Intelligent User Interfaces

Artificial Intelligence Meets Human Computer Interaction

For:
Prof. Balakrishnan
TA: Michael McGuffin
CSC 2514F

By:
Ben Marlin
Marlin[at]cs.toronto.edu

Introduction

The field of Intelligent User Interface design is often loosely described as the intersection of the fields of Artificial Intelligence and Human Computer Interaction. Defining exactly what is meant by the term “Intelligent User Interface” is a matter of some contention within the research community. According to Maybury and Wahlster, IUI’s are “human-machine interfaces that aim to improve the efficiency, effectiveness, and naturalness of human-machine interaction by representing, reasoning, and acting on models of the user, domain, task, discourse, and media” (Maybury & Wahlster, 2). The problem with such descriptions is that they elude to specific artificial intelligence techniques and as Waern writes “What, exactly, counts as an intelligent technique will vary over time” (Waern). On the other hand Sullivan and Tyler admit to the difficulties that arise in trying to define exactly what an IUI encompasses, and how one might enumerate those particular qualities that render an interface “intelligent”. To circumvent this problem they describe an IUI as any interface that results from the application of artificial intelligence techniques to problems of human-computer interface design (Sullivan,1-3). In the present paper we will adopt the more general view of what constitutes IUI research held by Sullivan and Tyler.

Under this broad view of IUI research Sullivan and Tyler consider the roots of the field to be a classic area of Artificial Intelligence known as natural language processing (NLP). NLP research has a long and interesting history stretching back to the 1940’s. During the 1950’s and 1960’s the physical interface to a computing system was limited to text-based teletype machines or display screens, and as a result the only interface metaphor available was natural language-like interaction (Sullivan & Tyler, 2). One of the goals of NLP research was to enable users to interact with computer systems in a fully natural language such as English instead of a rigidly defined command language. Allen Turing outlined the ultimate achievement for such an NLP system in 1950 when he proposed the now infamous *Turing Test* as the measure of a computer system’s intelligence (Russell & Norvig, 5).

Although IUI research may arguably have begun around the time the Turing Test was proposed, we will focus on more modern IUI research. Modern IUI research is characterized by a shift in focus away from NLP research as a result of the introduction of graphical user interfaces and multimedia systems beginning in the mid 1980’s (Sullivan & Tyler, 3). We will define modern IUI research as involving the application of AI techniques to issues of graphical and multimedia user interface design.

Much of the work on intelligent user interfaces conducted to date has been focused on the underlying artificial intelligence techniques. Often little consideration is given to ideas from human-computer interaction, and little is written on IUI design from an HCI perspective. In this paper we present IUI research in terms of three popular design approaches that are differentiated by the style of interaction between the AI components and the user. These are the integrated approach, the agent-based approach, and the mixed Initiative approach. We will discuss each approach, highlighting the main design principles through the examination of case studies drawn from recent research efforts. We will then compare the approaches to elucidate their relative strengths and weaknesses.

The Integrated Approach

As indicated in Figure 1, the integrated approach to IUI design is characterized by an integrated AI module that that can obtain status information from the application interface, as well as modify or adapt the interface in some way. The user interacts with the application through the application's graphical interface as they normally would. An important feature of this approach is that there is no direct interaction between the AI module and the user. The AI module must determine the user's needs and goals only through the information provided by the application. This may include interface status information such as the type of widget that is currently in focus, or atomic interface events such as mouse movement and keystrokes.

Integrated Approach

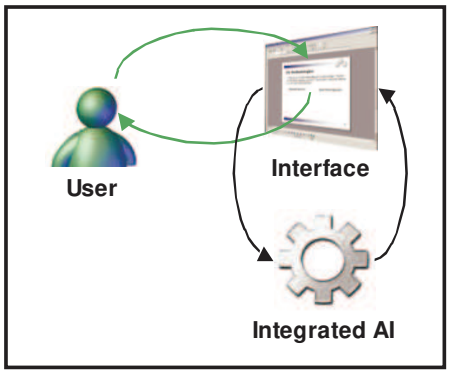

Figure 1: Diagram of integrated approach to IUI design.

Examples of this type of approach include automated systems for display management, multimedia presentation design, and graphics design. As a case study for the integrated approach to IUI design we will describe the Vista research project conducted by NASA (Horvitz & Barry), and a more ambitions research project called CHORIS started at the Lockheed AI Center (Tyler et al., 85). The goal of the Vista project was to develop an automated display management system for real-time monitoring of Space Shuttle propulsion systems. CHORIS is an acronym for Computer-Human Object-Oriented Reasoning Interface System. As the name implies, the long-term goal of the project was to develop a generic framework for intelligent interfaces.

The usability issue addressed by the Vista project was basically one of information overload. In the propulsion system monitoring task the operator receives telemetry data in real time from sensors that monitor different aspects of the Shuttle's propulsion system. Approximately 130 distinct pieces of data can be seen in the original text-based propulsion-monitoring interface shown in Figure 2 (Horvitz & Barry, 4). The operator's task is to determine if any of the values are outside normal parameters and to take necessary action. This task is described as both time critical, and high stakes because delays in responding to anomalies in the telemetry data have the potential to lead to loss of life. The original monitoring interface contained such a dense display of information that even expert operators could experience difficulty quickly identifying problems simply due to the time required to scan through the mass of data (Horvitz & Barry, 3).

While the CHORIS project had more general goals than the Vista project, Tyler et al. describe an example application of their general framework to the specific domain of emergency crisis management systems (ECMS). Their system contains large a large database of geographic information including maps, the location of important sites, as well as population data. This information is highly useful for the crisis management task, but only if the information is easily accessible to the user. Like Vista the basic usability problem addressed by the system is how to effectively display information (Tyler et al., 89).

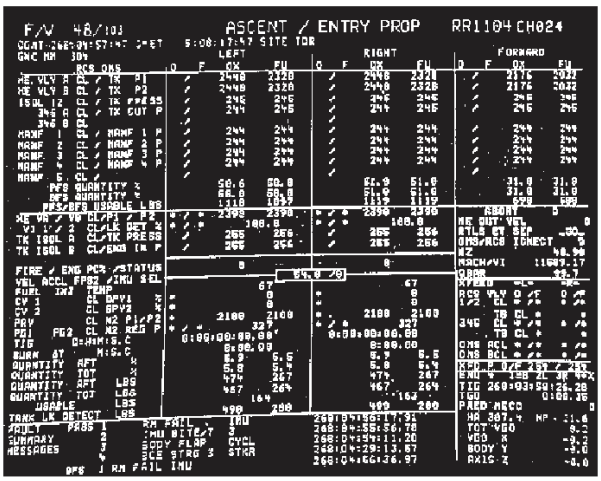

Figure 2: A classic Space Shuttle propulsion system monitoring interface.

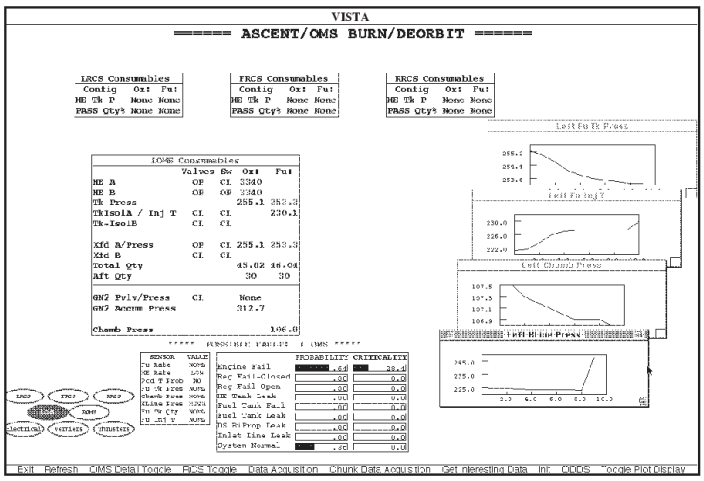

Figure 3: Vista 1 prototype Space Shuttle propulsion system monitoring interface.

The solution proposed by the Vista team was a display manager consisting of two main components. The first was a probabilistic reasoning module that received telemetry data from the Space Shuttle, and the second was a set of display templates designed with the help of NASA experts. The probabilistic reasoning module was used to infer the likelihood of different propulsion system failures like propellant tank leaks and engine failures. Based on the most likely system failure, the display manager would select a template that included information the user needed to diagnose the potential failure. Each template was customized for a specific propulsion subsystem, and each template had several versions at increasing levels of detail (Horvitz & Barry, 3).

The Vista I prototype interface is shown in Figure 3. It clearly contains less information displayed in a more coherent fashion than the classic monitoring interface. The use of display templates that were hand crafted by experts helps to insure that interface is stable, cohesive, and predictable. This is very important from an HCI standpoint; however, it means the Vista I display manager has only a very coarse grained control over the display. A second generation Vista system based directly on the Vista I prototype was put into service in 1993 (Horvitz & Barry, 9). The last available information on this project suggests that several additional capabilities were being investigated for Vista III including dynamically highlighting individual data items or data clusters within a template using color or intensity (Horvitz & Barry, 9).

Compared to the Vista project, the CHORIS project describes a display management framework that has much more autonomous control over adapting interface. This leads Tyler et al. to raise three important questions. These are “What to adapt?”, “When to adapt?”, and “How to adapt?” (99-101). We are primarily concerned with the question of what to adapt since it governs the level of granularity the system will have over the interface. As explained above, the Vista display manager could only choose from among a fixed set of static templates. The CHORIS project takes a more ambitious stance on the issue of what to adapt. As seen in Figure 4, the primary CHORIS ECMS interface widget is a large map display. Based on information extracted from natural language user queries, a CHORIS component called the Adaptor automatically zooms and pans the map display to a location and magnification it

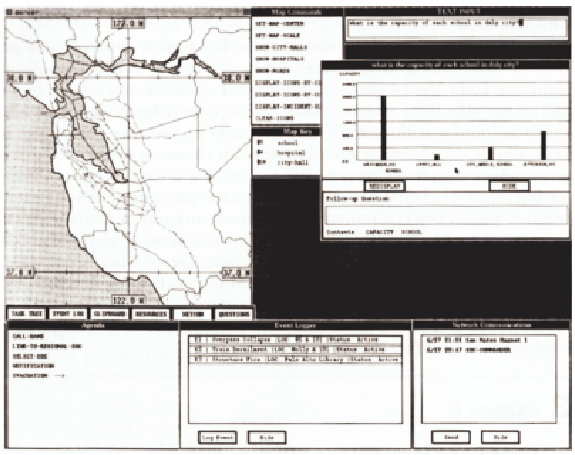

Figure 4: CHORIS ECMS interface showing map view and graph for evacuation scenario.

deems appropriate. User queries may also return specific data items or sets of items. The Adaptor decides how best to render these items by selecting from a set of options, including lists, tables, forms, and charts (as seen in Figure 4). An additional dimension of adaptively results from making the CHORIS ECMS menu system dynamic. Menu options that the Adaptor decides are not relevant given the current task context are hidden from direct view (Tyler et al., 89-91).

Joseph Sullivan, director of the Intelligent Interfaces project at Lockheed for ten years, reports that the CHORIS ECMS system was "the primary accomplishment" of the Intelligent Interfaces project. He also reports that the system was used in simulation to support actual crisis management situations including the 1989 Loma Prieta earthquake in Northern California (Sullivan). However, research on the CHORIS project never led to a production system.

Our analysis of these two projects suggests an important trade off between adaptively and usability in integrated IUI design. By giving the AI module more control over the interface, there is a greater risk that an automatically generated or adapted interface will violate usability principles. Conversely, to ensure that the AI module does not adversely affect the usability of the interface, it may be necessary to severely restrict the range of possible adaptations that are allowed. The CHORIS project leans toward the side of usability violation as a result of giving the Adaptor component too much control over the interface. For example, automating the choice of displayed menu items has the potential to make the CHORIS ECMS interface unstable. In the case of the Vista project, we are in the opposite situation. Since the display templates are handcrafted, there is a decreased possibility for usability violation, but the adaptation is also less dynamic.

The Agent Based Approach

Unfortunately the term "AI agent" or "intelligent agent" has been misconstrued in the popular media to the point where researchers are hesitant to use it (Shneiderman & Maes). In a pure AI context he term "agent" is used to refer to a piece of software that can perceive its environment through sensors, and act in its environment through effectors (Russell & Norvig, 31). The environment of an interface agent consists of the user and the application interface, which it "perceives" through application status information or user queries. An interface agent's "effectors" depend on its capabilities. In some cases the agent may be able to communicate directly with the user.

Agent Based Approach

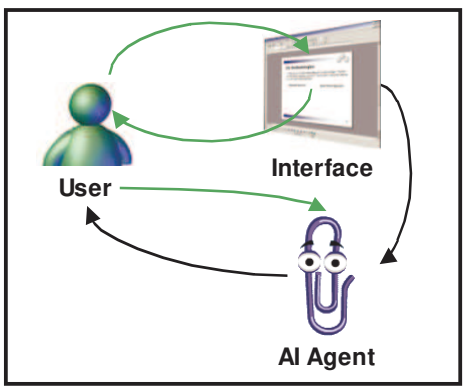

Figure 5: Diagram of the Agent Based Approach to IUI Design.

The agent based approach to IUI design differs from the integrated approach in several ways. As indicated in Figure 5 the AI agent is a separate "entity" from the main application interface. The AI agent is able to obtain status information from interface that it can use to observe user actions, but it is not able to control the primary application interface in any way. Instead, the agent often has a separate interface that it may use for direct dialogue with the user.

Canonical examples of the agent-based approach to Intelligent User Interface design include automated assistants such as help agents and web browsing assistants. The motivation behind research into agent-based systems is usually described as a desire to relieve some of the cognitive burden associated with using complex software systems by partially offloading it to the intelligent agent. According to long-time agent advocate Pattie Maes, an IUI agent that knows a user's needs and goals has an advantage over conventional software systems because it can be proactive. It can take initiative and perform tasks the user would otherwise have to perform herself (Shneiderman & Maes, 49-50).

As a case study of the agent based approach to IUI design we will describe the Lumière project conducted by the Microsoft Adaptive Systems and Interaction group and compare the results with the well-known Microsoft Office Assistant. The goal of the Lumière project was to design an AI agent Help Wizard system (Horvitz et al.). Such help wizard systems address what we might call a meta-usability problem. Mainly, complex software systems such as modern productivity packages are so packed with features they become quite difficult for novice users to learn and use effectively. McGrenere has referred to this problem as "feature bloat" (McGrenere, 2). A help wizard agent attempts to overcome this problem by actively monitoring the user. When the agent determines that the user needs assistance, it proactively offers help.

In order to demonstrate the potential of their approach, the Lumière team created a research prototype called Lumière/Excel. The prototype was designed around a custom version of Excel that exposed the user interface event stream to the Lumière agent. These atomic events were then translated by the agent into high-level user actions such as menu surfing. The computed high-level actions were fed into a set of Bayesian networks that were used to compute various probabilities including the likelihood that the user required assistance (Horvitz et al).

The Help Wizard interface used in Lumière/Excel consisted of a set of standard dialogue boxes. The main Help Wizard dialogue box provided a free-text search that the user could use to directly ask the agent questions. These queries could be combined with observed user actions to add context to the search. In response to a user's query, the Help Wizard would display a list of the AI system's best guesses for help topics sorted by their likelihood (Horvitz et al). The Lumière agent did more than simply observe the user and respond to queries, however. As Maes suggests, the Lumière agent was proactive. The Help Wizard would attempt to offer unsolicited assistance to the user if the likelihood that the user needed assistance passed a given threshold value. When this occurred a dialogue box would appear offering the agent's best guess at relevant help topics as shown in Figure 6.

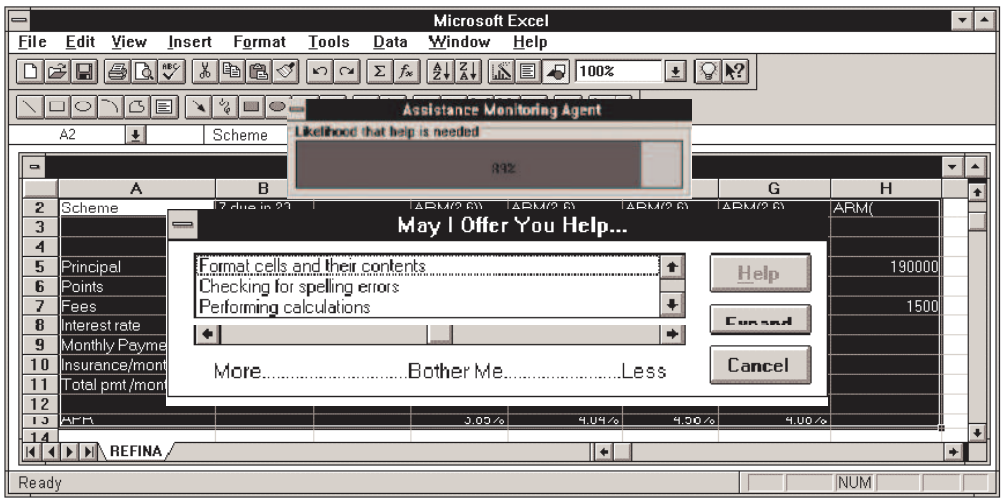

Figure 6: This figure shows the Lumiere/Excel research prototype. The top box shows the assistance monitor indicating 89% likelihood that assistance is required. The bottom box suggests help topics and provides a control for adjusting the likelihood threshold.

Some of the technology created for the Lumière project was eventually incorporated into Microsoft's Office 97 productivity software as the Office Assistant. In his 1998 paper on the Lumière project, Horvitz details exactly which Lumière components were actually implemented by the Office Assistant team and which were left out. A summary of the components omitted from the Office Assistant is presented in Table 1. In light of these omissions it is not hard to see why the Office Assistant has proven to be a poor performer relative to its much more sophisticated Lumière predecessor. While a basic system for reasoning about a user's goals was implemented in the Assistant (ie: "I see you are writing a letter"), most of the core reasoning modules were left out (Horvitz et al). These include modules for reasoning about the competency level of the user, reasoning about events through time, and most importantly reasoning about whether or not the user was in need of assistance. This last omission is doubtless responsible for the Office Assistant's tendency to offer unsolicited advice when it is least desired. Further, the Office assistant had little support for inferring high-level user actions from low-level events, and did not have the ability to combine action context information with free text queries. Finally, the Office Assistant did not keep persistent user profiles, or have user controls (Horvitz et al).

Component	Lumière/Excel	Office 97 Assistant
User Goal Reasoning	Yes	Yes
Need of Assistance Reasoning	Yes	No
User Competency Reasoning	Yes	No
Temporal Reasoning	Yes	No
User Action/Text Query Interaction	Yes	No
Support for Complex Actions	Yes	No
Persistent User Profiles	Yes	No
User Controls	Yes	No
Anthropomorphic Interface	No	Yes

Table 1: Components present in the Lumière Excel prototype compared to the Office Assistant.

Another interesting difference between the Office Assistant and Lumiere was the choice to implement an anthropomorphic interface as opposed to the standard window-based system used by the Lumière/Excel Help Wizard. While animated character based interfaces may have

at one point been synonymous with interface agents in the popular media, they have since fallen out of flavor with the research community almost completely. This led Shneiderman to confidently declare "anthropomorphic interfaces not to be the future of computing" (Shneiderman & Maes, 54). Maes, whose early work was strongly in favor of social interface agents, concedes "most successful interfaces are ones where the agents are pretty much invisible" (Shneiderman & Maes, 56).

Our comparison between Lumière and the fielded Office Assistant leads to some important lessons that must be taken into consideration regarding the agent-based approach. Unlike the Integrated approach, there is no danger of violating usability principles by making an interface agent more capable. To the contrary, an interface agent that is lacking in functionality may have a negative impact on user productivity (Catrambone et al.). Secondly, there is a potential for misuse of the agent based approach: An automated Help Wizard system can appear like an attractive fix for a poorly designed application relative to the prospect of redesigning the application from scratch. Maes cautions against this, reinforcing the point that an interface agent is not a substitute replacement for good primary application design (Shneiderman & Maes, 56).

The Mixed Initiative Approach

The Mixed Initiative Approach is different from both the purely integrated approach and the purely agent-based approach to intelligent user interface design. The primary metaphor behind the mixed initiative approach is collaboration between the human user and the AI agent. The approach is characterized by a special type of AI agent that is able to manipulate the application interface in much the same way a user would. In a GUI context this means the agent might be able to fill in the values for text boxes and control other interface widgets to support the task the user is currently performing. Note that this differs significantly from the integrated approach where the AI components controlled what information was being displayed on the screen and how it was laid out. Also note that this approach is significantly different from the agent-based approach where the agent has no control over the primary application interface. Apart from the requirement that the agent be able to manipulate interface components we, impose few restrictions on this class of methods. As indicated by the dashed arrows in Figure 7, the AI Agent that may or may not have a separate interface from the primary software application.

Mixed Initiative Approach

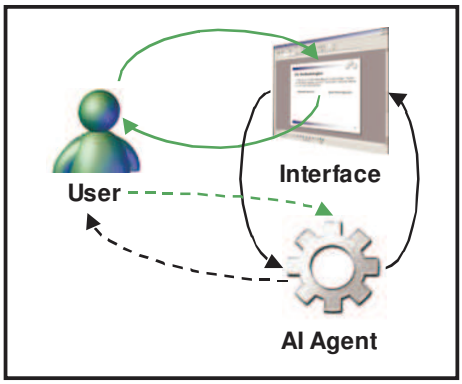

Figure 7: Diagram of the Mixed Initiative Approach to IUI design.

As we indicated at above, the mixed initiative approach seeks to enable collaboration between the agent and the user so that the user can perform his or her task more efficiently. The case study in the previous section is fairly typical of the traditional agent based approach in that the Help Wizard assists the user with a one off task. Namely, the Help Wizard offers the user assistance with using the application interface. However, the user's goal is not simply to *use* the application interface, but to *use the application interface to accomplish some task*.

Horvitz refers to a mixed initiative agent that assists the user in performing his primary task as providing “significant value-added automation” (Horvitz).

To illustrate the mixed initiative approach we will examine two different research projects. The first project introduces a web-based, multi-source travel planner. In this task a user whishes to plan a trip to some location. Under normal circumstances the user would have to visit several different websites such as airlines, hotels, and car rental agencies, as well as consult several applications including calendars and schedules in order to plan the trip. This can be a lengthy operation and it requires reentering the same basic data items multiple times (Frank et al.). The second project we consider is another offering from the Microsoft Adaptive Systems and Interaction group, a mixed initiative research prototype called LookOut. LookOut integrates with the Microsoft Outlook email client and attempts to automatically enter meetings into the user’s schedule based on the contents of email messages (Horvitz).

One of the main issues with the mixed imitative approach is coordinating control of interface components between the user and the agent. Obviously if the user and the agent were able to simultaneously manipulate the same interface widgets collaboration might quickly break down. In the system proposed by Frank et al. the user begins with control of a travel panning applet. As a first step, the user need only select the name of the person or organization she wants to visit. This selection initiates the travel planning agent which then retrieves the corresponding address from the users address book, the corresponding meeting dates from the users scheduling software, and an Internet weather forecast (Frank et al.).

Frank et al. resort to the use of color as the primary mechanism for indicating control of input fields in the travel planner as seen in Figure 8. While the agent is retrieving information to fill a field, that field appears with a red rectangle around it. Once the information has been retrieved, a field filled by the agent appears with a green rectangle around it. The user can modify any field that was, or is in the process of being filled by the agent by simply clicking on it, and changing the information it contains. The field is then marked in blue to indicate the user controls it. If the user decides to modify the value of an agent-controlled field that has dependent sub fields, the agent will automatically re-compute or retrieve appropriate values for those fields. Lastly, if the agent does not produce the desired results, the user can make modifications manually and let the agent refill dependent fields automatically (Frank et al.). In this simple collaborative model the agent eagerly attempts to fill as many fields as it can, but immediately concedes control of fields to the user if the user wants to modify them.

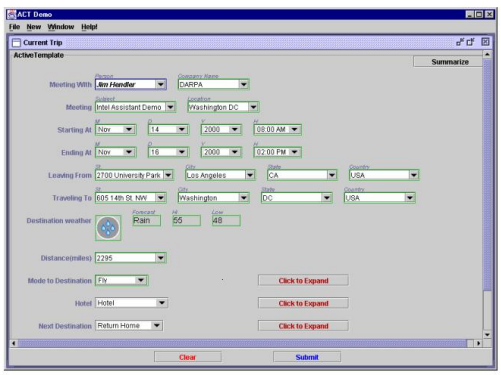

Figure 8: Travel planning applet.

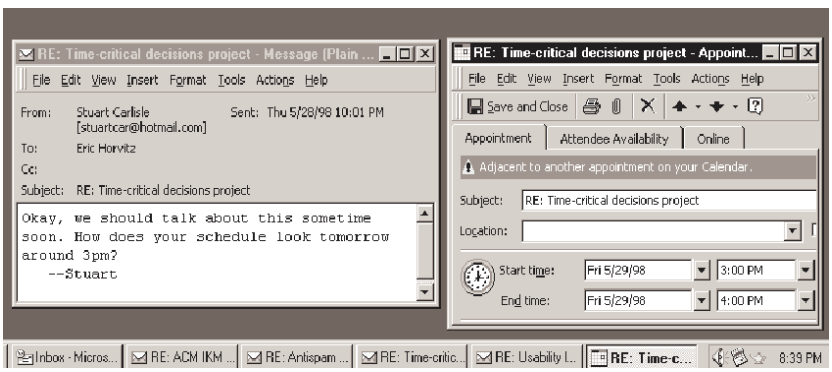

Figure 9: LookOut Intercepts meeting information from an email.

With respect to input control the LookOut automated scheduling system proposed by Horvitz works in a somewhat simpler fashion than the travel planning system created by Frank et al. When the LookOut agent detects an email message that references a proposed meeting it takes one of several actions depending on the available information, and the likelihood of user goals (Horvitz). As its most tentative action the agent will automatically open the Outlook calendar application to a month or week referenced in the email allowing the user to select a particular day. At the other end of the spectrum, if the Lookout agent was able to extract sufficient information from the email message, and the likelihood that the user wants to schedule a meeting is high, then the agent will automatically open the Outlook scheduler and fill in the information (Horvitz). An example of this is shown in Figure 9 where LookOut has determined that a meeting should be scheduled at 3:00pm based on the active email message. In any case, collaboration with the LookOut agent is a non-iterative process. The agent will proactively fill in all the scheduler fields it has information for and then immediately turn control over to the user. This eliminates the need for a complicated mechanism to indicate who has control of input fields, but at the same time it restricts the collaboration between user and agent to a less interactive level.

In addition to the issue of input control, the LookOut project addressed several other important issues involved in applying the mixed initiative approach. First is the issue of dealing with uncertainty in user goals mentioned briefly above. In the travel planning scenario we can assume that the user only has a single goal: to successfully complete the form by filling out successive fields. Simple rules and dependencies between form fields govern what action the travel planning agent needs to perform next (although the actions themselves may be quite complicated). On the other hand, an Outlook user can have one of any number of goals including composing and reading email, viewing the calendar, using the schedule, etc... Thus the LookOut system needs to have the ability to reason about what task a user is trying to perform and whether he wants to be bothered by LookOut. Such reasoning abilities were included in lookout, but the research team also provided another solution to this problem: they designed LookOut to have multiple interaction modalities. In the manual modality, the user must activate LookOut before it will perform any actions. This means the user can delegate tasks to the agent without having to worry about being unwittingly disturbed at an inopportune moment. In the automated-assistance modality LookOut will proactively perform actions according to its internal beliefs, and actively communicate with the user using standard dialogue boxes. As a third option LookOut can operate in automated-assistance mode with a social interface. These three options give the user greater flexibility and a degree of control over the LookOut agent. They also addresses concerns of direct manipulation proponents like Shneiderman who worry that agent based systems take too much control out of the hands of software users (Shneiderman and Maes, 48).

Conclusions:

In this paper we have explored three different approaches to intelligent user interface design from a human-computer interaction standpoint. These include an integrated approach, an agent based approach, and a mixed initiative approach. The integrated approach is characterized by AI systems that perform automated display management, graphics layout, and interface generation. As our examples in this area have shown, the integrated approach

necessitates building AI components into the software application from the bottom up. Designs that follow this approach like Vista and CHORIS ECMS are highly specialized applications. The main concern with systems that have direct control over the appearance of the display is the possibility for usability violations in the interface. This results in a tradeoff between the degrees of adaptation permitted and the possibility of automatically generating an interface with poor usability.

In contrast to the integrated approach we have the agent-based approach, which is characterized by wizard or helper-like applications that often assist the user with one-off tasks like using the primary interface of a word processor or spreadsheet application. While the promise of agent-based systems is that they will eventually be able offload some of a users cognitive burden, there is an immediate danger that the agent based approach will be viewed as a patch for poor application design.

Lastly we have the mixed-initiative approach, which differs from both the agent based approach and the integrated approach. A mixed-initiative system allows the agent to manipulate interface widgets in the same manner a user would. Characteristic examples of this approach are systems that help users to fill out lengthy forms or perform repetitive tasks within the interface. The primary issue with this approach is finding effective methods for structuring the collaboration between user and agent.

All three of these approaches provide intriguing possibilities for future research directions in the field of human computer interaction. Historically these directions have often been opposed by proponents of direct manipulation like Ben Shneiderman who has instead looked for inspiration in areas like scientific visualization (Shneiderman & Maes, 45). However, recent times have seen the beginning of a convergence between the IUI community and the direct manipulation community. Prominent IUI researchers like Maes have taken up more conservative positions (Shneiderman & Maes, 61), while Horvitz has begun to actively consider concerns over direct user control of agents (Horvitz, 2). Meanwhile, Shneiderman has indicated that he is prepared to accept the introduction of artificial intelligence techniques into software systems so long as it does not directly manifest itself in the interface (Shneiderman & Maes, 54). If this convergence continues it will doubtlessly result in new and interesting research with the potential to improve the usability of large, complex software systems.

Prominent IUI Researchers

Eric Horvitz

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

- Currently head of Microsoft's Adaptive Systems & Interaction group
- His focus is on using probability and decision theory in the construction intelligent, flexible computer-based reasoning systems.
- Head Researcher on NASA Vista Project

Pattie Maes

http://www.media.mit.edu/people/pattie/

- Associate Professor at MIT's Media Laboratory
- Founded and directs the MIT Software Agents Group
- Maes' MIT team claim to have built the first successful prototypes of agents for personalized information filtering, eager assistant agents, agents that buy and sell on behalf of a user, matchmaking agents, and remembrance agents.

Kristian J Hammond

http://www.infolab.northwestern.edu/infolab/people/personal.asp?id=40

- Director of the Intelligent Information Laboratory, Northwestern University.
- Chair of 2002 ACM SIGAI International Conference on Intelligent User Interfaces (IUI).

Joseph W. Sullivan

http://www.fxpal.com/people/sullivan/

- Lockheed Palo Alto Research Laboratory Intelligent Interfaces project leader for 10 years.
- Organized the workshop on *"Architectures and Elements of Intelligent Interfaces"* held in 1988 in Monterey, CA. This workshop is credited as the founding event for the ACM SIGAI International Conference on Intelligent User Interfaces (IUI).
- He is also coeditor of the IUI anthology, *Intelligent User Interfaces*.

Wolfgang Wahlster

http://www.dfki.de/~wahlster/

- Director and CEO of the German Research Center for Artificial Intelligence (DFKI GmbH)
- Head of the Intelligent User Interfaces Lab at DFKI.
- Professor of Computer Science at the Universität des Saarlandes, Germany.
- Coeditor of the anthology *Readings In Intelligent User Interfaces.*

Bibliography

Catrambone, R., Stasko, J., Xiao, J. (2002). Anthropomorphic Agents as a User Interface Paradigm: Experimental Findings and a Framework for Research. Proceedings of the 24th Annual Meeting of the Cognitive Science Society. George Mason University. Fairfax, Virginia. Aug 8-10, 2002.

Frank, M., Muslea, M., Oh, J., Minton, S., Knoblock, C. An Intelligent User Interface for Mixed-Initiative Multi-Source Travel Planning. (2001). Proceedings of IUI 2001, ACM SIGAI Conference on Inteligent User Interfaces. 2001. ACM Press.

Horvitz, E. (1999). Principles of Mixed Initiative User Interfaces. Proceedings of CHI '99, ACM SIGCHI Conference on Human Factors in Computing Systems, Pittsburgh, PA, May 1999. ACM Press. pp 159-166

Horvitz, E. and Barry, M. (1995). Display of Information for Time-Critical Decision Making. Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence. August 1995.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D. and Rommelse, K. (1998) The Lumiere Project. Proceedings of the.14th Conference on Uncertainty in Artificial Intelligence. July 1998.

Maybury, M. T. and Wahlster, W. ed. (1999). *Readings In Intelligent User Interfaces*. San Francisco, CA: Morgan-Kaufmann.

McGrenere, J., (2002). The design and evaluation of multiple interfaces: A solution for complex software. Dissertation. Department of Computer Science, University of Toronto.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach.* N.J.: Prentice-Hall.

Tyler, S. W., Schlossberg, J., Gargan, R. A., Cook, L., Sullivan, J.W. An Intelligent Interface Architecture For Adaptive Interaction. (1991). In: J. W. Sullivan, A. W. Tyler (eds): *Intelligent User Interfaces*, ACM Press, 1991

Shneiderman, B. and Maes, P. (1997). Direct Manipulation vs. Interface Agents: Excerpts from debates at IUI 97 and CHI 97. ACM Interactions, 4(6), 42-6.

Sullivan, J. W. and Tyler, S. W. ed. (1991). *Intelligent User Interfaces*. New York, NY: ACM Press.

Sullivan, J. W. (2002). *Joseph W. Sullivan's Personal Web Page.* Online. Available: http://www.fxpal.com/people/sullivan/. Accessed Dec 1, 2002.

Wærn, A. *What is an Intelligent Interface?* (1997). Notes from an introduction seminar, March 1997. Online. Available: http://www.sics.se/~annika/papers/intint.html.