Principles of Mixed-Initiative User Interfaces

Eric Horvitz
Microsoft Research
Redmond, WA 98025 USA
+1 425 936 2127
horvitz@microsoft.com

ABSTRACT
Recent debate has centered on the relative promise of focusing user-interface research on developing new metaphors and tools that enhance users' abilities to directly manipulate objects *versus* directing effort toward developing interface agents that provide automation. In this paper, we review principles that show promise for allowing engineers to enhance human—computer interaction through an elegant coupling of automated services with direct manipulation. Key ideas will be highlighted in terms of the LookOut system for scheduling and meeting management.

Keywords
Intelligent agents, direct manipulation, user modeling, probability, decision theory, UI design

INTRODUCTION
There has been debate among researchers about where great opportunities lay for innovating in the realm of human—computer interaction [10]. One group of researchers has expressed enthusiasm for the development and application of new kinds of automated services, often referred to as interface "agents." The efforts of this group center on building machinery for sensing a user's activity and taking automated actions [4,5,6,8,9]. Other researchers have suggested that effort focused on automation might be better expended on exploring new kinds of metaphors and conventions that enhance a user's ability to *directly manipulate* interfaces to access information and invoke services [1,13]. Innovations on both fronts have been fast paced. However, there has been a tendency for a divergence of interests and methodologies versus focused attempts to leverage innovations in both arenas.

We have pursued principles that provide a foundation for integrating research in direct manipulation with work on interface agents. Our goal is to avoid focusing solely on one tack or the other, but to seek valuable synergies between the two areas of investigation. Surely, we should avoid building complex reasoning machinery to patch fundamentally poor designs and metaphors. Likewise, we wish to avoid limiting designs for human–computer interaction to direct manipulation when significant power and efficiencies can be gained with automated reasoning. There is great opportunity for designing innovative user interfaces, and new human–computer interaction modalities by considering, from the ground up, designs that take advantage of the power of direct manipulation and potentially valuable automated reasoning [2].

PRINCIPLES FOR MIXED-INITIATIVE UI
Key problems with the use of agents in interfaces include poor guessing about the goals and needs of users, inadequate consideration of the costs and benefits of automated action, poor timing of action, and inadequate attention to opportunities that allow a user to guide the invocation of automated services and to refine potentially suboptimal results of automated analyses. In particular, little effort has been expended on designing for a *mixed-initiative* approach to solving a user's problems—where we assume that intelligent services and users may often collaborate efficiently to achieve the user's goals.

Critical factors for the effective integration of automated services with direct manipulation interfaces include:

(1) Developing significant value-added automation. It is important to provide automated services that provide *genuine value* over solutions attainable with direct manipulation.

(2) Considering uncertainty about a user's goals. Computers are often uncertain about the goals and current the focus of attention of a user. In many cases, systems can benefit by employing machinery for inferring and exploiting the uncertainty about a user's intentions and focus.

(3) Considering the status of a user's attention in the timing of services. The nature and timing of automated services and alerts can be a critical factor in the costs and benefits of actions. Agents should employ models of the attention of users and consider the *costs and benefits of deferring action* to a time when action will be less distracting.

(4) Inferring ideal action in light of costs, benefits, and uncertainties. Automated actions taken under uncertainty in a user's goals and attention are associated with context-dependent costs and benefits.

The value of automated services can be enhanced by guiding their invocation with a consideration of the *expected value of taking actions*.

(5) Employing dialog to resolve key uncertainties. If a system is uncertain about a user's intentions, it should be able to engage in an efficient dialog with the user, *considering the costs* of potentially bothering a user needlessly.

(6) Allowing efficient direct invocation and termination. A system operating under uncertainty will sometimes make poor decisions about invoking—or not invoking—an automated service. The value of agents providing automated services can be enhanced by providing efficient means by which users can directly invoke or terminate the automated services.

(7) Minimizing the cost of poor guesses about action and timing. Designs for services and alerts should be undertaken with an eye to *minimizing the cost of poor guesses*, including appropriate timing out and natural gestures for rejecting attempts at service.

(8) Scoping precision of service to match uncertainty, variation in goals. We can enhance the value of automation by giving agents the ability to *gracefully degrade* the precision of service to match current uncertainty. A preference for "doing less" but doing it correctly under uncertainty can provide user's with a valuable advance towards a solution and minimize the need for costly undoing or backtracking.

(9) Providing mechanisms for efficient agent–user collaboration to refine results. We should design agents with the assumption that users may often wish to complete or refine an analysis provided by an agent.

(10) Employing socially appropriate behaviors for agent–user interaction. An agent should be endowed with tasteful default behaviors and courtesies that match *social expectations* for a benevolent assistant.

(11) Maintaining working memory of recent interactions. Systems should maintain a memory of recent interactions with users and provide mechanisms that allow users to make efficient and natural references to objects and services included in "shared" short-term experiences.

(12) Continuing to learn by observing. Automated services should be endowed with the ability to continue to become better at working with users by continuing to learn about a user's goals and needs.

A TESTBED FOR MIXED-INITIATIVE UI

The LookOut project has focused on investigating issues with overlaying automated scheduling services on Microsoft Outlook, a largely direct-manipulation based messaging and scheduling system. LookOut automation identifies new messages that are opened and brought to focus and attempts to assist users with reviewing their calendar and with composing appointments.

Value-Added Service: Calendaring and Scheduling

When invoked, LookOut parses the text in the body and subject of an email message in focus and attempts to identify a date and time of associated with an event implied by the sender. The system then invokes Outlook's calendaring subsystem, brings up the user's online appointment book, and attempts to fill in relevant fields of an appointment record. The system displays its guesses to the user and allows the user to edit its guesses and to save the final result.

LookOut's scheduling analysis centers on a goal-specific parsing of the text contained in the email message that has focus. The system notes when a new message is being read, or when a message comes into focus that has not yet been analyzed. The system first establishes the date a message was sent as an *anchor date* and attempts to normalize its view based on the composition date. For example, if a message was written yesterday and contains text referring to scheduling a meeting for "tomorrow," the system will understand that the message is referring to "today."

If LookOut cannot identify an implied date and time, the system degrades its goal to identifying a span of time that is most relevant given the text of the message (i.e., a specific day, week, or month), and then displays a scoped view of the calendar to the user. The user can directly manipulate the proposed view and, if appropriate, go on to schedule appointments manually.

LookOut has knowledge about typical patterns of expression in email about meetings and times. Beyond understanding the variety of ways that people refer to dates and times, the system understands the temporal implications of suggestions about information in email messages about holding meetings at various times in the future (*e.g.*, "sometime tomorrow," "later in the week," "next week," "within a couple of weeks," "in May," etc.), at prototypical times during the day (*e.g.*, "morning," "afternoon," and "evening"), as well as during typical recurrent events (*e.g.*, "at breakfast," "grab lunch," and "meet for dinner," etc.).

LookOut analysis reduces the number of interactions and complexity of navigation required of the user. Without LookOut, users must navigate to the appropriate graphical button or menu item to open their calendar, search for the appropriate day, input the appropriate times and fill in the subject of the meeting. LookOut performs this operation automatically or via a single interaction, depending on the modality selected. Even when LookOut guesses incorrectly, the user is placed in an approximately correct position in the calendar and can refine an approximate guess about the implied appointment.

Decision Making Under Uncertainty

Users can directly invoke LookOut by clicking on an icon that is always present on the system tray of the Microsoft

Windows shell. However, the system also works to automatically identify a user's goals by considering the content of messages being reviewed. LookOut processes the header, subject, and body of the message and, based on this information, assigns a probability that a user would like to view the calendar or schedule an appointment, by employing a probabilistic classification system that is trained by watching the user working with email. The system makes decisions about appropriate actions as a function of an inferred probability that the user has a goal of performing scheduling and calendaring operations. In particular, the inferred probability that service is desired is used by LookOut to make a decision about whether to apply a second phase of analysis that provides the user with automated calendaring and scheduling.

Depending on the inferred probability—and on an assessment of the expected costs and benefits of action—the system decides to either (1) do nothing but simply wait for continued direct manipulation of Outlook or manual invocation of LookOut, (2) to engage the user in a dialog about his or her intentions with regards to providing a service, or (3) to go ahead and attempts to provide its service by invoking its second phase of analysis.

Multiple Interaction Modalities

LookOut can be configured to be operated in a solely manual modality or can be placed in one of several automated-assistance modalities. In manual operation, the system will only take action if a user clicks on the small LookOut icon appearing in the system. When invoked, LookOut analyzes the email message that has system focus. Users can tell the system to display an alerting symbol (red check mark) on the system-tray icon when LookOut would have taken action if it had been in an automated-assistance modality. By hovering the cursor over the icon on the system tray, a summary of the intended action appears. Figure 1 displays the direct invocation of LookOut. As shown in the figure, a menu with dynamically populated options pops up, letting the user schedule or organize a meeting, or schedule from text on the system clipboard.

When placed in a basic automated-assistance mode, LookOut works by launching and populating fields in Outlook windows. In this mode, the system also employs traditional dialog boxes to request additional information from users when appropriate. LookOut can also operate in a *social-agent modality* projecting an explicit social presence in the form of animated characters, drawn from the MS Agent social user-interface package. When in this mode, the system issues queries to users and announces the results of analyses with an anthropomorphic presence.

When LookOut is in the social-agent modality, it operates in a handsfree manner, establishing an audio channel for interacting with LookOut, further reducing mouse and keyboard interaction with the Outlook system. In the handsfree mode, the system employs a text-to-speech (TTS)

Figure 1. Manual invocation of LookOut. By hovering a cursor over the LookOut icon, a user can examine LookOut's guess. By clicking on the LookOut icon on the system tray, the user invokes the appointment service.

system and automated speech recognition system developed by Microsoft Research to engage users in a natural dialog about their intentions. If LookOut is confident enough in its assessment of a user's goals, a character appears and mentions that it has readied a calendar view to show the user or has created a tentative appointment before displaying the results. At lower levels of confidence, LookOut inquires about a user's interest in either seeing the calendar or scheduling an appointment, depending on the system's analysis of the message being viewed. After asking the user, the system listens for an answer without requiring additional keys or buttons to be pressed.

Figure 2 displays a sequence of screens demonstrating LookOut's operation within the social-agent modality. After a message is analyzed behind the scenes, the system decides it is worthwhile to engage the user in a dialog about creating an appointment. An animated assistant appears and engages the user with speech (a text balloon option is turned on in this case to relay the content of the speech with text). The user can indicate via speech that an appointment is desired with one of several natural acknowledgments, including "yes," "yeah," "sure," "do it." Given a go ahead, LookOut creates an appointment and reviews it with the user with text-to-speech, before evaporating, leaving the result behind for refinement and saving. If the user had expressed disinterest in going ahead with the appointment by simply closing the message or by responding with a variety of natural phrases including "no," "not now," "nah," and "go away," the agent would have immediately nodded to confirm an understanding and disappear.

LookOut dynamically scopes the calendar view to its best guess, given uncertainty or indications about an appropriate view from the message text. For the case captured in Figure 3, LookOut cannot confidently identify a specific

time and day. Rather than making a poor guess, LookOut brings up an appropriate week view on the user's calendar.

Handling Invocation Failures

As LookOut is expressly continuing to reason under uncertainty about the value of taking action, or engaging the user in a dialog as messages are opened and closed, the system can make guesses that simply turn out to be wrong. If the LookOut system fails to automatically infer that users wish to see their calendar or schedule an appointment, the system can be directly invoked by clicking on the LookOut icon on the system tray. If LookOut guesses that it is worthwhile to engage the user in a dialog about scheduling but the user is busy or disinterested in interacting with the service, the system will pose a question, wait patiently for a response, and then make a respectful, apologetic gesture and evaporate. The amount of time the system waits before timing out is a function of the inferred probability that a user desires the service. Also, the system increases its dwell on the desktop if it detects signs that the user is thinking, including "hmmm…", "uh…," etc. The design of LookOut's behaviors for handling delays with responses and for reacting to signs that service is being declined was guided by the goal of giving LookOut the sensibility of an intuitive, courteous butler, who might make potentially valuable suggestions from time to time, but who is careful to note when the user is simply too busy to even respond—and to get out of the user's way with minimal disturbance.

INFERRING BELIEFS ABOUT A USER'S GOALS

If we wish to assist users with potentially complex services, it can be valuable to consider how such automation can be provided effectively in light of the uncertainties agents may have about users goals. Thus, developing machinery that endows a system with the ability to explicitly assign likelihoods to different feasible user intentions can be critical in mixed-initiative systems. Such machinery can extend from sets of rules linked to tables of probabilistic information to more complex, real-time inference.

In related work in user modeling, probabilistic models of a user's goals have been employed to continue to perform real-time inference about the probability of alternate feasible goals as a function of observables including the current program context, a user's sequence of actions and choice of words used in a query [4,6]. Some of this work has leveraged recent successes in building and reasoning with Bayesian network models [7,11].

LookOut leverages work in automated text classification for making decisions about actions. Alternate text classification methodologies were explored, including a naïve Bayesian text classifier and text classification based on the Support Vector Machine (SVM) analysis [3]. The current version of LookOut assigns probabilities of user intention by employing an SVM text classification based on an efficient linear SVM approximation method developed by Platt [12]. The method was coupled with a methodology

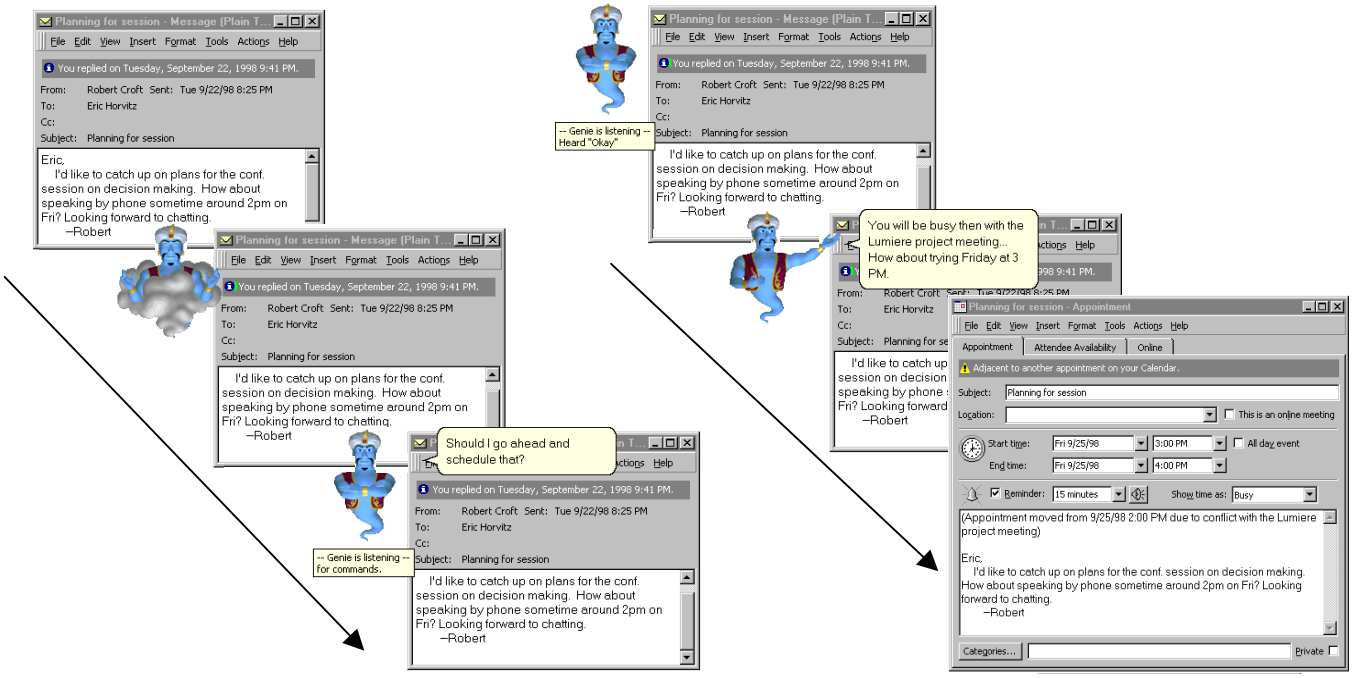

Figure 2. LookOut sequence showing its operation in its explicit social-agent modality. A new message (top left) is analyzed and a decision is made to engage the user in a dialog (left). After receiving confirmation via speech input, the system creates an appointment and presents its guess to the user for refinement (right).

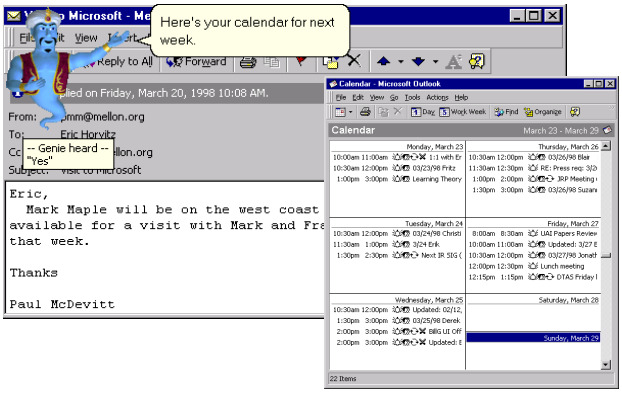

Figure 3. Automated scoping of calendar. If LookOut cannot establish a specific day and time, it attempts to select a most appropriate span of time to display to the user for review or refinement through direct manipulation.

for including custom-tailored, task-specific sets of text features. Rather than employing text classification in the classical manner for tasks such as labeling or categorizing documents, we harness the methods for learning and reasoning about the likelihood of user goals or tasks within a context. For the assumed context of a user reviewing email, we wish to assign a likelihood that an email message that has just received the focus of attention is in the *goal category* of *"User will wish to schedule or review a calendar for this email"* versus the goal category of "*User will not wish to schedule or review a calendar for this email"* based on the content of the messages.

A linear SVM text classifier is built by training the system on a set of messages that are calendar relevant and calendar irrelevant. At runtime, for each email message being reviewed, the linear SVM approximation procedure outputs the likelihood that the user will wish to bring up a calendar or schedule an appointment. The current version of LookOut was trained initially on approximately 1000 messages, divided into 500 messages in the relevant and 500 irrelevant messages.

FROM BELIEFS TO ACTIONS

Given uncertainties about a user's goals, what automated actions should be taken? We shall consider the case of a decision about whether or not to invoke the services performed by an intelligent agent. From the perspective of decision theory, decisions about action versus inaction should be directed by *expected utility*. Autonomous actions should be taken only when an agent believes that they will have greater expected value than inaction for the user, taking into consideration the costs, benefits, and uncertainties in the user's goals.

	Desired Goal	Not Desired
Action	$u(A,G)$	$u(A,\neg G)$
No Action	$u(\neg A,G)$	$u(\neg A,\neg G)$

Table 1. Four outcomes considered in decisions about whether to engage an intelligent agent to provide service.

Actions, Intentions, and Outcomes

Let us assume an agent has access to inference about the likelihood of a user's goals given observed evidence, written $p(G|E)$. In LookOut, the probability that a user wishes to schedule is computed from evidence in patterns of text contained in a message that has been recently opened or brought to focus.

For decisions about action versus inaction, we must consider four deterministic outcomes: Either the user indeed has the goal being considered or does not have the goal and, for each of these states of user intention, the system either can take an action or not take the action. We map a measure of the value associated with each outcome to a *utility* on a zero to one scale, and define utilities as follows:

- $u(A,G)$: the utility of taking action A when goal G is true
- $u(A,\neg G)$: the utility of taking action A when goal G is not true
- $u(\neg A,G)$: the utility of not taking action A when goal G is true
- $u(\neg A,\neg G)$: the utility of not taking action A when goal G is not true

These outcomes are summarized in Table 1.

The expected utility of taking autonomous action to assist the user with an action given observed evidence, $eu(A|E)$, is computed by combining the utilities of the outcomes for the case where the user desires service and does not desire a service, weighted by the probability of each outcome, as follows:

$$eu(A|E)=p(G|E)u(A,G) + p(\neg G|E)\, u(A,\neg G) \qquad (1)$$

We can rewrite this equation in terms of $p(G|E)$, by noting that $p(G|E)=1-p(\neg G|E)$. Thus, the expected utility of providing autonomous service is,

$$eu(A|E)=p(G|E)u(A,G) + [1-p(G|E)]\, u(A,\neg G) \qquad (2)$$

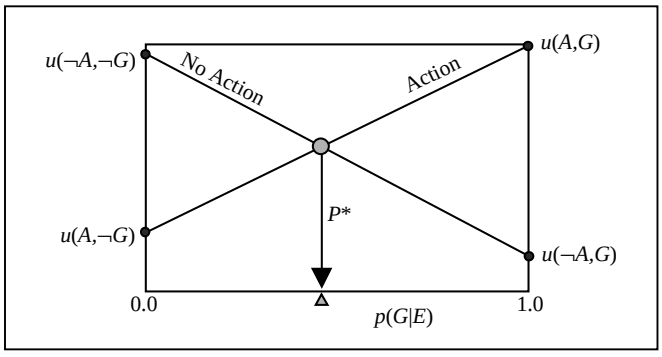

Figure 4. Graphical analysis of the expected utility of action versus inaction, yielding a threshold probability for action.

The expected utility of *not* taking autonomous action to assist the user, $u(\neg A|E)$, is

$$eu(\neg A|E)=p(G|E)u(\neg A,G) + [1-p(G|E)]u(\neg A,\neg G) \quad (3)$$

We can visualize the implications of these equations by plotting the expected utility as a function of probability. Figure 4 displays a graph where the horizontal represents the probability the user has a goal, ranging from zero to one. The vertical axis indicates the expected value of the system's response. The two outcomes displayed on the right vertical axis have an expected utility associated with $p(G|E)$=1.0—the user indeed having the goal under consideration. The outcomes listed on the left vertical axis indicate the value of the outcomes when $p(G|E)$=0. The expected value of acting for intermediary probabilities of $p(G|E)$, as dictated by Equation 2, is a line joining the two deterministic outcomes associated with taking action. The expected value of not acting as dictated by Equation 3 is a similar line joining the two outcomes associated with inaction.

Expected Utility and Thresholds for Agent Action

The lines representing expected utility cross at a specific inferred probability of the user having a goal. At this threshold probability, referred to as p^*, the expected value of action and inaction are equal. The best decision to make at any value of $p(G|E)$ is the action associated with the greatest expected utility at that likelihood of the user having the goal. By inspecting the graph, it is easy to see that it is best for the system to take action if the probability of a goal is greater than p^* and to refrain from acting if the probability is less than p^*.

The threshold probability can be computed for any four utilities by setting Equations 2 and 3 equal to one another and solving for $p(G|E)$. Given four utilities associated with the four outcomes of interest, a system needs only to check whether the probability of the goal is greater or less than such a threshold probability to decide on whether it is in the best interest of the user to invoke a service.

The threshold probability, p^*, can be influenced by context-dependent changes of the utilities associated with one or more of the outcomes. For example, the utility, $u(A,\neg G)$, associated with the situation where a system takes action when a goal is not desired, can be significantly influenced by the status of a user's attention. The utility of unwanted action can diminish significantly with increases in the depth of a user's focus on another task. Such a reduction in the value of action leads to a higher probability threshold. In contrast, the utility, $u(A,\neg G)$, associated with the situation where a system takes action when a goal is not desired, might be greater when more screen real estate is made available. Increased screen real estate can diminish the perceived cost of the needless operation of a scheduling service that might bring up an appointment that obscures items at a user's focus of attention. As another example of

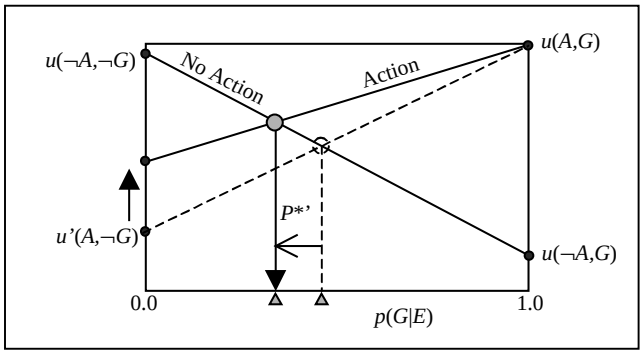

Figure 5. The result of increasing the value of taking erroneous action. Context-dependent shifts in any of the utilities can change the probability threshold for action.

a context-dependent outcome, the utility, $u(\neg A,G)$, representing the situation where a system does not take action when a user indeed has the goal, may decrease as a user becomes more rushed. Diminishing the value of this action reduces the threshold probability for action.

Figure 5 displays geometrically how p^* can change with context. In this case, increasing the utility (decreasing the cost) of outcome $u(A,\neg G)$ of acting when service is not desired leads to a lowering of the threshold probability that must be crossed before action occurs.

Dialog as an Option for Action

Beyond reasoning about whether to act or not to assist a user with an autonomous service, we can also consider the action of asking users about their goals. We can integrate action for dialog into the expected utility framework by considering the expected value of asking the user a question. We now consider the utility of two additional outcomes: the case where an agent initiates dialog about a goal and the user actually desires the goal under consideration, $u(D,G)$, and the case where the user does not have the goal, $u(D,\neg G)$. We compute the expected utility of performing dialog under uncertainty with an equation analogous to Equation 3.

Figure 5 displays a graph with the addition of a line representing the expected utility of engaging in a dialog. As highlighted in the graph, the utility of engaging in a dialog with a user when the user does not have the goal in question is typically greater than the utility of performing an action when the goal is not desired. However, the utility of *asking* a user before performing a desired action is typically smaller than the utility of simply performing a desired action when the user indeed has the goal. In such circumstances, if we follow the rule of selecting the option with the greatest expected utility, we see that action can be guided by two new threshold probabilities: the threshold between inaction and dialog, $p^*_{\neg A,D}$, and the threshold between dialog and action, $p^*_{D,A}$. These two thresholds provide an instant index into whether to act, to engage the

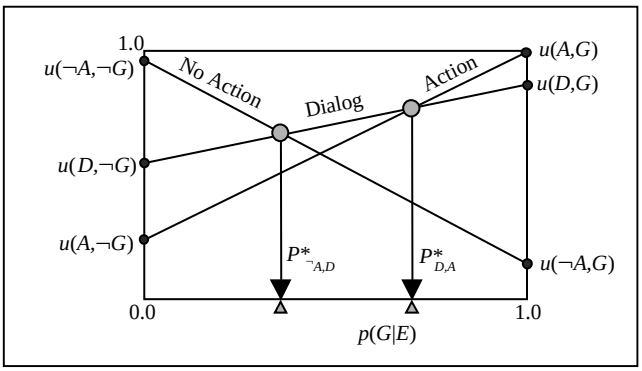

Figure 6. Adding a second action option consisting of dialog with users about their goals. In this case, the graphical analysis higlights the origin of two threshold probabilities for guiding the action of autonomous services.

user in a dialog about action, or to do nothing, depending on the assessed likelihood of the user having a goal.

Systems for guiding autonomous service do not necessarily need to perform explicit computations of expected value. Thresholds can be directly assessed by designers or users. Such directly assessed thresholds for action imply a deeper implicitly assumed expected-utility model.

The LookOut system employs default utilities for guiding dialog and action. However, the system also allows users to specify the utilities for outcomes. Given a set of assessed utilities, the system computes and uses modified threshold probabilities. LookOut also allows users to simply specify two key threshold probabilities for controlling dialog and action. At run time, LookOut considers whether the inferred probability that users desires service is above threshold for dialog or action, versus being consistent with inaction.

USER ATTENTION AND THE TIMING OF SERVICE

Automated activity occurring before a user is ready or open for the service can be distracting. On the other hand, delays in the provision of service can diminish the value of automation. We have found that the value of services and alerts can be enhanced through building and applying models of attention that consider the temporal pattern of a user's focus of attention.

Given the potential value of approaching users with dialog or actions when users are most ready for a service, we performed studies to identify the most appropriate timing of messaging services as a function of the nature of the message being reviewed by the user. We added instrumentation to LookOut to monitor the length of time between the review of messages and the manual invocation of messaging services and collected data from several users with a goal of building a default temporal-centric model of attention. We identified a nonlinear relationship between the size of the message being reviewed and the amount of time users prefer to dwell on the content of the messages before accepting automated calendaring and scheduling operations. We found that the relationship between message size and the preferred time for deferring offers of service can be approximated by a sigmoid function as represented by the sample data from a user displayed in Figure 7. Continuing studies on timing within the LookOut project are aimed at examining other factors that can explain dwell time including ambiguity and complexity of dates and times mentioned in the message.

In the general case, we can construct a model of attention from such timing studies and make the utility of outcomes time-dependent functions of message length. Alternatively, we can use timing information separately to defer service until a user is likely ready to receive it.

The current version of LookOut employs a predetermined default automated-service timing model based on user studies. However, the system can also be instructed to build a custom-tailored timing model by watching a user interacting with email. The system records the size of each message being reviewed and the amount of time spent on each message before scheduling operations are invoked and stores cases when it is used in a user-directed manner. When the system enters a learning mode, the system performs a regression analysis on the data and fits a piecewise linear model to the data. Alternatively, users can tell the system to delay for a fixed amount of time before the service is invoked.

MACHINERY FOR LIFE-LONG LEARNING

LookOut contains a pretrained probabilistic user model and timing model. However, the system is designed to continue to learn from users. Methods for embedding the capability for life-long learning is a key challenge in Artificial Intelligence research [14]. LookOut continues to store messages as calendar relevant and irrelevant, by watching the user working with email. If a calendar or scheduling facility is invoked within a predetermined time horizon, the system saves the message as schedule-relevant. The system also continues to record the time users dwell on schedule-relevant messages before invoking a calendaring operation.

User can specify a policy for continual learning. Users can dictate a training schedule that guides the learning component of the system periodically to incrementally refine the probabilistic user model and time-based attention model. The ongoing model continues to hone the models used for guessing about the relevance of the automated scheduling services as well as to become a better estimator of the best time to invoke the services.

SUMMARY AND CONCLUSIONS

We reviewed key challenges and opportunities for building *mixed-initiative* user interfaces—interfaces that enable users and intelligent agents to collaborate efficiently. We first presented a set of principles for designing mixed-initiative user interfaces that address systematic problems with the use of agents that may often have to guess about a user's needs. Then, we focused on methods for managing the uncertainties that agents may have about users' goals

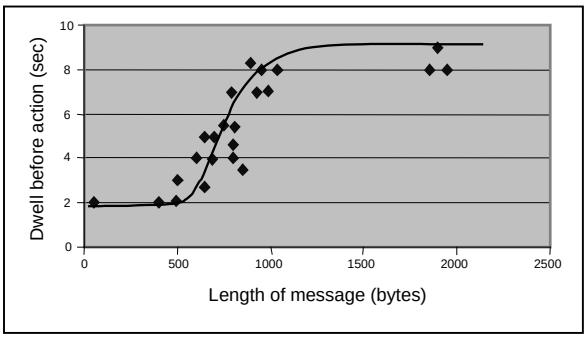

Figure 7. Sigmoid fit on sample of data from a user displaying the relationship between dwell time on schedule-relevant messages and the quantity of text in the message being reviewed.

the uncertainties that agents may have about users' goals and focus of attention. We discussed the consideration of uncertainty, as well as the expected costs and benefits of taking autonomous action in different situations. We highlighted methods and design principles with examples drawn from the LookOut system. Research on LookOut has elucidated difficult challenges and promising opportunities for improving human–computer interaction through the elegant combination of reasoning machinery and direct manipulation. We believe continuing efforts to address problems with the design of mixed-initiative user interfaces will likely yield fundamental enhancements in human–computer interaction.

ACKNOWLEDGMENTS

Andy Jacobs has served as the primary software engineer on the LookOut prototype. John Platt developed the linear SVM text-classification methodology used in the current version of LookOut. Mehran Sahami assisted with the early studies of the use of Bayesian text classification for identifying schedule-relevant messages. Jack Breese, Mary Czerwinski, Susan Dumais, Bill Gates, Ken Hinckley, Dan Ling, and Rick Rashid provided valuable feedback on this research.

REFERENCES

1. Ahlberg, C., and Schneiderman, B. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. *Proceedings of CHI '94 Human Factors in Computing Systems* (April 1994) ACM, 313-317.

2. Birnbaum, L., Horvitz, E., Kurlander, D., Lieberman, H., Marks, J. Roth, S. Compelling Intelligent User Interfaces: How Much AI? In *Proceedings of the 1997 ACM International Conference on Intelligent Interfaces* (Orlando, FL, January 1996).

   http://www.merl.com/reports/TR96-28/index.html

3. Dumais, S. T., Platt, J., Heckerman, D., and Sahami, M., Inductive learning algorithms and representations for text categorization. *Proceedings of CIKM98.* (Bethesda MD, November 1998). ACM Press, 148-155

4. Heckerman, D., and Horvitz, E. Inferring Informational Goals from Free-Text Queries: A Bayesian Approach, *Fourteenth Conference on Uncertainty in Artificial Intelligence* (Madison WI, July 1998), Morgan Kaufmann Publishers, 230-237.

   http://research.microsoft.com/~horvitz/aw.htm

5. Horvitz, E., and Barry, M. Display of Information for Time-Critical Decision Making. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (Montreal, August 1995). Morgan Kaufmann Publishers, 296-305.

   http: //research.microsoft.com/~horvitz/vista.htm

6. Horvitz, E., Breese, J., Heckerman, D., Hovel, D., Rommelse, D. The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users, *Fourteenth Conference on Uncertainty in Artificial Intelligence* (Madison WI, July 1998). Morgan Kaufmann Publishers, 256-265.

   http: //research.microsoft.com/~horvitz/lumiere.htm

7. Horvitz, E.J., Breese, J., and Henrion, M. Decision theory in Expert Systems and Artificial Intelligence. *International Journal of Approximate Reasoning*, Special Issue on Uncertainty in Artificial Intelligence, 2:247-30.

   http: //research.microsoft.com/~horvitz/dt.htm

8. Lieberman, H., Letizia: An Agent That Assists Web Browsing, *International Joint Conference on Artificial Intelligence* (Montreal, August 1995). IJCAI.

9. Maes, P. Agents that Reduce Work and Information Overload. *Commun. ACM* 37,7, 31-40

10. Maes, P., and Schneiderman, B., Direct Manipulation vs. Interface Agents: A Debate. *Interactions*, Vol. IV Number 6, ACM Press, 1997.

11. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers: San Francisco, 1991.

12. Platt, J. Fast training of SVMs using sequential minimal optimization. To appear in: B. Scholkopf, C. Burges, and A. Smola (Eds.) *Advances in Kernel Methods – Support Vector Learning*, MIT Press, 1999.

13. Schneiderman, B. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, ACM Press. 1992.

14. Selman, B. Brooks, R.A., Dean, T., Horvitz, E., M. Mitchell, T., Nilsson, N.J. Challenge Problems for Artificial Intelligence, In: *Proceedings of AAAI-96, Thirteenth National Conference on Artificial Intelligence* (Portland, OR, August 1996). AAAI Press, 1340-1345.