Designing for Human-Agent Interaction

Michael Lewis
School of Information Sciences
University of Pittsburgh

Abstract

Interacting with a computer requires adopting some metaphor to guide our actions and expectations. Most human-computer interfaces can be classified according to two dominant metaphors: agent or environment. Interactions based on an agent metaphor treat the computer as an intermediary which responds to user requests. In the environment metaphor a model of the task domain is presented for the user to interact with directly. The term "agent" has come to refer to automation of aspects of human computer interaction such as anticipating commands or autonomously performing actions. Norman's 1984 model of HCI is introduced as reference to organize and evaluate research in human-agent interaction. A wide variety of heterogeneous research involving human-agent interaction is shown to reflect automation of one of the stages of action or evaluation within Norman's model. Improvements in HAI are expected to result from more heterogeneous use of methods which target multiple stages simultaneously.

Agent vs. Environment

Interacting with a computer requires adopting some metaphor to guide our actions and expectations. Most human-computer interfaces can be classified according to two dominant metaphors: *agent* or *environment*. Interactions based on an agent metaphor treat the computer as an intermediary which responds to user requests. In the environment metaphor a model of the task domain is presented for the user to interact with directly.

Command line and screen editors provide a simple contrast between these approaches. To change a word in a command line editor such as "ex" you must: 1) instruct the editor to move to the appropriate line number then 2) enter some command such as s/old/new (locate string "old" then substitute for string "old" string "new"). To view this change in the context of the surrounding text would then require composing a command such as p .-10,.+20 (print from 10 lines before to twenty lines after the current location). This same task is far easier with a screen editor where you simply move the sprite using mouse or cursor key to the offending word, delete it perhaps by backspacing over it, and type in the replacement. This keystroke superiority of screen over commandline editing is a well known (Card, Moran, and Newell 1983) HCI

This research was supported by ONR grant N-00014-96-1-1222

result. If the task were changed to "change every occurrence of old_word to new_word" the relative advantage is reversed and an instruction to an agent such as "g/old/s/old/new" (locate string "old" then substitute for string "old" string "new" for g=every occurrence of string "old") is far simpler than scouring the surrogate document for occurrences of "old_word", erasing each, and typing "new_word" in its place. For this example the character of predictable errors will differ as well; the subject interacting directly with the document is likely to miss some occurrences of "old_word" while the subject issuing the global command may suffer unintended consequences such as changing "not_old_word" to "not_new_word".

In practice, the better features of line editors such as string searching and global replace have almost always been retained in screen oriented editors, leading to interfaces in which indirect requests can be issued to perform tasks for which direct manipulation proves too cumbersome. The distinction between agent and environment metaphors is not identical to that between agent-based and direct manipulation-based interfaces which has been much debated (Schneiderman and Maes 1997). The agent/environment distinction reflects the semantics (action vs. request) of the interaction rather than its syntax (command line vs. button press). The binocular icon search button found on Netscape browsers, for example, uses the affordances of a pushable button to advertise its availability and means for initiating search but leaves the task of locating a string to an "agent" rather than requiring the user to search the text line by line. Task actions communicatable using an environmental metaphor are a proper subset of those which could be specified to an agent and are just those tasks such as iconic desktops, text editing, draw programs, or geographical information systems which can provide a clear, literal correspondences between task domain and onscreen representation.

The power of this approach which provides advertisement and unique identification and selectability of available objects and actions is reflected in the ascendence of graphical user interfaces (GUI's). The value of the agent metaphor to interaction only becomes apparent when objects are not present or fully visualizable and actions are repetitive, delayed, or poorly specified. The distinctions between agent and environment based HCI are very similar to those between manual and automated action in the physical world. It is much simpler for us to drive a car or set a table than to instruct a robot to do so, yet we would rather adjust a thermostat or program a milling machine than repeatedly performing these actions by hand. While the computer offers the ultimate in flexible automation, instructing it do what we wish may be arbitrarily hard for humans as demonstrated by the difficulty experienced in using traditional programming and scripting languages. The growing popularity of "agent-based" interaction reflects the emergence of an increasingly powerful and complex computing environment bringing with it desires to perform flexible tasks involving multiple or unknown objects by users who do not wish or may not have the ability to program.

Norman's (1988) ecological model of HCI will be reviewed and used to organize research in human-agent interaction. The premise is that software agents are intended to automate repetitive, poorly specified, or complex processes by bridging the gulfs between a user's desires and actions which could satisfy them. Tasks, domains, and interaction methods will be categorized according to the uncertainties they bring to or

reduce at stages in this model. A maturing paradigm of human agent interaction is envisioned in which adaptation, user profiles, demonstration, and scripting are used as appropriate to facilitate human-agent interactions.

Reference Model

Don Norman (1986) characterizes human-computer interaction as the problem of bridging twin gulfs of *execution* and *evaluation*. The execution side of the cycle involves translating a goal into a sequence of actions for achieving that goal. The evaluation side involves using feedback from the domain to compare the result of the action to the goal. The model is cybernetic rather than logical in that it emphasizes feedback and incremental action rather than problem space search or planning. A crucial feature of this model is that tampering with either side of the loop can lead to detrimental or unanticipated results. If the execution side is automated the human may fail to observe effects of actions and be unable to correct errors or modulate ongoing behavior. If the evaluation side is automated the human may be unable to track the effect of actions and adjust to their results.

Norman proposes seven stages of action in this model to link the user's goals to the world. The stages of execution are: forming an intention to act, translating this intention into a planned sequence of actions and executing that sequence. The stages of evaluation are perceiving the state of the world, interpreting this perception in light of prior action and evaluating that change with respect to initial goal. The gulfs refer to the interface/metaphor which separates the user's goals from the application domain in which they must be realized. An example of this form of analysis is shown below:

Changing the format of letter (Norman 1986, p. 44)	
Goal	improve the appearance of the letter
Intention	change the paragraph style from indented to blocked; replace all new-paragraph commands with skip-line commands
Action Specification	sub/.pp/.sp/whole
Execution	type "sub/.pp/.sp/whole"
Perception	each line of text begins at left margin
Interpretation	the paragraphs are blocked
Evaluation	the letter now "looks better"?

Although approximate rather than precise and proposed in the context of early GUI's Norman's model provides a useful reference for analyzing human computer interactions of all forms because it identifies the cognitive processes and linkage among them which must be supported for human-computer interactions to succeed.

An *agent* is defined to be a program which automates some stage(s) of this human information processing cycle. This definition does not extend to software-only agents found in multiagent systems and excludes human-computer interactions involving simple direct manipulation actions or explicit command line requests.

Figure 1 shows the Norman reference model and ways in which the involved cognitive

processing might be automated. Automated processes are indicated in italics within dashed boxes. Serial (non looping) automation strategies range from direct aiding such as action sequencing or attentional filtering to executive functions such as anticipatory action or prioritized filtering. Agents which interact continuously with the domain may be semiautonomous with or without feedback or may perform their assistive functions completely independent of the user. These categories could be elaborated to incorporate the transparency of automated processes and other aspects likely to affect human-agent interactions, but these nine should suffice to suggest some of the structural complexities and concerns which should inform the design of agents to assist human users. A rule of thumb would be that any serial configuration which automates a complete side of the loop isolates the user and may lead to breakdown as may semiautonomous configurations which do not explicitly supply appropriate feedback.

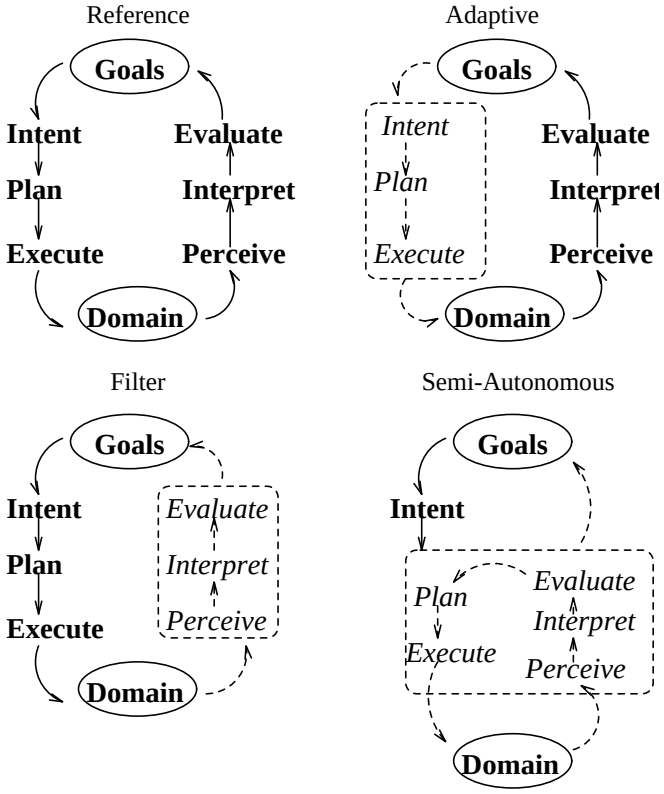

Figure 1. HAI Reference Model

As shown in Figure 1 automation can occur through either semiautonomous processes set in motion by the user or through by-passing stages of execution or evaluation which the user must otherwise perform. An *agent* which searches several databases for a favorable price would be an example of a semiautonomous process. This form of closed loop automation is found in process industries where operators establish a variety of setpoints for valves, breakers, and proportional controllers. For automation of this sort to succeed the user needs a relatively detailed mental model of the domain and what the automation is to do in order to "program" it (transparency) and subsequently would benefit from good displays and methods for monitoring its

performance (feedback).

An adaptive agent which volunteers "ls -t" after learning that a user invariably types "ls -t" to see what the current paper is after changing directories to "Papers" automates stages involved in instantiating goal, forming intent, and planning sequence of actions. An attentional filter that rings a bell and raises the flag on a mailbox icon whenever mail from members of the work group but not anyone else, automates the stage involved in perceiving the domain. Examining other behaviors attributed to software agents will establish that this model of closed loops and bridged stages accommodates most.

Equating "software agent" with programs which automate aspects of interaction agrees with many popular definitions for example, Shneiderman (Shneiderman and Maes 1997) lists:

adaptive behavior (automating execution)
accepts vague goal specification (automating execution)
gives you just what you need (automating execution & evaluation)
works when you don't (semiautonomous process)
works where you aren't (semiautonomous process)

in his "autopsy on agents"
leaving only

anthropometric representation

at variance with our definition.

On the other side of the argument Patti Maes (Shneiderman and Maes 1997) identifies "removed in time and place" (semiautonomous) and "adaptive to user's habits, performance, and interests" (automating execution/evaluation) as an agent's defining characteristics. She goes on to deny the necessity of personifying or anthropomorphizing agents and proposes as an additional distinguishing characteristic of successful agents that they rely on user programming or machine learning rather than traditional artificial intelligence (AI) techniques. Elsewhere (Maes 1994) she argues for "indirect management" (semiautonomous process) alone as distinguishing human-agent interactions.

Other theorists (Payne & Edwards 1997, Negroponte 1995, Laurel 1997) stress the agent's assistance in "common tasks" rather than mechanisms used to assist as the defining characteristic. Finally there are broad and ambitious taxonomies of agents based on laudatory characteristics such as "inferential capability" (Bradshaw 1997, Milewski & Lewis 1997), degree of "agency", "mobility", and "intelligence" (Gilbert 1997), degree of "autonomy", "cooperation", and "learning" (Nwana 1996), and place within the universe of biological, robotic, and digital entities (Franklin and Graesser 1996).

Our objective is not to add to these typologies of software agents but instead to examine the potential of agent automation to facilitate human-computer interaction. The basic premise of this approach is that:

The greatest impediment to assisting human users lies in communicating their intent and making results intelligible to them.

Today in almost all cases the limiting factor in HCI is not computing cycles or connectivity to information sources or characteristics of peripherals (the machine side) *but* in the user's ability and/or willingness to communicate these desires and sift, organize, and represent the machine's response to satisfy them (the human side). So, for example, although I have a Perl interpreter for which I could in principle write a script which would visit a list of web sites, extract particular information from each, perform comparisons, and return the result to my Inbox, I will almost certainly not do so. The effort of prescribing how to navigate, how to parse HTML at each of the sites, and how to analyze and return the results is infinitely more difficult than pushing buttons and following links on my browser myself. It would probably remain more difficult even if I had to repeat the search ten or fifteen times. Even when the time to search repeatedly equaled the time to program, I would still prefer the manual search because of the lower cognitive effort. As this example suggests, scripting languages may fit our definition as maximally powerful instructable "agents", yet they fail miserably in satisfying Negroponte's (1995) desire for an implacable butler or mine for a no hassle autonomous web searcher. The problem is a human variant of Turing equivalency. Scripting languages or for that matter assembly code may meet the letter of a definition of agents but the spirit clearly lies in the *ease* with which our desires can be communicated.

As this example suggests, there is considerable merit in the vaguer definitions, such as "assistance at common tasks" which implicitly incorporate ease and utility. To reflect this insight the definition of "software agent" is amended to:

A program which automates some stage(s) of the human information processing cycle leading to a significant decrease in human effort.

This conception is consistent with Maes' (Shneiderman and Maes 1997) notion that "an agent basically interacts with the application just like you interact with the application" in that the locus of agent automation is on the executive tasks otherwise performed by the human. As in other instances of pervasive automation software agents will "disappear" from notice as their behavior becomes sufficiently familiar just as we no longer think of elevators as "automated" but instead as directly "controlled".

Tasks and Constraints

As the script vs. interpreter contrast suggests, any constraint on the agent's behavior from outside reduces both the generality of possible tasks and the human effort of initiating them. The arguments for the superiority of direct manipulation interfaces (Shneiderman and Maes 1997, Lanier 1994) rest on just this point. The presence of buttons, toolbars, dropdown boxes, and other visible controls makes the set of possible actions goals perceptually available without relying on memory or other cognitive resources. In the point and click environment the problem of planning and executing sequences of actions is trivialized to selecting a visible object and executing the

action associated with it. Immediate feedback is typically provided for selection and operation short circuiting the problems of perceiving and interpreting the action's effect leaving only *evaluation* as a task for the user. Any less constrained task becomes a candidate for automation. Because stages of execution involving the translation of a goal into an intention to act and the translation of this intention into a set of internal commands are unobservable until commands are executed, most automation of execution bridges all three stages. Automation of execution therefore is primarily a matter of goal recognition. The other class of agent aiding lies in automating the entire loop. This may be as simple as independent execution of a loop to a complex interplay of guidance and execution.

Automating Execution

Hypothesized agents such as Negroponte's (1995) digital butler or the Knowledge Navigator in bow tie portrayed in Apple's 1977 video achieve their effect by accurately anticipating their users' desires; i.e.; automating goal instantiation. For this to work there must be very few, preferably one, goal(s) satisfied by a stereotyped sequence of actions associated with particular states and events. Automation of this sort is open loop leaving evaluation up to the user. The user initiates actions under "ambiguous" conditions which are interleaved with agent initiated actions where appropriate goals can be identified. This form of automation can be achieved by matching a user's characteristics to that of a sample or demographic group with known goals and actions, observing and replicating the user's own state $\rightarrow$ action sequences, or by explicit instruction as in login scripts or resource files.

Automating the Cycle

Any process for which evaluation may be separated in time, space, or representation from initiating action is a candidate for automation of this sort. Conventional computer applications such as a payroll program which are set in motion and later return confirmation of success or failure are examples of this type of interaction. With closed loop automation, sequences of actions, comparisons of results, and adjustment of activity all must be coordinated with a user's intentions. While ordinary programming meets the "automation" requirement it does not satisfy the flexibility and ease of expressive effort expected from software agents. Most examples of agents of this sort are only distinguishable from conventional programs in being schedulable in advance, interacting with other programs or information sources as do scripts, or by virtue of anthropomorphizing the interaction to be perceived as "intelligent" or "believable".

The problem of bridging the gap between human intent and complex environmentally guided action by software agents remains open. Put broadly this is the question of how to make programming computers easy and accessible to the general public. The problem is not solved but various techniques including programming by demonstration, visual languages, context sensitivity, forms, and direct manipulation may offer partial solutions.

Constraining Expectation and Interpretation

A key factor in the success of direct manipulation is its ability to constrain intent (you can only select objects you can see) and action (you can only click on what you see) so that there is little ambiguity about what can be done or how to do it or what has been done. The asynchronous, semiautonomous behavior of a closed loop agent by contrast offers little in the way of affordances or feedback to constrain a user's expectations or evaluations. As a consequence the designer has not only the problems of automating actions but of conveying to user what actions are available, what goals they may satisfy, how to know that they have been performed, and how to verify that they have been performed successfully. These problems can be ameliorated by efforts to make the control of behavior apparent, understandable, or justifiable to the user. Anthropomorphism, mental models of agents, and explanation facilities can aid in reducing these ambiguities.

These problems of expectation are shared in common with human interaction with earlier forms of automation and intelligent systems and should benefit from accumulated experience in these areas.

Conventional Automation

Many issues affecting human interaction with software agents have already been explored in the context of conventional automation. Just as e-mail, the web, and burgeoning information overload are making the development of software agents necessary to allow information workers to continue to perform their jobs, increasing complexity in industrial systems such as nuclear power plants and aircraft long ago made them uncontrollable without automation. The point is that in excessively high workload domains we delegate tasks to machines not merely as a labor saving convenience but because it is necessary to do so. The model of a subservient if intelligent machine waiting at our beck and call may not fit these high workload environments in which a human failure to initiate automated action can be just as detrimental as an inappropriate action taken by a machine.

Autonomy

The broadest issue raised by necessary automation involves the relative autonomy of the human and machine. Relative autonomy has effects in areas ranging from acceptance and willingness to use automation to the stability and effectiveness of the resulting human-machine systems. While at least one party needs autonomy, systems in which the machine has the autonomy or both human and machine act independently are possible.

The basic issues affecting trade-offs in autonomy are:

1) the degree to which automation requires explicit initiation (machine autonomy)

> Starter and Woods (1995) classify errors involving automation as errors of *commission* or *omission*. Their error of *commission* refers to events in which the human fails to engage automation, engages it in the wrong mode, or engages it with inappropriate settings. The failure to spell-check a document, the unintended change of "not_old" to "not_new", or the Unix command "rm -r *" (recursive

delete) issued in lieu of "rm -f" which merely suppresses commentary would be examples of errors of commission.

2) the degree to which automation provides explicit feedback

The problem of adequate feedback is the complement of that of control. A basic precept of human engineering is that human errors cannot be completely eliminated. The reliability of human-machine systems therefore depends crucially on providing opportunities for humans to recover their mistakes. When automation intervenes between the human and the domain the ability to detect and hence correct errors may be impaired or eliminated.

If a situation has high workload and the user needs to delegate tasks to automation the detail and timing with which the automation provides feedback needs to be established. One approach followed in process control and aviation is to have automation report back only when and if some predetermined setpoint is exceeded. This has worked well in process control where engineers can carefully predetermine the thousands of setpoints. In aviation where response times are shorter and programming more flexible results have been less pleasing with numbers of controlled flights into terrain (Corwin, Funk, and Bloomfield 1993, Starter & Woods 1994) attributable to failures of pilots' interactions with their automated "servants". Errors caused by omission (Starter and Woods 1995) of feedback about initiation are becoming increasingly evident in highly automated modern environments such as the glass cockpit of the Airbus A-320. In errors of omission the automation initiates action without alerting the human, the human fails to recognize what the automation has done, and failures occur because of this loss of situational awareness.

Jones and Mitchell (1995) propose a set of prescriptive principles for resolving these issues in the human's favor:

Human Authority
human retains authority and responsibility

Mutual Intelligibility
human and machine maintain accurate model of one another and the domain

Openness and Honesty
machine's behavior is observable, machine communicates intentions, machines capabilities and limitations are made explicit

Management of Trouble
machine possesses strategies to recognize and repair losses of mutual intelligibility

Multiple perspectives
machine provides multiple perspectives to support alternate human problem solving strategies.

Although this list begins with a nod to human sovereignty, it quickly makes clear that the basic problem afflicting our interactions with automation is not a struggle over authority but a lack of understanding and predictability of the automation. The primary

concerns involving autonomy, initiative and feedback, also reflect this problem of human-machine-domain coherence. Referring to our reference model it is easy to see why these problems arise. When automation is imposed between the user and the domain it exacerbates both the gulfs of action and evaluation. Where automation is allowed initiative (adaptive agents), the user's evaluation becomes problematic because it may be difficult or impossible to disentangle the effects of the machine's actions from her own. Even more troubling, the attentional tuning between actions and their effects is lost so the user may utterly fail to notice significant changes in the domain. Even when the user retains initiative (semiautonomous agents) he is cut off from the domain by the intervening action/evaluation loop of the automation. The task is now, not simply to plan, execute, monitor, and interpret but to express these behaviors as an intention and then monitor performance indirectly. The alarm systems of a nuclear power plant or the "return" message from an executing shell script are examples of this sort of second order monitoring.

Trust

Many of the complex of issues involving mutual human-machine modeling, awareness, and coordination are captured by the anthropomorphic term trust. If we examine the considerations that enter into our decision to delegate a task to a subordinate, instruct the subordinate in how to perform the task, monitor that performance, or authorize some class of tasks without follow-up, our trust in the subordinate will almost certainly play an explanatory role. Closer consideration will show our use of the term to be multidimensional. The trust we have that our secretary will remember to pick up the mail, is distinct from our trust that he/she will compose a postable business letter, which in turn is distinct from our trust in the lawyer who assures us that the letter is not actionable.

Bonnie Miur (1987, 1994, 1996) adopted a taxonomy of trust for human-machine relations from sociologist Barber (1983) giving a nod to social psychologists (Rempel, Holmes, and Zanna 1985) for a complementary taxonomy and a source of conjectures about the dynamic character of trust. Barber (1983) defines trust in terms of three specific expectations:

1) *persistence* of natural, biological, and social "laws"; for example, gravity, pain following injury, and parents protecting their offspring

2) *competence* of others to perform their technical roles; for example, our trust that a bus driver will take us safely to our stop

3) *fiduciary responsibility* of others to fulfill their obligations; for example, our trust that a lawyer will administer an assigned estate without theft

Rempel et al. (1985) propose a similar taxonomy to account for couples' attributions about one another's behavior. Initially trust in a partner depends on predictability in their behavior such as a preference for sea food or punctuality for dates. As the relationship matures trust extends to the longer term traits of dependability in matters such

as remembering birthdays or picking up the laundry. In its final phase, trust extends to faith in the partner's affections and allegiance. As trust deepens its bases and expectations shift from the instrumental and observable to attribution and unobservable traits. Rempel et al are concerned with both the acquisition and resistance to extinction of these forms of trust. The faith in an alcoholic partner's affections, for instance, may last long after any hopes of consistency or dependability are gone. The two taxonomies can be approximately merged as advocated by Lee and Moray (1992) as distinguishing:

1) trust which is based on observed consistency of behavior (*persistence* or *predictability*)

I trust my watch to keep relatively accurate time

2) trust which is based on a belief in competence or well formedness (*competence* or *dependability*)

I trust Martha Stewart's recipe for hollandaise

3) trust which is based on faith in purpose or obligation (*fiduciary responsibility* or *faith*)

I trust my physician to monitor my health

The match up is somewhat inexact with Barber (1983) breaking out competence into *expert knowledge*, *Technical facility*, and *everyday routine performance* which are essentially the same as our higher level synthetic categories. Miur (1987) goes on to equate these three levels of competence with Rasmussen's (1983) taxonomy of skill-based, rule-based, and knowledge-based behaviors. Lee and Moray (1992) propose an additional correspondence to Zuboff's (1988) stages of *trial and error experience* (consistency), *understanding* (competence), and *leap of faith* (fiduciary responsibility) in developing operators' trust in new technology. A further analogy can be drawn to Dennet's (1987) description of three stances people may adopt to predict the behavior of systems:

physical stance prediction based on physical characteristics and laws (persistence & predictability)

design stance prediction based on what a system was designed to do (competence & dependability)

intentional stance prediction based on an assumption of rationality (fiduciary responsibility or faith)

The convergence of five independent taxonomies purporting to describe how humans or machines behave or can be predicted to behave supports the appeal of this approach.

As bases for human modeling of machine agents these taxonomies suggest that agents can be made predictable by: 1) consistently pairing simple observable actions with inputs, or 2) making the causes and rules governing an agent's behavior

transparent or 3) making the purpose, capability, and reliability of the agent available to the user. Miur (1987, 1994, 1996) refers to the process of acquiring predictive models of these sorts as *trust calibration*. The idea being that performance will be better for human-machine systems in which trust is accurately calibrated because the human's model will allow more accurate predictions. Disregarding issues of acquisition and extinction trust based on observable consistency or competence should yield more accurate prediction than faith in a black box's performance. The greater predictability of consistent or competent agents should also make boundary conditions and "brittleness" more apparent and remediable. Agents trusted on faith, by contrast, would require a very high degree of reliability across their range and more communication to maintain accurate coordination. Experiments in our laboratory (Lenox, Roberts, and Lewis 1997, Sycara, Lewis, Lenox, and Roberts 1998) support these hypotheses finding that although subjects interacted more extensively with an opaque agent requiring faith, an agent of observable competence led to substantially better target identification and resolution of encounters.

Trust and Experience

Miur's (1987, 1996) and Lee & Moray (1992) experiments address the effects of errors on ratings of trust and willingness to use automation. Miur (1987, 1996) reports a large constant error (predictable) had approximately the same impact on ratings of trust as a much smaller variable error as the level of trust model would suggest. Both Miur (1987, 1996) and Lee & Moray (1992) found correlations between ratings of trust and reliance on automation, however, recent studies of pilots (Reiley 1996 and Singh, Molloy and Parasuraman 1993) found no relation between attitudes toward automation and reliance on automation in a laboratory task. Time limits on laboratory experiments make it difficult to test Rempel et al.'s (1985) and Zuboff's (1988) hypotheses about developmental stages of trust. Observation of new and experienced users of real systems, however, provides an opportunity to examine these predictions.

Contrary to conventional wisdom, human autonomy and control is not always preferred by experienced users of automation (Parasuraman 1997). Reily (1996), for example, found that pilots were more reliant on automation than college students. In the past year I have conducted a series surveys of University of Pittsburgh students examining their use of automated features of popular software. Contrary to my expectation that experienced users would prefer options giving them greater control, I have found that it is the beginning users who prefer high levels of control while the more experienced ones prefer to relinquish control to the system. In use of Micro Soft Word's "replace text as you type" spell correction option, for example, I found that over half of the experienced users chose it while only twenty-two percent of the less experienced users left it on. In agreement with our conjecture that successful black box automation requires explicit attention to feedback, the experienced users overwhelmingly chose the message for confirming changes while the inexperienced rejected it. Use of the printer icon in preference to the menu print option which brings up a dialog box allowing customization of job showed similar results. Approximately seventy percent of the experienced users chose the icon while a similar proportion of the novices preferred the control offered by the menu selection. While nonlongitudinal

observations of this sort cannot trace the development of trust, the preference for control by new users and the willingness to relinquish it by the experienced suggests such a progression.

Effort and Individual Differences

Return on investment is a recurring theme in human use of automation. Very often the functions most easily automated such as level flight are those which need it least. In fact, automation is often perceived as increasing not decreasing workload. Wiener (1985, 1989), for example, investigated pilots attitudes toward cockpit systems. He found that only a minority agreed with the statement "automation reduces workload" while a substantial proportion felt that automation, especially programmable systems such as the Flight Management System (FMS) increased their workload. Early expert systems provide another example of automation that increases work. Miller and Masarie (1990) report that INTERNIST-1 often required 30 to 90 minutes of data gathering and interaction to complete a single diagnostic consultation. Almost any form of procedural instruction has the potential of becoming more work than doing it by hand and in a choice environment where automation is optional or can be turned off or disabled, it often will be.

In a recent review of human use of automation Parasuraman (1997) identifies three strategies for better using automation. The first of these "Better operator knowledge of how automation works" deals with the predictive model that has been discussed in terms of "trust". The second strategy requires anticipating and designing for "large individual differences (which) make systematic prediction of automation use.. difficult". The third strategy holds that automation must not require a high level of cognitive overhead.

Extending this guidance to agents is a daunting task. Supporting trust requires that they either be very simple and observable, intelligible and make their interior processes and products visible, or highly reliable with explicitly managed communications. Agents must be designed at varying levels of sophistication to accommodate inter and intra individual differences and must be so easy to instruct that it is no more difficult than doing the task ourselves. These same key points: cost, modeling and trust, and adaptivity are stressed by Milewski and Lewis (1997) in their review of the human automation and agent literature.

One clear conclusion of this review of psychological literature involving human trust of automation is that anthropomorphism can do little to improve human agent interaction in the near term. Faith, the only form of trust associatable with a presentation mimicing human behavior, is the least helpful in terms of coordinated behavior and the most difficult to acquire. Transparent forms of presentation which make the competence and rules of agent behavior apparent are more promising for improving performance and establishing trust.

Human Agent Interaction

Norman’s human reference model organizes agents into three broad classes: anticipatory agents, filtering agents and semiautonomous agents. Each class presents its

own design challenges to promoting effective human interaction. In this section representative examples of each of these classes will be discussed.

Anticipatory and Filtering agents

Anticipatory agents are classified by our model as those which automate some portion of the action execution cycle. Although an anticipatory agent may have significant information gathering interactions with the domain this information is gathered in order to determine a context used to infer the user's likely intent. Anticipatory agents are a sort often portrayed in fiction and typified by an intimate acquaintance with their user's intentions and preferences like an experienced butler. Since these intentions and preferences may be quite idiosyncratic, complex, and difficult to express, anticipatory agents are often constructed as "learning agents." As such their task is not only to act in accordance to what is probably their user's intention but to learn mappings from the software context and sequences of user actions to the subsequent action.

Our distinction between action anticipation and filtering is somewhat artificial because the learning methods used and "actions" learned may be very similar. This very similarity emphasizes the cyclical character of human relations to domains and the inseparability of action in directing attention and evaluation in determining subsequent actions. Anticipatory agents can be grouped into two broad classes, those which perform a simple concept learning task to mimic a human user's categorization scheme and those which seek to infer a user's plan of action by observing a more complex sequence of actions in order to recognize the plan they are instantiating. Lesh and Etzioni (1995) attack this second sequential decision problem of plan recognition for performing simple tasks using the UNIX operating system such as finding and printing a file. After each observed command the set of possible plans supported by the sequence decreases. This problem which is exponentially large in the number of goal predicates illustrates just how difficult tasks appearing "easy" to humans can become. Deployed and tested agents which learn a user's "intention" are overwhelmingly of the simple concept learning sort.

CAP (Calendar APprentice), (Mitchell, Caruana, Freitag, McDermott, Zabowski 1994) an agent that learns room scheduling preferences, exemplifies the action learning approach. CAP assists the user in managing a meeting calendar by providing an online calendar and email facility. Users can enter and edit meetings on the calendar and instruct CAP to send invitations. CAP observes user actions "over the shoulder" and as it acquires a ruleset for predicting durations, locations, times, and weekdays of meetings, begins making "suggestions" to the user. Performance of learned rules is tracked and as they are validated and new and hopefully more robust rules are learned the agent will be able to make increasingly useful suggestions. Even the relatively stable environment of Carnegie Mellon room scheduling had enough variance to stymie CAP a good deal of the time. Plots of prediction accuracy show a clear term to term periodicity with accuracy falling to the 40th percentile a transitions and rising to 60% or above by the middle of the term. Referring to our reference model, CAP is an agent which automates the processes of intention formation and action planning. It is not autonomous because it requires a human action to initiate its suggestion process.

The go/no go option given the user for executing CAP's plan is appropriate given the observed accuracies. CAP's users can tolerate this level of unreliability because the system is consistent, relatively transparent, and provides explicit feedback (the suggestion). A similar meeting scheduling agent which doesn't specify the room to be used is reported by Kozierok and Maes (1993). Their calendar agent explicitly displays certainty factors providing more detailed feedback and a more favorable basis for reliability judgements.

The most popular task for anticipatory agents is the simple concept learning problem of learning a human's categorization scheme. Using a feature vector extracted from a source such as email headers a concept learning algorithm such as ID3 is used to replicate that categorization. New instances can then be sorted into the learned categories. Agents of this sort frequently bridge the gap between anticipation (automation of action) and filtering (automation of evaluation) because after classifying (matching the user's intent) they may go an additional cycle by accepting this implicit evaluation and executing a subsequent action such as deleting the message. It makes no difference to the concept learning algorithm which can as easily categorize incoming messages as "buy IBM" or "invade Iraq" as the prototypical classification, file in folder 'A' or 'B'. The difference comes on the other side of the interface where a user may need explicit feedback or transparent justification for automating actions to adapt to the agent without concern.

The classic concept learning agent is the mail or news agent which learns to file, delete, prioritize, and notify a user of incoming mail. Maxims (Lashkari, Metral, and Maes 1994), for example, performs all these functions after observing how a user chooses to deal with email using the same mechanisms as the MIT scheduling agent (Kozierok and Maes 1993). Other systems of this sort include Magi (Payne and Edwards 1997) a learning agent which runs behind the Xmail reader, and a news story categorizer used with an online edition of a campus newspaper (Gustafson, Schafer, and Konstan 1998).

While agents discussed so far have used learning to adapt to and anticipate the intentions of an individual user, a similar class of systems does just the opposite. If one presumes that an individual's preferences will be similar to that of similar individuals a system can learn from a large sample of interactions (for robust learning) with different individuals without extensive and error prone trials for any one person. This approach, known as collaborative filtering has spawned a variety of systems. Some filter "manually" by making available comments and ratings for example helping people select corporate documents by annotating their reactions (Goldberg, Nichols, Ok, and Terry 1992), or rating articles in NetNews (Resnick, Iacovou, Suchak, Pergstrom, and Riedl 1994). Others act as learning agents by comparing a user's profiles with those on file and estimating preferences on that basis, Ringo Maes' (1994) music recommendation system for example. Krulwich's (1997) Lifestyle finder takes this approach to its logical conclusion by using 62 demographic clusters identified by market researchers as a starting point.

A common thread across these varied systems is the use of inductive learning and feedback to automate the process of expressing our goals and intentions to the

computer. The results are both impressive and humbling. Where learning agents are expected to do more than recommend movies and music their inaccuracies become apparent. Mechanisms such as Maes' user selectable "tell me" and "do it" thresholds or displayed certainty factors provide opportunities for partial learning to put its best foot forward. As a consequence, for a given degree of learning agent reliability is improved and trust is more easily granted. Strategies of suggestion rather than action such as followed by CAP are another adjustment necessary at these levels of reliability.

Instructions and Semi-Autonomous Agents

Just as friends and spouses cannot always guess what we want unless we tell them, it is unreasonable to presume that primitive inductive learning methods will empower agents to do so. Many of the routinized cognitive tasks we would like to delegate to agents involve relatively complex sequences of behaviors, such as extracting particular information from a site on the web, monitoring a news group for a recent topic or performing some set of conditional actions. What is significant about this list is that these requests all involve sequences of actions which although individually constrained combine to form unlearnable wholes. The problem of communicating these desires directly, however, is precisely the problem of end user programming that intelligent agents were supposed to resolve.

Actions which are removed in time and place from the user cannot be learned by observing and must be explicitly requested. The most commonly encountered instructable agents of this sort are the notification agents found at many web sites. These agents typically use a form based interface to allow a visitor to enter an event of which he/she wishes to be notified such as a sale on airfares. When the event occurs the agent is triggered and notification sent by email.

Potential remedies for more sophisticated instructions are the usual suspects: high level scripting languages, forms and direct manipulation interfaces, programming by demonstration, and visual languages. Each approach has limitations which make it difficult for use in instructing semiautonomous agents. Scripting languages are difficult to learn and use. Much of the functionality of branching, looping, and running asynchronously needed by semiautonomous agents is expressible but takes too much effort. Scripting languages are far more general than the constrained sets of tasks one might wish performed by an agent, yet if all the scripts likely to be needed were written a long reference manual and commands with too many parameters to understand would result. If direct manipulation were chosen the accessibility problems are solved but at the cost of an impoverished language with deficits in just the areas, branching, looping, asynchrony needed.

A notable approach combining the power of scripting and the accessibility of direct manipulation and form filling is Etzioni and Weld's (1994) Softbot. The user interacts through a form based interface with a planning program which taking the user's goal as an input, searches a library of action schema to generate a sequence of actions achieving that goal. The use of goals as inputs and action schemata as agent primitives makes this agent consistent with our reference model and consistent with

our definition of an agent as a process automating stages of this model.

Programming by demonstration also appears a good candidate for instructing agents although not in the programming of animation as in KidSim Smith, Cypher, and Spohrer (1997) or Bitpict Furnas (1991). Accepting Maes' (1997) characterization that "an agent basically interacts with the application just like you interact with the application" programming by demonstration in the fashion of emacs macros would achieve just this effect. This is similar in spirit to over the shoulder learning but not subject to the pitfalls of pure inductive learning.

The final option, a visual language, seems particularly promising for instructing semiautonomous agents. Specialized applications in which programs can be assembled from modules (such as the action schema and goals from Etzioni and Weld 1994) are particularly suited to visual languages. LabVIEW provides such a programming environment for signal processing where logic and signal processing can be specified by laying out and connecting icons. A visual language for semiautonomous agents might allow the user to associate reusable behaviors such as communications, remote accesses, or asynchronous responses with appropriate goals in a similar fashion. Referring to the reference model, this explicit specification of the relation between goals and actions allows a nearly "transparent" means of instruction.

Two current systems, the Agent editor of the Retsina system (Sycara, Decker, Panu, Williamson, and Zeng 1996) and Thurman, Brann, and Michell's (1997) Apprentice extension to OFMspert (Operator Function Model) follow this approach. The key component of the Retsina reusable agent architecture is a hierarchical representation of task-subtask relationships. Temporal constraints between nodes are used to handle task planning and execution. The agent's plan library contains skeletal plans and plan fragments (cases) that are indexed by goals and can be retrieved and instantiated according to the current input parameters. The retrieved and instantiated plan fragments are used to form the agent's task tree that is incrementally executed. The formalism can also handle sophisticated control structures, such as deadlines, looping and periodic tasks. The scheduling module schedules each of the plan steps taking as input the agent's current set of executable actions, and deciding which action, if any, is to be executed next. The communication module accepts and interprets messages from other agents. The agent editor allows its user to visually construct and manipulate task trees to assemble agents having customized capabilities and behaviors.

The OFMspert Apprentice uses a similar mechanism for editing the relation between goals and actions to allow a user to improve the performance of an expert system. The system performs the task until it encounters some impasse or error condition at which time it summons its operator. The operator "solves" the problem and "programs" the system to deal with similar impasses. The OFMspert operations model consists of a model of the system being controlled, an activity model of operator activities, and a plan archive indexing activity trees. The OFMspert Apprentice is a computational implementation of the operations model. The Apprentice is designed as a vehicle for incremental automation of the control task. The plan executor and monitor enable an operator to observe the plan in action and repair minor problems which arise by making changes to displayed activity trees. Apprentice is being developed to

support operations automation in satellite ground control.

Both the Retsina editor and the OFMspert Apprentice have the properties identified as desirable for human interaction with powerful agents. Both visually specify actions through goal based trees and allow users to instruct the agent at this level. Despite their sophistication both agents are transparent and their competence and predictability can be inspected from the activity trees. Rather than relying on anthropomorphism faith and uncorrectable behaviors these agents present themselves as they are and promise to be more useful as a result.

References

Barber, B. 1983. The Logic and Limits of Trust. New Brunswick, NJ: Rutgers University Press.

Bradshaw, J. 1997. Software Agents, Cambridge, MA: AAAI Press/The MIT Press.

Card, S., Moran, T., and Newell, A. 1983. The Psychology of Human-Computer Interaction. Hillsdale, NJ: Lawrence Erlbaum.

Corwin, W., Funk, H., Levitan, L. and Bloomfield, J. 1993. Flight Crew Information Requirements (Contractor Rep. DFTA-91-C-00040). Washington, DC: Federal Aviation Administration.

Dennet, D. 1987. The Intentional Stance. Cambridge, MA: The MIT Press.

Etzioni, O. and Weld, D. 1994. A Softbot-Based Interface to the Internet, Communications of the ACM 37(7):72-76.

Franklin, S. and Graesser, A. 1996. Is It an Agent or Just a Program? A Taxonomy for Autonomous Agents. In Proceedings of the 3rd International Workshop on Agent Theories, Architectures, and Languages. New York, NY: Springer-Verlag.

Furnas, G. 1991. New Graphical Reasoning Models for Understanding Graphical Interfaces. In Proceedings of Human-Computer Interaction ’91, 71-78. New York, NY: ACM Press.

Gilbert, D. 1997. Intelligent Agents: The Right Information at the Right Time. IBM white paper (http://www.networking.ibm.com/iag/iaghome.html). Research Triangle Park, NC: IBM Corporation.

Goldberg, D., Nichols, D., Ok, B., and Terry, D. 1992. Using Collaborative Filtering to Weave an Information Tapestry. Communications of the ACM 35(12):61-70.

Gustafson, T., Schafer, B., and Konstan, J. 1998. Agents in their Midst: Evaluating

User Adaptation to Agent-Assisted Interfaces. In Proceedings of 1998 International Conference on Intelligent User Interfaces, New York, NY: ACM Press.

Jones, P. and Mitchell, C. 1995. Human-Computer Cooperative Problem Solving: Theory, Design, and Evaluation of an Intelligent Associate System, IEEE Transactions on Systems, Man, and Cybernetics. 25(7):1039-1053.

Kozierok, R. and Maes, P. 1993. A Learning Interface Agent for Scheduling Meetings. In Proceedings of ACM SIGCHI International Workshop on Intelligent User Interfaces, 81-88. New York, NY: ACM Press.

Krulwich, B. 1997. Lifestyle Finder: Intelligent User Profiling Using Large-Scale Demographic Data. AI Magazine. 37-45.

Lanier, J. 1995. Agents of Alienation. Interactions Jul., 66-72.

Lashkari, Y., Metral, M., and Maes, P. 1994. Collaborative Interface Agents. In Proceedings of the National Conference on AI. Cambridge, MA: The MIT Press.

Laurel, B. 1997. Interface Agents: Metaphors with Character. In Software Agents, ed. J. Bradshaw, 67-77. Cambridge, MA: AAAI Press/The MIT Press.

Lee, J. and Moray, N. 1992. Trust, Control Strategies, and Allocation of Function in Human-Machine Systems. Ergonomics 35(10): 1243-1270.

Lenox, T., Roberts, L. and Lewis M. 1997. Human-Agent Interaction in a Target Identification Task. In 1997 IEEE International Conference on Systems, Man, and Cybernetics, 2702-2706. Orlando, FL: IEEE.

Lesh, N. and Etzioni, O. 1995. A Sound and Fast Goal Recognizer. In Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1704-1710. Menlo Park, CA: International Joint Conferences on Artificial Intelligence.

Maes, P. 1994. Agents that Reduce Work and Information Overload. Communications of the ACM 37(7): 31-40.

Milewski, A. and Lewis, S. 1997. Delegating to Software Agents. International Journal of Human-Computer Studies 46(4): 485-500.

Miller, R. and Masarie, F. 1990. The Demise of the "Greek Oracle" Model for Medical Diagnostic Systems. Methods of Information in Medicine 29(1): 1-2.

Mitchell, T., Caruana, R., Freita, D., McDermott, J., and Zabowski, D. 1994. Experience with a Learning Personal Assistant. Communications of the ACM 37(5): 81-91.

Miur, B. 1987. Trust Between Humans and Machines, and the Design of Decision Aids. International Journal of Man-Machine Studies 27(5&6): 527,539.

Muir, B. 1994. Trust in Automation: Part I. Theoretical Issues in the Study of Trust and Human Intervention in a Process Control Simulation. Ergonomics, 39(3): 429-460.

Negroponte, N. Being Digital. New York, NY: Alfred Knopf.

Norman, D. 1988. The Psychology of Everyday Things. New York, NY: Basic Books.

Norman, D. 1986. Cognitive Engineering. In User Centered System Design: New Perspectives on Human-Computer Interaction, eds. D. Norman and S. Draper, 31-61. Hillsdale, NJ: Lawrence Erlbaum.

Nwana, H. 1996. Software Agents: An Overview. Knowledge Engineering Review 11(3): 205-244.

Parasuraman, R. 1997. Humans and Automation: Use, Misuse, Disuse, Abuse, Human Factors 39(2): 230-253.

Payne, T. and Edwards, P. 1997. Interface Agents that Learn: An Investigation of Learning issues in a Mail Agent Interface, 11(1): 1-32.

Rasmussen, J. 1983. Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering. New York, NY: North-Holland.

Reiley, V. 1996. Operator Reliance on Automation: Theory and Data, In Automation and Human Performance: Theory and Applications. eds. R. Parasuraman and Mouloua. 19-35. Hillsdale, NJ: Erlbaum. rf Rempel, J., Holmes, J., and Zanna, M. 1985. Trust in Close Relationships. Journal of Personality and Social Psychology 49(1): 95-112.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work. 175-186. New York, NY: ACM Press.

Shneiderman, B. and Maes, P. 1997. Direct Manipulation vs. Interface Agents. Interactions (4)6: 43-61.

Singh, I., Molloy, R., and Parasuraman, R. 1993. Individual Differences in Monitoring Failures of Automation. Journal of General Psychology. 120(3): 357-373.

Smith, C., Cypher, A. and Spohrer, J. 1997. KidSim: Programming Agents Without a

Programming Language. In Software Agents. ed. J. Bradshaw. 165-190. Cambridge, MA: AAAI Press/The MIT Press.

Starter, N. and Woods, D. 1994. Pilot Interaction with Cockpit Automation II: An Experimental Study of Pilot's Model and Awareness of the Flight Management System. International Journal of Aviation Psychology. 4(1): 1-28.

Starter, N. and Woods, D. 1995. From Tool to Agent: The Evolution of (Cockpit) Automation and its Impact on Human-Machine Coordination. In Proceedings of the Human Factors and Ergonomics Society 39th Annual Meeting. 79-83. Santa Monica: CA: HFES.

Sycara, K., Decker, K., Panu, A., Williamson, M. and Zeng, D. 1996. Distributed Intelligent Agents. IEEE Expert. 11(6): 36-46.

Sycara, K., Lewis, M., Lenox, T., and Roberts, L. 1998. Calibrating Trust to Integrate Intelligent Agents into Human Teams. In Proceedings of the Hawaii International Conference on Systems Science. Kona, HI:HICSS.

Thurman, D., Brann, D., and Mitchell, C. 1997. An Architecture to Support Incremental Automation of Complex Systems. In 1997 IEEE International Conference on Systems, Man, and Cybernetics, 2702-2706. Piscataway, NJ: IEEE.

Wiener, E. 1985. Beyond the Sterile Cockpit. Human Factors, 27: 75-90.

Wiener E. 1989. Human Factors of Advanced Technology ("Glass Cockpit") Transport Aircraft, Technical Report 177528, NASA Ames Research, Moffett Field, CA.

Zuboff, S. 1988. In the Age of the Smart Machine: The Future of Work and Power. New York, NY: Basic Books.