 37,* 7 (1994), 81–90.
7. Modi, P.J., *et al.* CMRadar: A Personal Assistant Agent for Calendar Management. *Proc. AOIS 2004*. (2004). 134–148.
8. Mueller, E.T. A Calendar with Common Sense. *Proc. IUI 2000*. ACM Press (2000). 198–201.
9. Oh, J. and Smith, S.F. Learning User Preferences in Distributed Calendar Scheduling. *Proc. PATAT 2004*. (2004). 35–50.
10. Palen, L. Social, Individual and Technological Issues for Groupware Calendar Systems. *Proc. CHI 1999*. ACM Press (1999). 17–24.
11. Rich, C. and Sidner, C.L. COLLAGEN: When Agents Collaborate with People. *Proc. AGENTS 1997*. ACM Press (1997). 284–291.
12. Shneiderman, B. and Maes, P. Direct Manipulation vs. Interface Agents. *interactions 4,* 6 (1997), 42–61.
13. Tomasic, A., *et al.* Learning to Navigate Web Forms. *Proc. IIWEB 2004*. (2004). 27–32.