Computers as people: human interaction metaphors in human-computer interaction

by Benjamin Fineman

A thesis submitted to candidacy for the degree of
Master of Design in Interaction Design

The School of Design
Carnegie Mellon University

_____

Benjamin Fineman

_____

Chris Neuwirth, Advisor

Pittsburgh, Pennsylvania
May, 2004

I would like to thank Chris Neuwirth for her insight, suggestions and support, Jonathan Clemmer for his patience, and my mother for her inspiration.

Introduction

In the movie *2001: A Space Odyssey* (Kubrick 1968), crewmembers of a spaceship interact with the ship's computer by talking to it. The computer, named HAL, has no keyboard or mouse; the only way to communicate with it is conversationally. HAL behaves the same way people do in conversation, using subtle strategies such as hedging, indirect requests, and politeness to accomplish its goals. The crewmembers also converse with HAL in the same way they converse with each other, using natural language and social courtesy. Judging by the way the rest of the technology on the ship is presented, the viewer is led to believe that this conversational capability is a natural extrapolation of the computer trends of the sixties and would not be implausible by the year 2001, just 33 years after the release of the movie.

Why would the makers of *2001*, in an era before graphical user interfaces, when computers were controlled by command lines and punch cards, see a conversational computer as plausible and perhaps even inevitable? In HAL's case, part of the motivation was probably simple dramatic effect. A conversational computer implies intelligence, and spoken dialog conveys emotion and drama better than typing at a keyboard. In addition, when HAL goes out of control, the fact that the computer has no tangible physical presence adds to the menace — in order to get anything done on the ship the crewmembers have to talk to the computer as they would to another human being, submitting in a sense to its agency. But HAL is not alone; science fiction

is full of conversational computers, from the ship's computer on the television series *Star Trek* to C3PO in the movie *Star Wars*, to a countless number of B-movie robots. Why does the idea seem so compelling and so believable?

I believe part of the answer lies in our everyday experience with real-world computers. Computers, like people, seem to have agency — they act on their own, according to an agenda, and sometimes appear to behave unpredictably. We ask them to do things for us and trust in their agency to complete the task. One could argue, quite correctly, that computers possess no agency at all since they only process instructions written by human programmers. But in the experience of using a computer, its behavior *suggests* agency, even though we know it's only a machine. Furthermore, computers communicate to users aspects of their internal state, actions, and knowledge. When people need to communicate such information, they typically use conversation. In some sense, then, computers are like people, and certain types of human-computer interaction are like conversations. One reason science fiction conversational computers seem so plausible is that many human-computer interactions are *metaphorically* conversational, and we're used to thinking about them (often unconsciously) in terms of a metaphor of conversation.

This paper examines the metaphor of conversation and other human interaction metaphors as they apply to human-computer interaction (HCI).[1] Rather than analyzing science fiction, I focus on uses of these metaphors in the theory and practice of designing real-world interactive computer systems. To lay the groundwork, I first summarize the current theory of metaphor's role in human cognition, then outline different uses of metaphor in HCI. After placing human interaction metaphors within a

[1] For the purposes of this paper human-computer interaction encompasses much more than a person sitting in front of a screen at their desk using a keyboard and mouse. Any product with a microchip can be considered a computer and falls under the HCI domain. In addition to the obvious computer chips in cell phones, VCRs and other consumer electronic devices, computers are also embedded in toys, buildings, and even clothing. Many of these computers are invisible to us — when we press the brake pedal in our cars, for example, we don't typically think about the computer that actually stops the car (Norman 1998).

larger structure of HCI metaphor, I then summarize work that researchers and designers have done with conversational human-computer interactions. Finally, I highlight one special case of human interaction metaphor — working relationships — and argue for the benefits of designing interactive products using a working relationship model.

The contemporary theory of metaphor

Many people think of metaphor as a literary device, an expressive or poetic departure from literal, concrete, everyday language — when speaking, one can either be literal and factual, or metaphoric and fanciful. Current scholarship suggests that this distinction is false. Not only is metaphor used in everyday factual language, but it is also a fundamental part of human cognition.

During the 1980s, a new theory of metaphor emerged among linguists and cognitive scientists. According to this theory, metaphor is the process of mapping a set of correspondences from a source domain to a target domain. These correspondences let us reason about the target domain using the knowledge we have about the source domain (Lakoff 1993). For example, we often think of life as a journey, with birth as the beginning and death as the end. Major life goals are destinations, problems are roadblocks or impediments in reaching those destinations, and major life choices are forks in the road. We can be off course, behind schedule, spinning our wheels, or going in circles. These aren't just colorful descriptive terms, but point to a deeper conceptual mapping that structures how we think and reason about our lives.

Metaphor highlights some aspects of the target domain while hiding others. For this reason we often use simultaneous and inconsistent metaphors for important concepts, and for good reason according to Lakoff and Johnson:

> Our conceptual systems have inconsistent metaphors for a single concept. The reason is that there is no one metaphor that will do. Each one gives a certain comprehension of one aspect of the concept and hides others. To operate only in terms of a consistent set of metaphors is to hide many aspects of reality…. The use of many metaphors that are inconsistent with one another seems necessary for us if we are to comprehend the details of our daily existence. (Lakoff and Johnson 1980)

In addition to viewing life as a journey, for example, we also use the metaphors LIFETIME IS A DAY, with death as sunset, LIFETIME IS A YEAR, with death as winter, LIFE IS A STRUGGLE, with death as defeat, and DEATH IS DEPARTURE (Lakoff 1993). Rather than causing cognitive dissonance, multiple metaphors allow us to focus on different aspects of life according to the needs of the particular context.

The role of metaphor in design

The shift from viewing metaphor as a simple linguistic expression such as "he's spinning his wheels" to seeing it as a systematic mapping of many correspondences explains the power of metaphor for creative disciplines such as science and design. The underlying metaphoric system, LIFE IS A JOURNEY, enables people to understand many linguistic expressions, not just "he's spinning his wheels," as part of a coherent framework. More importantly, it allows people to invent and understand new expressions based on the same conceptual system (Lakoff and Johnson 1980). So if I were to say, "he missed the exit for a career change several miles back," you might think the expression awkward, but you could easily understand the metaphorical meaning.

Scientists often use metaphor to model and describe new concepts, such as the solar system model of the atom. Not only does the metaphor allow scientists to understand and explain new concepts, but they can also propose new extensions to the metaphor, and then use the validity of those extensions to test the appropriateness of the underlying metaphoric structure. In addition, multiple or hybrid metaphors, such as the wave and particle theories of light, allow scientists to focus on different aspects of the concept in cases where the target domain is more complicated than any one metaphor can describe. Although scientists still debate metaphor's value as a scientific tool, many have accepted metaphor as a necessary and even useful part of the scientific process (Brown 2003).

In design, as in science, the role of metaphor is a subject of some debate. Some designers point to interfaces built around awkward and over-extended metaphors as a sign of problems inherent in the metaphor approach (eg, Cooper 1995).

Others view metaphor as an integral part of the design process (eg, Heckel 1991). In order to sort out these different claims, we first need to make several distinctions between different types of metaphor. Table 1 provides an overview of the distinctions I will discuss in this section.

Table 1: HCI metaphor distinctions

Purpose of metaphor	• Familiarizing • Transporting • Invention
Domain of metaphor	• Functionality • Interface • Interaction
Awareness of metaphor	• Tacit • Explicit

First, designers use metaphor for three different purposes: familiarizing, transporting, and invention (Heckel 1991; Schön 1993). *Familiarizing metaphors* make a product or interface easier to understand by creating correspondences with a more familiar domain (Heckel 1991). The desktop metaphor, for example, leverages users' experiences with paper files and folders to familiarize the mechanism of organizing documents. Based on prior experience with filing in office environments, new computer users can make assumptions about digital "files" and "folders" that they couldn't make about directories, catalogs and b-trees.

Unlike familiarizing metaphors, which simply help users understand and learn a product, *transporting metaphors* allow them to view and solve problems in new ways. Transporting metaphors unify or generalize, collecting individual experiences into one conceptual framework, and allowing that conceptual framework to form the basis for new experiences (Heckel 1991). For example, a path metaphor allows computer users to "go back," retracing their steps in web browsers. The metaphor is transporting because it could be (and is) applied to a range of contexts such as database record

browsing, folder views in desktop interfaces, phone trees, and software installation wizards. Once learned, the metaphor becomes a tool users can apply to new interfaces and interactions.

In addition to familiarizing and transporting metaphors, which focus on the user's interaction with a product, designers sometimes use "throwaway" *invention metaphors* during the design process to help them come up with ideas. These invention metaphors might never be explicitly understood by the user, and aren't intended to be. Whereas many conventional metaphors operate tacitly, creative people explicitly invoke invention metaphors to help them see problems in new ways (Schön 1993). As part of the creative process, invention metaphors help designers "think outside the box," breaking away from conventional approaches to the product. So if designers think of a television as, for example, a babysitter, it suggests a set of correspondences that lead them to focus on aspects of the product they otherwise would not have noticed. In the babysitter example, the television might keep a record of watched programs for later parental review, shut itself off at a designated hour, or allow parents to monitor home activity remotely. Although many of these correspondences might be inappropriate, some of them will be useful, and the exercise brings a new perspective to the product. The end users of the television, on the other hand, might not find this particular metaphor useful, but if it leads to new design ideas it will have served its purpose. Designers may repeat the process, applying several invention metaphors to the same product.

I'd like to make a second distinction relevant to design: the distinction between functionality metaphors, interface metaphors, and interaction metaphors. This distinction describes the target domain of the metaphor. *Functionality metaphors* deal with what the product can do. Email, for example, uses a functionality metaphor based on an office environment: THE EMAIL SYSTEM IS INTEROFFICE MAIL. Email users reason about what email can do based on this metaphor; they expect to be able to send, receive, read and write messages, and they expect a service infrastructure with capabilities similar to an office mail system.

Many interactions go beyond the functionality of the real-world source domain, introducing "magic" into the system. Magic properties of interfaces give users shortcuts to do things that would be time-consuming or impractical (but usually not impossible) in the real world, such as automatically resorting the contents of a folder (Neale and Carroll 1997). Magic addresses one of the criticisms of metaphor in design: that it limits computer interfaces to inefficient simulations of the real world while ignoring new types of interactions made possible by the new technology. This focus on the real-world source domain, critics argue, prevents users from developing an adequate mental model of how the computer actually functions (Cooper 1995). However, some scholars suggest that properly-managed mismatches between the source and target domains, such as those introduced by magic, might help users develop a better mental model of the system by encouraging users to explore the points of difference between the two domains (Neale and Carroll 1997). Resorting the contents of a folder, for example, can be done in the real world, but not automatically. This magic feature of virtual folders does not contradict the primary metaphor, but instead adds a complimentary metaphor of assistance. In exploring magic capabilities, computer users begin to understand the differences between real-world and virtual folders, such as the ability to place virtual folders within folders within folders.

If functionality metaphors describe what the system can do, *interface metaphors* deal with the mechanics of how tasks are accomplished. Most email systems, for example, extend the physical aspects of real-world mail to the interface, providing an inbox and outbox, icons of envelopes, paper clip icons for "attachments," and so on. These interface elements are expressions of the underlying metaphor EMAIL IS PHYSICAL MAIL, providing email users with familiar cues and affordances based on their knowledge of physical mail. The choice of these particular interface elements supports, but is not determined by, the functionality metaphor. For example, email interfaces could choose to focus on the physical space of an office rather than the physical attributes of interoffice mail. Such an interface might have a "mailroom" that users would visit in order to send and receive mail rather than an inbox and outbox. One of

the challenges of interface design is the appropriate choice and representation of these metaphors.

It is important to remember that interface elements are not themselves metaphors but rather *expressions* of the underlying metaphor. An icon of an inbox has more to do with synecdoche than metaphor; it represents a real-world physical object, and that object in turn represents the entire interoffice mail system. The metaphor is the set of structural similarities between the mail system and the email program, not the iconic representation of the inbox. The icon builds on the metaphor, but over time may lose its metaphoric connection. As users become more familiar with the email program, for example, they may cease to equate receiving email with getting physical mail in their inbox.

If interface metaphors deal with the mechanics of action, *interaction metaphors* deal with the more abstract nature of its form. While the target domain for interface metaphors is the user interface of the computer, the target domain for interaction metaphors is the interaction between human and computer — the *type of relationship* they have. Interface metaphors apply to a specific interface, while interaction metaphors are task independent, enabling many different interfaces (Hutchins 1989, cited in Neale and Carroll 1997).[2] The EMAIL IS PHYSICAL MAIL interface metaphor, for example, relies on the concept of *direct manipulation*. Before email users can act on the interface elements described above, they must understand a more general interaction metaphor: DATA IS A PHYSICAL OBJECT. This metaphor allows users to "drag" files into folders or outboxes, "open" email messages, and "send" messages (objects) from their computer to someone else's.

The direct manipulation metaphor structures a particular relationship between human and computer, with the human as agent and the computer as a passive collec-

[2] Hutchins uses the term "mode of interaction metaphor," which I have shortened to "interaction metaphor." His other two categories — activity metaphors and task domain metaphors — don't map exactly to functionality and interface metaphors as I use the terms in this paper, although there are some similarities.

tion of objects waiting to be manipulated. The human manipulates these objects directly, without the need for an intermediary (Frohlich 1997). This metaphor is a cornerstone not only of email systems, but graphical user interfaces in general. Once users understand the interaction metaphor, they can apply it to a range of interfaces. While interface metaphors are often familiarizing, reducing the time needed to learn the mechanics of a new system, interaction metaphors are usually transporting, providing a generalized relationship that can be used in many contexts.

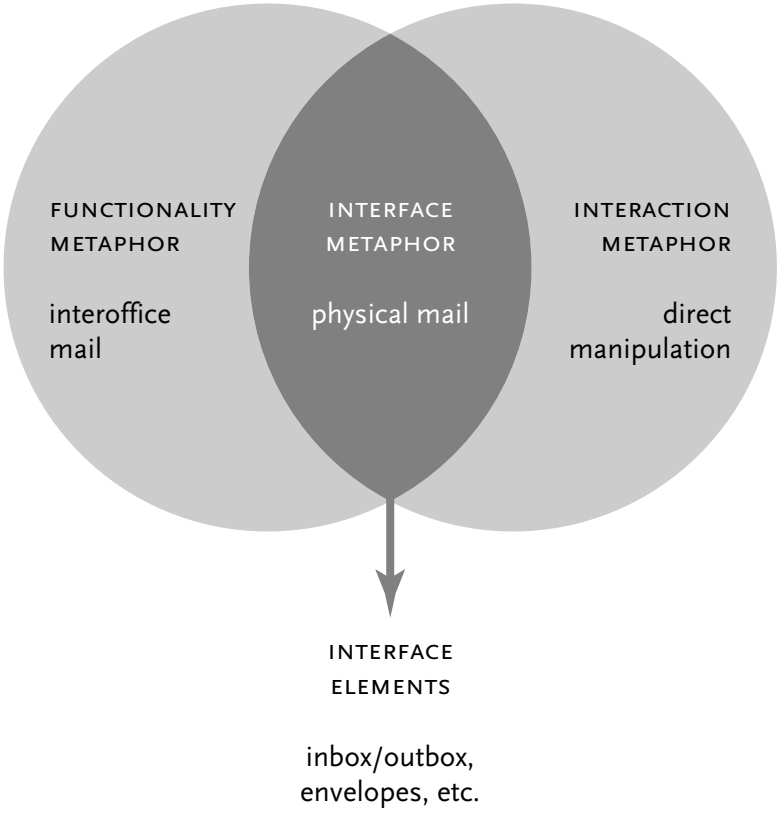
Although the interaction metaphor enables interface metaphors, it doesn't determine them, nor does it determine the system's functionality metaphor. Email systems could just as easily use the functionality metaphor THE EMAIL SYSTEM IS A FAX MACHINE. Enabled by direct manipulation, interface elements might include a send button, sheets of paper rather than envelopes, and a mechanism for "dialing." Questions of appropriateness aside, the example demonstrates how both functionality metaphors and interaction metaphors work together to determine interface metaphors. Figure 1 shows this relationship in diagrammatic form.

Of course the same metaphor might target several domains simultaneously. For example, thinking of email as physical mail affects both the interface and the interaction. In many cases the three domains are closely intertwined, using overlapping metaphors. Still, the distinction between the three different domains can aid designers in targeting problems — they may have a faulty interface, for example, but needn't change their functionality metaphor.

The email example highlights a limitation of relying on metaphor for familiarizing purposes: the user must understand the source domain in order to make the proper correspondences. Given the growth in email use beyond its original academic user base and the changes in office environments over the past few decades, fewer email users have ever used an inbox and outbox or carbon copied a memo. If users don't understand the source domain, they may bring their own metaphors to bear, such as understanding email functionality in terms of the postal service. A change in

the perceived functionality metaphor may, in turn, cause mismatches between interface elements and functionality.

Figure 1: Functionality, interface, and interaction metaphors



A third and final distinction can be made between tacit and explicit metaphors. The contemporary theory of metaphor tells us that almost all complex concepts involve metaphor. When confronted with something new, people usually try to relate it to something they already know. According to this view, metaphor is not an option in design, but an inevitability — the question then becomes whether the user or the designer is aware of the metaphors they use. While users may not always need explicit knowledge of product metaphors, designers have much to gain by surfacing the tacit

metaphors at work in the products they design. Since metaphor plays such an important role in cognition and reasoning, knowing the metaphors our users tacitly bring to our products helps us better meet their needs and anticipate problems. And by thinking through explicit familiarizing or transporting metaphors, we can help users learn about new products and build upon knowledge they have from existing products.

For designers, invention metaphors have great potential. In addition to functioning as a fanciful "what if?" invention metaphors can also question the tacit metaphors designers, clients, and users bring to products. This questioning, or "frame restructuring," can play an important role in the problem-setting part of the design process, and can help to resolve differences between stakeholders who bring different tacit metaphors to the project (Schön 1993).

Much of the criticism of metaphor in design (eg, Cooper 1995) comes from the limitations of familiarizing interface metaphors. In many cases the metaphor fails to explain all of the functionality of the interface, leading designers to abandon or stretch the metaphor and creating confusion in users. In addition, familiarizing metaphors often hide novel functionality provided by new technology or ignore the possibilities that digital devices provide for improving the interfaces of their real-world analogs. This paper will not address the contentious debate over the value of interface metaphors. Rather, I will focus on interaction metaphors — which I believe to be an inevitable feature of human-computer interaction (HCI) — and their usefulness for designers.

Three basic-level human-computer interaction metaphors

According to George Lakoff, almost all conceptual metaphors can be traced back to basic-level image schemas related to the human body in space. So from a few basic building blocks — containers, paths, front/back, hot/cold, etc.— we build complex concepts through metaphor (Lakoff 1993). Without deconstructing HCI metaphors quite so far we can identify three basic-level human-computer interaction metaphors in common use today: direct manipulation, navigation, and human interaction (*The*

Stanford HCI design learning space). Most HCI metaphors can be viewed as a special case of one or a combination of these three basic-level metaphors.

The *direct manipulation metaphor*, already introduced in the email example above, can be expressed as DATA IS A PHYSICAL OBJECT. Special cases of this metaphor may involve specific physical objects, such as files, documents, pieces of paper, books, bulletin boards, and so on. By applying knowledge about the source domain, computer users reason that they can manipulate data the way they manipulate physical objects: by moving, piling, discarding, grouping, splitting, joining, dragging, arranging, locking, hiding, sending, receiving, and sharing it. Data "takes up space," but can be "compressed" to make more of it "fit" within a fixed amount of memory. The computer desktop is the classic example of the direct manipulation metaphor, in which computer users "drag" files "into" folders, but nearly all GUI interfaces involve some form of direct manipulation.

The *navigation metaphor* can be expressed as DATA IS IN SPACE. Special cases of this metaphor may involve specific spaces, such as buildings, rooms, landscapes, oceans, outer space, neighborhoods, bars, shopping malls, and so on. By applying knowledge about the source domain, computer users reason that they can navigate through data the way they navigate through space: by searching, wandering, using landmarks, entering, leaving, exploring, backtracking, following, and foraging. Some pieces of information can be "destinations," and the "path" to them can be straightforward or convoluted, with wrong turns and dead-ends. The world wide web is the classic example of the navigation metaphor, in which users "go to" (or "visit") sites by "following" links and using the "back" and "forward" buttons.

The *human interaction metaphor* can be expressed as COMPUTERS ARE PEOPLE. Special cases of this metaphor may involve specific types of people, such as servants, advisors, children, strangers, and friends, and specific types of interactions, such as interviews, arguments, presentations, small talk, and so on. By applying knowledge about the source domain, computer users reason that they can interact with computers the way they interact with people: by talking, gesturing, asking, answering, ordering,

obeying, scolding, and ignoring. Software can be "smart" or "stupid," "unresponsive" or "helpful." Software agents are the classic example of the human interaction metaphor, in which the computer "asks" and "answers" questions, "helps" or "guides" the user, and "retrieves information." The human interaction metaphor attributes agency to the computer, a concept missing from the other two basic-level metaphors.[3]

Because human interaction takes place within a context of social rules and expectations, the human interaction metaphor implies a social aspect to human-computer interaction. Computers can be either socially intelligent — behaving in ways that match social expectations — or socially awkward. When we say a computer is "stupid," we usually don't mean that it has limited processing power, but rather that it doesn't understand our intentions or behaves inappropriately. Conversely, a "smart" computer seems to anticipate and react appropriately to our needs. Computers can appear socially intelligent without elaborate or complex artificial intelligence systems since they only need to display the appropriate behavior, not understand it (Schmandt 1990).

Although humans can interact with each other in a variety of ways, they predominately use conversation. Therefore the *conversation metaphor*, INTERACTING WITH A COMPUTER IS CONVERSING WITH A PERSON, forms a very large special case of the human interaction metaphor. It implies a narrower range of human interaction — exchange of information coupled with mutual social consideration — but still leaves room for a great deal of variability in the specific form and manner of the conversation. Much of the work done in HCI to date uses the more specific metaphor of conversation rather than the broader metaphor of human interaction.

By breaking human-computer interaction metaphors down into these three categories, I don't mean to suggest that the categories are mutually exclusive. Many interactions combine elements of all three types. File folders in Windows XP, for ex-

[3] Of course other metaphors could also attribute agency to computers, such as COMPUTERS ARE ANIMALS, but the human interaction metaphor provides a more fertile source domain for the purposes of this paper.

ample, rely on direct manipulation of files. But users can also retrace their steps and bookmark favorite "locations," adding elements of navigation to the interface.

The three basic-level interaction metaphors describe different relationships users can have with computers. Since these relationships enable the interface metaphors, designers can benefit from an awareness of which interaction metaphors users are likely to employ and some of the implications of each. The remainder of this paper focuses on only one of the three — the human interaction metaphor — and explores some of its applications.

Benefits of the conversation metaphor

As suggested in the introduction to this paper, idealized conceptions of computer systems of the future often involve people conversing in a natural, social manner with computers. The computer not only understands their speech, but can also interpret nuances in meaning far beyond the capabilities of any current speech recognition system and respond in a socially acceptable manner. While impressive progress has been made in research labs on creating systems that converse the way people converse (see Cassell et al. 2000 for an overview), the vision of generalized, intuitive, natural-language interaction with computers remains a distant dream. Think of the difference between the effortless interaction science fiction characters have with HAL or the *Star Trek* computer and the frustrating experiences you've had with the new generation of "conversational" automated customer-service phone systems.

So if human-computer conversation is technologically impractical and current implementations are more frustrating than GUI systems, why would we want to think of human-computer interaction in terms of conversation? To see the benefits, we first need to distinguish two different approaches to human-computer conversation. The first I call the *literal approach*, which attempts to simulate human conversational behavior in computers by using artificial intelligence models based on human social interaction. Projects in this tradition try to create computers that behave like people because they observe, think, and reason like people. Both the science fiction comput-

ers and the real-world phone systems mentioned above follow this approach. The goal of this approach is to create conversations with computers that are indistinguishable from conversations with people.

The other option is a *metaphoric approach*, which attempts to create interactions that parallel human interaction without being literal conversations. Rather than simulating conversation, the metaphoric approach asks what functions conversation performs in human interaction and tries to reproduce those functions in the computer. Feedback, for example, is an important function in human conversation. When engaged in conversation we give numerous nonverbal cues to our conversational partners to let them know they have been understood. Computers can reproduce the function of feedback without using the same mechanisms that humans use: body language, gaze, etc. It's important to note that under the metaphorical approach, conversational interactions do not necessarily have to involve speech at all. More central to the metaphor is the conception of the computer as a person.

To take a familiar example, we could think of programming a VCR as a conversation, albeit an extremely simple one. We issue a set of instructions to the VCR — record channel 3 from 8:30 to 10:00 — and trust in the VCR's agency to do so. The "conversation" doesn't involve anything close to natural language, but it does convey information to a metaphoric "agent." Of course this same interaction could be viewed in any number of alternate ways, but the choice to use the human interaction metaphor highlights certain aspects of the interaction that might otherwise go unnoticed. Whether or not the VCR is a good conversational partner becomes a possible design consideration, and we can use knowledge about human conversations to arrive at an answer. Does the VCR acknowledge the request, ask for clarification if it hasn't understood correctly, convey its intended course of action, and report back on the results? Is it polite and socially appropriate in its tone? What type of conversation is most appropriate to the interaction, and how does it begin and end?

The literal and metaphoric approaches represent two poles on a continuum, with many points in between. By extending our understanding of conversational in-

teractions to include the more metaphoric range of that continuum, we can look beyond natural language interfaces to see human interaction mechanisms already at work in a broad range of interactive products, and apply the metaphor to the new products we create. Because others have already explored the literal approach to conversational computing (eg, Cassell et al. 2000; McTear 2002; Thórisson 1995), in this paper I focus on the possibilities of a more metaphoric approach to using human interaction as a model for human-computer interaction. A number of recent trends make the perspective afforded by the human interaction metaphor in general — and the conversation metaphor in particular — compelling for interaction designers today.

First, computers are quickly moving away from the desk and into our everyday lives. The trend, often called pervasive or ubiquitous computing, has led to a proliferation of devices that are either mobile or embedded in environments. Interaction with computers used to be limited to a desk environment, either at work or at home. Whether for work or entertainment, these interactions were typically solitary and focused; computer users would devote their full attention to the computer for some period of time in order to complete some task, but when getting up from their desk would leave the computer and the interaction behind. The computer was an activity segregated from the rest of life. While these types of interactions still occur, computers are beginning to integrate into all aspects of our everyday lives, and we often don't think of these devices as computers. Interactions with pervasive computers are often more "lightweight," requiring less than our full attention, because pervasive computers focus on one activity rather than the multifunction approach of desktop computers (Norman 1998). In addition, the goals of computing have changed. People use computers less for work and more for entertainment and communication. Since pervasive computing is still in its infancy, we can expect the trend to continue and even accelerate in the future.

Cell phones provide a good example. People carry cell phones with them almost everywhere, so they become more than a functional piece of hardware. Nokia recognized this shift early on, designing cell phones to be fashionable as well as func-

tional. Because cell phones are used in social settings, they communicate the tastes, aspirations, and standing of their owners, entering into the complex set of mechanisms by which people represent themselves to others (Miner, Chan, and Campbell 2001). But if ugly phones can embarrass their owners, why can't socially unintelligent phones? Applying the human interaction metaphor, current phones behave like socially inept people (children, perhaps, or even infants): they interrupt in the middle of conversations and performances with no regard for their social environment. As computers move from our offices and dens into our social spheres, they must learn to behave accordingly.

In addition, since the graphical user interface is often impractical on pervasive computing devices without keyboards and mice, a new set of interface conventions must be created. Many designers and researchers addressing these interface and interaction challenges are turning to human-human interaction as a framework for enabling new types of interactions (eg, Bellotti et al. 2002).

Furthermore, some researchers believe that people spontaneously develop limited social relationships with computers. Through a series of experiments, Reeves and Nass demonstrated that people extend certain social considerations to computers and other media based on minimal cues. In the experiments, participants knew they were interacting with computers, but they behaved in some ways as though they were interacting with people. For example, subjects were asked to use a computer and then give an evaluation of the computer's performance. Half of the subjects gave the evaluation on the same computer that they were evaluating, and half on a different computer. The first group of subjects gave the computers consistently higher marks, in effect showing politeness to the computer by not criticizing it directly. Reeves and Nass explain their results with the following hypothesis:

> Computers, in the way that they communicate, instruct, and take turns interacting, are close enough to human that they encourage social responses. The encouragement necessary for such a reaction need not be much. As long as there are some behaviors that suggest a social presence, people will respond accordingly. When it comes to being social, people are built to make the conservative error: When in doubt, treat it as human. Consequently, any medium that is close enough will get human treatment,

> even though people know it's foolish and even though they likely will deny it afterward. (Reeves and Nass 1996)

These experiments suggest that the metaphor COMPUTERS ARE PEOPLE has a basis in observable behavior.[4] If we even unconsciously think of computers as people and interact socially with them, this behavior suggests an opportunity for the human interaction metaphor to improve the quality of those interactions.

Since many interactions with computers can be viewed metaphorically as human interaction, and since studies show that people do in fact subconsciously treat computers as humans, designers can only benefit by being aware of the human interaction metaphor's mechanisms, even if the metaphor operates unconsciously for the user. The questions for interaction design then become: how can we make human-computer interaction more like human interaction, and when would we want to? By studying human interactions, designers can learn more about the expectations that users bring to certain types of human-computer interactions, get ideas about how to use human interaction as an appropriate familiarizing or transporting metaphor, and push our invention metaphors further. To aid our understanding, we can draw upon a large body of social science knowledge about the source domain, human interaction. In particular, many social scientists study conversation, and HCI researchers and designers use these theories to inform their projects.

Discourse analysis

Social scientists take different approaches to the study of conversation. While the meanings of individual words or clauses might interest linguists, many social scientists look to larger linguistic units, the social contexts of utterances, and the interactions between people engaged in conversation — a broad area of inquiry often called "discourse analysis" (Stubbs 1983). Discourse analysis is a large and complex field, so this

[4] Other studies indicate that the circumstances in which people extend these social considerations and the degree to which they extend them is considerably more complicated than suggested by the Reeves and Nass study (eg, Shechtman and Horowitz 2003; Kiesler and Sproull 1997).

paper focuses on the major developments that have influenced HCI designers and researchers.

George Herbert Mead was one of the first to study conversation in social context, in his examinations of the meanings of gestures (Mead and Morris 1934), but it was Erving Goffman who laid a substantial amount of the groundwork for discourse analysis. Goffman — in a departure from the academic mainstream of his day — chose to look not at formal, abstract language systems, but at talk: informal, situated, everyday conversation. Where previously linguists saw only disorder, Goffman found patterns and rules in the minutia of conversation that reveal much about social interaction (Slembrouck 2003). Goffman studied what he calls the "interaction order," the set of rules and behaviors surrounding face-to-face social encounters. A social interaction, for Goffman, encompasses much more than the exchange of verbal messages. Shared activity, body language, physical presence, group dynamics, social context, an individual's sense of self, expectations, and cultural conventions all play a role in the interaction order (Goffman 1983).

H. Paul Grice, taking a different approach, developed the "cooperative principle," whereby speakers agree to cooperate in the development of a conversation. Speakers do this by observing four maxims: quality, quantity, manner, and relation. By following these maxims, utterances should be relevant, brief, unambiguous, and contain useful information (Grice 1975). Building on Grice's cooperative principle as well as Goffman's concept of "face," politeness theory, typified by the work of Penelope Brown and Stephen Levinson, describes how people balance the need to be seen as a social person with their desire for unimpeded action (Slembrouck 2003).

Herbert H. Clark, along with many collaborators, explored the mechanisms whereby partners in a conversation establish "common ground:" shared knowledge or understanding that makes communication possible. Like Grice, Clark views conversation as a collaborative activity requiring evidence that each contribution is understood. This presentation of evidence allows for the correction of misunderstandings and the building of common ground (Clark and Schaefer 1989).

Harvey Sacks, Emanuel Schegloff, and Gail Jefferson created the field of conversation analysis in the 1970s, building on the work of both Goffman and Harold Garfinkel. Conversation analysis focuses on the same subject matter as Goffman, ordinary conversational interaction, but with a more empirical method, examining details of conversation such as turn taking, question-and-answer pairs, sequencing, and repair. Building on these simple observed frameworks, conversation analysts explain how context and expectations are continuously created and changed within ordinary conversation (Goodwin and Heritage 1990).

Conversation in HCI: the literal approach

Building on the social science literature outlined above, a number of researchers and designers have worked to explicitly make their projects more conversational. Most use a literal approach, attempting to simulate human behavior using artificial intelligence. Recent work in this area involves quite sophisticated conversational models. Digressing briefly from my main focus on conversation metaphors, a quick overview of the literal approach shows the remarkable progress being made in the field. Designers of metaphoric conversational products may want to borrow bits and pieces of this technology as it becomes available. In addition, looking at how social science concepts are translated literally into conversation can give us ideas about how they might be translated metaphorically.

Within the literal approach, computer agents make up the majority of projects, probably due to the inherently conversational nature of the very idea of agents. By definition, a computer agent possesses agency, so the basic-level metaphor COMPUTERS ARE PEOPLE applies. Special cases are used as functionality metaphors for different agents, involving specific types of people such as assistants, servants, advisors, and so on. The interaction metaphor is almost always conversational, and the interface includes written or audio "speech," usually with a visual or even physical representation of the agent.

Brenda Laurel identifies three key characteristics of agents. First is agency, they must be "empowered to act on behalf of another." Second is responsiveness, including learning. Third is competence, including both knowledge about the domain and about the user (Laurel 1990). I would add another element of competence to her list: social intelligence. Anyone who has used Microsoft Word knows the frustrations of dealing with a socially incompetent computer agent. While Clippy (arguably) meets Laurel's three key characteristics, it lacks the basic social skills that would let a human know when he's not wanted.

To avoid incompetent agents, many software agent designers and researchers turn to discourse analysis, both to give their agents more social intelligence and to make the interaction more like human conversation. A brief review of recent literature on conversational agents reveals not only the complexity of today's artificial intelligence computational models, but also the role of discourse analysis in the endeavor to make conversational agents behave more like people. Not surprisingly, conversation analysis plays a leading role in such efforts, undoubtedly due to its detailed, rule-based approach (eg, Taylor, Néel, and Bouwhuis 2000). Many of today's agents engage in "multimodal" conversation: they can understand speech, intonation, gestures and facial expressions, and they respond with speech, gesture and facial expressions of their own.

Kristinn Thórisson's software agent "Gandalf," for example, uses the Ymir architecture organized around Sacks, Schegloff, and Jefferson's turn-taking model. Gandalf can read a human's speech, vocal intonation, facial expression, and gesture, and then interpret those inputs and make decisions based explicitly on social science models, including Clark & Brennan's grounding model. Gandalf, according to his creator, "has proven to be capable of fluid turn-taking and dynamic dialogue sequencing, and integrates topic and dialogue knowledge seamlessly" (Thórisson 1999).

Conversation in HCI: the metaphoric approach

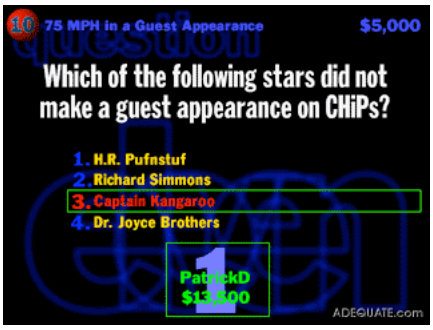
One of the first conversation programs was ELIZA, created by Joseph Weizenbaum in the mid-1960s. ELIZA simulated a Rogerian psychotherapist — users could type statements or questions, and ELIZA would respond, using a system of keyword recognition, templates, and rules. Even though the system had no real knowledge about anything other than a list of keywords, it nonetheless fooled many people into thinking that they were conversing with a person (Weizenbaum 1966).

So how was a rudimentary (by today's artificial intelligence standards) software program able to carry on such a convincing conversation? First, Weizenbaum leveraged linguistic patterns common to the English language. From these basic patterns, ELIZA could understand certain aspects of the function of statements without understanding their meaning. Second, and more important to our purposes, ELIZA benefited from social and cultural expectations that users brought to the conversation. People expect psychotherapists to ask strange questions and give vague responses, assuming them to have some sort of therapeutic purpose. When ELIZA repeats a statement in the form of a question, or simply says "tell me more about that," it is taken as a socially appropriate response within the context of a psychotherapy session (Weizenbaum 1966). In addition, the very structure of a therapy session accommodates the limitations of the system: sessions are limited to two people, therapist and patient alternate utterances in a regular pattern, and the patient drives the conversation. If the user doesn't type anything, ELIZA will remain silent indefinitely. In most conversations this silence would be awkward, if not rude, but within the context of psychotherapy it seems quite normal.

Taking a similar approach, the computer game *You Don't Know Jack* (YDKJ) engages players in a conversational interaction. In the game, up to three players answer trivia questions in a television game show format. A sarcastic and somewhat rude character, with whom the bulk of the interaction takes place, hosts the game, although there are also a few other minor characters. Players use only the keyboard, but the game speaks to them using sound, speech, text, and animation. Rather than rely-

ing on elaborate artificial intelligence systems, however, YDKJ takes its inspiration from television and movies, creating a suspension of disbelief. When we watch a science fiction movie, we know that the aliens aren't real, but we accept the world constructed by the movie and play along. Similarly, YDKJ players know that they're interacting with a computer, but the game allows them to forget the computer and behave as though they were interacting with a real person (Gottlieb 1997).

Figure 2: Screenshot from *You Don't Know Jack*



You Don't Know Jack accomplishes this suspension of disbelief using methods similar to those described in Brenda Laurel's groundbreaking book *Computers as Theatre*. Harry Gottlieb, one of the creators of YDKJ, describes the principles behind creating a successful conversational interface: maintain pacing, create the illusion of awareness, and maintain the illusion of awareness (Gottlieb 1997). Within these three areas, much of his advice could be taken from any set of human-centered HCI design guidelines. Focusing the user's attention on the task at hand, limiting the number of choices, remembering the user's past choices — all are commonly-recommended approaches to designing any type of interface. But Gottlieb also adds advice on what Laurel would call maintaining the mimetic context, or keeping the user focused on the content and characters rather than on the computer (Laurel 1993).

For Gottlieb, one of the principal methods for doing this is to make the characters behave the way people would behave in similar situations. Timing, for example,

is much different in *You Don't Know Jack* than in most software applications. Your word processor doesn't care if you leave your desk to do something else. It will wait patiently for hours, days, or months until you return to find it just as you left it. The host of YDKJ, on the other hand, gets impatient easily. If you don't respond quickly enough, the host will remind you, possibly insult you, and eventually move on without you (Gottlieb 1997). Like a real person, the game requires attention and has its own needs and motives. Compared with the passive assistant model of the *Star Trek* computer (or the "butler"-type software agents in development today), the host of YDKJ requires players to react, adapting to his personality. Although they have less abrasive personalities, many phone systems work in a similar manner — asking the questions and driving the interaction — placing the caller in the reactive role.

Although Gottlieb doesn't mention any direct influence from the social sciences in his paper, he relies, like Joseph Weizenbaum, on an intuitive understanding of conversation and social interaction. Timing is certainly an issue discussed by conversation analysis, especially as it affects how people negotiate turn-taking (Sacks, Schegloff, and Jefferson 1974). The focus on conversational interfaces staying in character calls to mind Goffman's early work in self presentation as well as his later work with the concept of face (Goffman 1959, 1981). And Gottlieb's mechanism for constraining responses could easily be explained by the conversation analysis framework of "next positioning" (Goodwin and Heritage 1990).

You Don't Know Jack falls somewhere between the two approaches discussed earlier. Although in some sense it simulates human conversation, it does so without relying on complex artificial intelligence models, and without worrying about duplicating every aspect of social interaction. By shifting the emphasis from modeling of human cognition and decision-making to suspension of disbelief, YDKJ shows us that artificial intelligence is not a prerequisite for successful conversational interaction.

Both Weizenbaum and Gottlieb chose to use conversation as a model for their human-computer interactions. This choice led them to novel design decisions they might otherwise have overlooked, such as the use of timing in *You Don't Know*

Jack or the pattern recognition in ELIZA. Both projects created convincing and engaging conversational interactions. But neither used a structured social science framework for understanding conversation. Several HCI scholars explore this opportunity, arguing for the advantages of using social science to understand human-computer interaction. This more theoretical approach allows us to see conversational mechanisms at work in a surprising number of computer interactions.

Susan Brennan, in a 1990 article, challenges the distinction between conversational and direct manipulation interfaces by applying the conversation metaphor to graphical user interfaces. The metaphor allows us to see these interfaces in a different light, noticing conversational elements rather than simply direct manipulation. Brennan makes three main points:

> First, direct manipulation interfaces succeed *because* they share important features with real conversations. Second, when so-called "conversational" interfaces fail, it is because they lack these pragmatic features — that is, words alone do not a conversation make. Third, real conversations actually fulfill many of the criteria for direct manipulation. (Brennan 1990)

Taking the canonical direct manipulation interface, the desktop, as an example, Brennan shows how its features are fundamentally conversational. Selecting objects is the equivalent of pointing, users then rely on commands to tell the computer to perform some action (Brennan 1990). Commands grant agency to the computer — the user issues an instruction that the computer must "understand" and execute — and so could be viewed as a conversational interaction. When issuing a command, the user does not directly cause action through direct manipulation. For example, compare dragging a file to the trashcan (direct manipulation) with choosing the "empty trash" command from a menu (conversation). In addition, Brennan states that the virtual workplace could be seen as a shared mental model, and direct manipulation a mechanism for shared activity. The distinction between direct manipulation and conversational interfaces, according to Brennan, is a false dichotomy (Brennan 1990).

Using the terms of this paper, we can say that the desktop is an interface enabled by both the direct manipulation and conversation interaction metaphors. Like-

wise, the desktop's interface metaphors combine both approaches: drag and drop from direct manipulation, and commands from conversation. Given the conversational nature of the command interaction, the specific choice of menus as interface elements is arbitrary — the same commands could be given via a command-line or a speech interface.

Brennan goes on to note several social science models of conversation useful for designers. Grice's cooperativeness principle, for example, could be used to design better error messages, focused on more relevant information (Brennan 1990). Next, Brennan focuses on specific areas where conversation might be a better approach than direct manipulation, such as searching large databases. She makes the point that files on a computer, since you can't see them all at once, function more like a database than a desktop, and quickly become unwieldy under a pure direct manipulation approach (Brennan 1990). Brennan concludes her article by giving designers a list of five design strategies distilled from social science knowledge of human conversation:

> 1) don't continue until an understanding that is sufficient for current purposes is reached; 2) assume that errors will happen and provide ways to negotiate them; 3) articulate the answer or response in a way that preserves the adjacency with (and apparent relevance to) the question or command; 4) represent the interface in a way that invisibly constrains the user to act in ways the system understands and to stick to the application domain; and 5) integrate typed input with pointing and other input output channels. Applying these strategies should make interaction with a computer more *conversational*, whether it's via a desktop or a natural language interface. (Brennan 1990)

Although Brennan writes about a world of desktop computing, her strategies seem just as relevant to pervasive computing devices, even if the mechanisms for implementing them might differ.

A dozen years after Brennan's article, a group of researchers at Xerox PARC revisited the conversation metaphor from the perspective of ubiquitous computing and tangible user interfaces (TUIs). In their paper, the researchers note that the conventional interaction genres of the graphical user interface don't translate well to new devices with means of input other than keyboard, mouse and stylus. They propose a

solution for designers confronted with creating new interactions: look to human-human interaction, and the lessons already learned by social scientists. Highlighting the communicative rather than the cognitive aspects of interaction, they propose a model of five questions to consider in designing systems in which people and computers solve problems together. For each question they summarize the GUI approach, explain how people address the question in human-human interaction, and give examples of projects which answer the question in new ways (Bellotti et al. 2002). These five questions provide a useful framework for exploring some of the applications of discourse analysis to conversational interactions.

The first question, *address*, asks "How do I address one (or more) of many possible devices?" When surrounded by a number of devices that can read your gestures, how do you specify the intended recipient of your gesture or indicate that none of the devices should respond (Bellotti et al. 2002)? Goffman addressed similar issues in his discussion of "participation status," describing three types of listeners: the primary addressee, ratified participants in the conversation who are not the primary addressee but are entitled to hear the utterance, and overhearers who are not ratified members of the group but hear the utterance by accident or eavesdropping (Goffman 1981). For Goffman, physical orientation in space ratifies groups, and the direction of the speaker's gaze indicates primary addressees (Goffman 1963). Designers use both of these approaches with pervasive computing devices. Conversational Desktop, for example, uses a system of microphones to tell whether users are looking at it when they speak (Schmandt 1990). And technologies like Bluetooth are based on physical proximity.

The second question, *attention*, asks "How do I know the system is ready and attending to my actions?" (Bellotti et al. 2002). In human interaction terms, we would ask, "How do I know if another person is aware of my presence and ready to engage in conversation?" Goffman addresses this very issue, giving as an answer a complex set of cultural expectations around posture, gaze, and facial expression (Goffman 1963). Although most computers today don't have posture or facial expression, they do signal

attentiveness. Command-line interfaces, GUI text input fields, and word processing programs all use a flashing cursor to indicate that the computer expects text from the user. And graphical user interfaces often use a watch or an hourglass to signal inattention or a temporary inability to engage in conversation. In smaller mobile devices, signals of inattention are more common than explicit signals of attention. Many cell phones, for example, allow users to lock the keys so as not to make accidental calls when the phone is placed in a pocket. Typically these phones have a "key lock" icon to indicate inattention when this feature is active.

The third question, *action*, asks "How do I effect a meaningful action, control its extent and possibly specify a target or targets for my action?" Bellotti et al. address this question using a direct manipulation framework, asking how best to associate gestures or physical objects with desired actions. In choosing this framework, they temporarily move away from the conversation metaphor, missing some of its most interesting implications.

Instead of asking how people can associate virtual actions with physical movements, let's grant agency to the computer and ask how people accomplish tasks conversationally. Speech act theory, an early contribution to discourse analysis, examines the social actions people perform when speaking. John L. Austin describes three types of "speech acts:" locution, illocution, and perlocution, which may all apply to the same utterance. If I were to say, "I could use a drink," that utterance would be a locutionary act in that it is a meaningful statement, describing my current condition. But it would also be an illocutionary act in that it implies an intention to get myself a drink, or a request for someone else to get me a drink. Depending on the context, the utterance might also prompt action in someone else — offering me a drink or pointing me to the nearest bar — in which case it would become a perlocutionary act. Illocutionary acts can be either direct ("Turn the radio down") or indirect ("The radio is too loud") (Austin 1975; Searle 1969; Slembrouck 2003).

In conversational human-computer interaction, both the human and the computer need to perform illocutionary acts — promises, communications of intent,

requests, commands, and so on — in order for perlocutionary action to occur. Designers usually provide users with commands as their primary method for illocutionary action, largely because computers have difficulty understanding other types of illocutionary acts such as promises or threats. But the metaphor allows us to imagine other possibilities. Bluetooth technology currently allows my computer to pause my music when I get a call on my cell phone. The computer interprets the call as an indirect request for quiet so I can hear my conversation.

The question of action implies a further question: "How do I know what the device can do?" (Bellotti et al. 2002). Goffman answers this question for human interaction with his concept of "front." A front for Goffman is the set of signals — both appearance and actions — that others use to determine our social status, mood, intentions, and so on. As humans in a social environment, we constantly control these signals when in the presence of others in order to make our front match the kind of person we are or aspire to be (Goffman 1959). While similar to the concept of affordances in direct manipulation interactions (see Norman 1990), face implies a consistency of character across multiple actions. Consistency in behavior helps conversational interaction users build models of the type of agent with whom they are interacting.

Bellotti et al.'s fourth question, *alignment*, asks "How do I know the system is doing (has done) the right thing?" (Bellotti et al. 2002). Feedback has always been an integral part of direct manipulation, allowing users to see the consequences of their actions (Frohlich 1997). But under the conversational metaphor, feedback means that the system shows its *understanding*, even if no immediately visible action takes place. In conversation analysis, "backchannel" communication — a system of "uh-huhs," "yeahs," and "mm-hms" — tell the speaker that they're being heard and understood (Sacks, Schegloff, and Jefferson 1974). In addition, "participation displays" let speakers know whether listeners agree or disagree with their position, leading to a possible modification of their utterances (Goodwin and Heritage 1990).

Brennan, as one of Clark's frequent collaborators, addresses this question from the perspective of common ground. Focusing specifically on computers as agents, Brennan divides the computer's process of conveying its understanding to the user into six states:

- *Receiving*. The system is receiving input (but the input is not yet recognized as well formed).
- *Recognizing*. The system recognizes the input as well formed (but has not yet mapped it onto any plausible interpretation).
- *Interpreting*. The system has reached an interpretation (but has not mapped the utterance onto an application command).
- *Intending*. The system has mapped the user's input onto a command in its application domain (but has not yet acted).
- *Acting*. The system attempts to carry out the command (an attempt that may or may not turn out to be successful).
- *Reporting*. The system has attempted to carry out the command, and reports to the user any relevant evidence from the application domain. (Brennan 1998)

The computer can give feedback on any of these states; it needn't wait until the end. When dragging files from one folder to another in the OSX operating system on a Macintosh, for example, the destination folder highlights as the mouse rolls over it, before the user releases the mouse button to copy the files. This highlight shows the system's interpretation and intent before action occurs.

Finally, the fifth question, *accident*, asks "How do I avoid mistakes?" In addition to prevention, this question implies a need for both acknowledgement and repair of mistakes (Bellotti et al. 2002). According to conversation analysts, people misunderstand each other in conversation all the time. Through both backchannel communication and what they say during the next speaking turn, listeners constantly provide feedback on what they understand the speaker to have said. If this understanding is incorrect, the speaker can repair it during the next turn (Schegloff 1992). The process is so common that it goes unnoticed much of the time. As discussed above, conversational products, just like people, should broadcast not only their hearing but also their understanding. If the computer has misunderstood, users should be able to repair the understanding before any irreversible action occurs. An undo function provides the

most obvious example of repair after next turn in GUI interactions, allowing users to repair erroneous actions by redoing them entirely. In some cases, however, a clarification might work better than redoing the entire action. For example, on numerous occasions I have printed a document, closed the file, and then realized I sent the job to the wrong printer. Rather than reopening the file and printing all over again, I should be able to simply take the erroneous job and send it to a different printer, repairing rather than repeating my earlier request.

Conversational interaction: an example

Brennan and Bellotti et al. both propose using social science models of conversation as a framework for human-computer interaction. How would this framework change the design process and the resulting product? What kinds of design questions does the conversation metaphor raise, and how does it help make design decisions? To answer these questions, let's look at an example: MetaMonitor, a patient monitoring and alarm system concept for intensive care units (Fineman 2004). Currently, intensive care units (ICUs) contain scores of medical devices, each with their own unique monitoring functions and alarms. In envisioning a more coherent monitoring and alarm system, MetaMonitor considers human interaction as a model.

Issues of address and attention are critical in the ICU environment. Currently, when an alarm sounds, a loud tone emerges from a device, addressing whoever happens to be close enough to listen, regardless of whether they are already aware of the problem. Applying the conversation metaphor, we could view the alarm as a statement and the people in the intensive care unit as Goffman's three types of listeners: the primary addressee, ratified participants, and overhearers (see above, page 27). This mapping raises several design questions. What participation status does each person in the ICU have? Should the different types of listeners receive different information? How do people know if they are the primary addressee for a particular alarm? Is there some information that shouldn't be overheard for legal or privacy reasons?

MetaMonitor addresses these questions by providing a different notification for each type of listener. Because every patient has a nurse assigned specifically to him or her, this nurse is the primary addressee, receiving a notification of that patient's alarm condition via a personal wearable device. For less serious alarms, other nearby nurses play the role of ratified participants, receiving a less intrusive notification. And a light outside each room provides less specific information for any "eavesdroppers" who might be interested in the alarm. In the case of life-threatening emergencies, however, everyone nearby becomes a primary addressee.

Returning to the conversation metaphor, we could view patient monitoring alarms as a type of speech act (see above, page 28). Although they have a locutionary component — they convey information about an alarm condition — the perlocutionary function is more important. Alarms prompt action in nurses, specifically checking on the patient and performing any necessary intervention to stabilize the patient's condition. In human conversation, if I said "it's hot in here," and you got up and opened the window, it would make no sense for me to keep repeating "it's hot in here" until either a) the temperature dropped to a more acceptable level or b) you explicitly told me to stop. Yet this is precisely what current alarm systems do, by continuing to sound until the nurse goes to the machine and turns it off. The metaphor raises several design questions. How does an alarm convey its illocutionary force, prompting perlocutionary action in nurses? How does the nurse know what to do? How does the system know when a nurse has responded to an alarm?

MetaMonitor addresses the last question by tracking the location of nurses in the unit. The system silences the alarm if a nurse enters the patient's room after an alarm sounds, without waiting for an explicit command to be silent. MetaMonitor expects a certain perlocutionary action and ceases its speech act when that action occurs. This allows nurses to focus on responding to the patient rather than responding to the alarm.

Finally, a major problem with patient monitoring systems is the number of false and clinically irrelevant alarms. As illustrated in the classic fable of the boy who

cried wolf (Aesop), people who issue inaccurate statements quickly lose credibility. Although MetaMonitor doesn't change the underlying detection algorithms, and so can't reduce the number of false alarms, the conversation metaphor suggests a design question that might otherwise go unasked: how does the system maintain credibility when it is wrong so often?

For Goffman, credibility is a face issue. One solution to the problem, then, would be to work on the system's face to make it more credible. For example, research suggests that people place more trust in computers that use color and clip art (Kim and Moon 1997, cited in Bickmore 2003), or that profess to be experts in a certain domain (Reeves and Nass 1996). Nurses, however, don't want a system that seems more credible than it actually is, they want an accurate picture of its limitations. When people want to indicate that their utterances might not be accurate, they often employ a strategy linguists call *hedging* to distance themselves from their statements. Hedging involves using phrases such as "sort of" or "I suppose" to call into question the truth-value of the proposition (Markkanen and Schröder 1997). In this case, the solution used by people is probably not appropriate for human-computer interaction, but the question would not have been raised without the conversation metaphor.

MetaMonitor addresses the credibility issue in two ways. First, it provides context for the alarm in the form of a trend graph. The graph gives nurses the ability to judge for themselves the seriousness of the patient's condition. Second, MetaMonitor separates the most frequent false alarms — those generated by the pulse oximeter — into their own category. When nurses hear an alarm in this category, they understand that it is more likely to be a false alarm. By categorizing alarms, lower credibility in one category can lead to higher credibility in the others.

Although more could be said about MetaMonitor's use of conversational interactions, the examples given should show how the metaphor leads to both new questions and answers in interaction design.

Relationships

Both Brennan and the PARC researchers provide frameworks for understanding human-computer interaction in terms of conversation. But both frameworks leave out an important aspect of social interaction: how it changes over time. In humans, interactions over time become a relationship — conversing with a friend is not like conversing with a stranger. Social scientists study two broad categories of relationships: personal (social or intimate) relationships, and working (task-based) relationships (Gabarro 1987). Although some researchers focus on purely social interaction with agents, many human-computer interactions in the foreseeable future would probably best fit the working relationship model.

John Gabarro, in a 1987 article, outlines three different models for social or intimate relationships, and uses them to build a new model of working relationships. Summarizing the work of many other scholars, Gabarro describes eleven dimensions along which social and intimate relationships develop: openness and self-disclosure, knowledge of each other, predictability of other's reactions and responses, uniqueness of interaction, multimodality of communication, substitutability of communication, capacity for conflict and evaluation, spontaneity of exchange, synchronization and pacing, efficiency of communication, and mutual investment. Progression along these dimensions results from social exchange over time, self-disclosure, exploration, negotiation, and testing, driven by the prospect of greater personal rewards from the deeper relationship. Relationships tend to move through stages, from safe, stereotypical interactions to deeper, more committed, and more personal interactions (Gabarro 1987).

In addition to these general relationship processes, working relationships also have unique features due to their task-based orientation. Since achieving the task is the final reward, the "pragmatic imperative" often makes people overlook social flaws in colleagues who are good task performers. Similarly, personal disclosure is less important than task-related openness in building trust. Within organizations, roles and

power relationships also tend to be more explicitly defined, which can have a dampening effect on personal disclosure (Gabarro 1987).

Finally, Gabarro presents a four-stage model for working relationships: orientation, exploration, testing, and stabilization. Working relationships progress "from role-specified surface encounters to a greater degree of mutual exchange and task-related efficacy." Successful working relationships, for Gabarro, depend on two factors. The first is time. All relationships are built on a series of interactions over time, and working relationships take time to develop. Accelerating the process requires devotion of emotional energy into the relationship. The second factor is the development of mutual expectations around the task and outcome, and the logistics of working jointly and individually on the task (Gabarro 1987).

Although relationships are an important aspect of social interaction, not as much work has been done in applying relationship models to human-computer interaction. One exception is the Gesture and Narrative Language group at MIT Media Lab, which advocates "relational agents" that build trust in users over time. Based on politeness theory and Goffman's concept of "face threats," Timothy Bickmore and Justine Cassell modified REA, a virtual real estate agent, to evaluate the level of trust she has established with the user. Working on the assumption that relationships involve a progression of deepening familiarity, Bickmore and Cassell suggest that conversational topics that are at a deeper level of familiarity than the current relationship could be perceived as face threats. In order to close a deal, REA has to ask sensitive questions about how much money the homebuyer is willing to spend. If she thinks a question might be threatening, she uses small talk to deepen her relationship with users before asking it (Bickmore and Cassell 2001).

REA uses relational strategies to manage individual conversations, but relationships develop over a longer period of time through a series of interactions. To address these issues, Bickmore created Laura, an agent that helps students maintain an exercise regime. Laura remembers and talks about details of previous interactions, and her conversations follow a script of greater familiarity with each interaction. The rela-

tionship is based on the patient-therapist "working alliance," which like exercise involves a significant change in behavior on the part of the patient (Bickmore 2003).

The theoretical underpinnings of both Laura's and REA's behavior are elaborated in Bickmore's Ph.D. thesis. Bickmore believes relational agents can provide some of the social and emotional support that people get from human relationships, especially in applications such as learning and helping people through major behavioral changes. Taking a broad view of relationships — "a unique pattern of interdependent behavior between two entities" — he identifies four categories of relational agents in human-computer interaction: social agents, affective agents, anthropomorphic agents (including embodied conversational agents and sociable robots), and relational agents. Relational agents may or may not use social interaction, affect, or anthropomorphic form, but they are specifically designed to build relationships with users. As examples, Bickmore cites entertainment products such as Tamagotchi, Furby, and Aibo, and of course the much more sophisticated embodied conversational agents REA and Laura (Bickmore 2003).

Bickmore grounds his approach to relational agents in a broad review of social psychology and discourse analysis, discussing such issues as face threats, trust, motivation to meet relational expectations, how relationships change over time, and relationship maintenance. He applies this theory to both the "micro-structure" of individual conversations, as seen in REA, and the "macro-structure" of interactions over time, as seen in Laura (Bickmore 2003).

Taking a literal approach, Bickmore focuses on embodied conversational agents that attempt to simulate human conversation. But what about a more metaphoric approach — how might relationship theory apply to conversational interactions that aren't literally conversations? While some of Bickmore's design implications are specific to the agent approach, many of them could apply to conversational interfaces of all sorts, and have already been implemented on many websites.

One set of implications involves persistence and continuity across interactions (Bickmore 2003). When people in a relationship interact, they don't have to start over

each time; they remember where they left off and continue from there. For most HCI applications, this capability to remember implies an awareness of individual users and the computer's past interactions with each. Amazon.com, for example, employs this strategy by remembering what customers did on previous visits to the site.

For people, remembering previous interactions is an essential part of relationships, but computers could still behave relationally within individual interactions without this memory, as evidenced by REA. For example, they could use an awareness of the level of intimacy or trust they have with the user. This awareness would then determine what actions or conversational topics would be appropriate for that stage of the relationship (Bickmore 2003). Many websites use a simple version of this approach by targeting different content and allowing a greater range of action to visitors that are logged in or registered.

Computers could also employ strategies proven to make themselves seem more likeable or trustworthy, such as using flattery and praise (Reeves and Nass 1996), anthropomorphic form (Lester et al. 1997, cited in Bickmore 2003), or an ability to justify and explain decisions (Miller and Larson 1992, cited in Bickmore 2003). Or computers could use non-task-related talk, or "social dialog," to build rapport with users. Social dialog includes small talk, storytelling, gossip, jokes, and getting acquainted talk (Bickmore 2003). Although social dialog is not a common strategy for interactive products, one could view certain flash animations on websites as a type of small talk. While waiting for long animations, for example, some sites show a faster-loading diversion. And some flash intros — animations that appear when visitors first arrive at certain sites — are only tangentially related to site content. The effectiveness of such social talk is questionable, however, as evidenced by the declining number of flash intros in recent years and backlash movements such as skipintro.com.

Another set of design implications involves how relationships change over time. Relationships grow in trust and intimacy, and in the types of activities partners perform. While some of this growth is the result of interactional negotiation, cultural conventions also play a role, in the form of expected trajectories for relationship

change (Duck 1990, cited in Bickmore 2003). In our culture the expected relationship pattern for lovers, friends, and neighbors, for instance, differ quite radically. Computers without the complex artificial intelligence models necessary to track relationship changes over time could use these expected patterns to make guesses about appropriate behavior for their particular type of relationship. These computers would not need a detailed memory of previous interactions, just a sense of how many interactions a particular user has had. Like social dialog, this strategy is uncommon today, although some websites offer help or guided tours to visitors only the first few times they visit.

One potential problem mentioned by Bickmore involves the competence of the computer. If computers can converse in a natural manner and manage relationships over time, users might have unrealistic expectations of the computer's intelligence, knowledge, or abilities. While many of these expectations might arise from the anthropomorphic form and behavior of his agents, Bickmore's advice applies equally well to less realistic applications of the relationship metaphor. To avoid unrealistic expectations, Bickmore suggests meta-relational communication — having the computer explicitly talk about their limitations — and avoidance of free-form natural language input (Bickmore 2003).

Finally, relationships require maintenance in order to keep going. Some relationship behaviors are routine; the simple process of shared activity and regular interaction keeps relationships strong. But others are strategic, requiring conscious effort such as talking about the relationship with one's partner (Bickmore 2003). Both processes can be seen on amazon.com, which not only gathers relational data from the browsing and shopping behaviors of customers, but also asks them to rate items and manage a list of preferences. These preferences indicate appropriate topics of conversation within the relationship (whether they want to receive promotional emails, for example).

Working relationships

To many people, designers and non-designers alike, the relationship approach outlined above appears to have questionable utility. I, for one, don't care to be best friends with my DVD player or engage in idle chit-chat about the weather with my phone. And although it's an interesting challenge, there are complex moral issues with Bickmore's vision of computers providing social and psychological support. We don't want to create a world in which the relational capabilities of computers are used as an excuse to isolate elders, children, or people in institutional care from the much richer companionship of other people.[5] In addition, when people invest social and emotional energy into relationships, strong emotions such as love, jealousy or rage can result from forming, maintaining, disrupting or terminating them (Bowlby 1979, cited in Bickmore 2003). Do we really want computers to provoke these emotions?

I believe Gabarro's distinctions between working relationships and social relationships provide a solution to these dilemmas. Bickmore focuses on the social and emotional aspects of relationships rather than the task element that distinguishes working relationships. Even the working alliance model he borrows from psychology focuses on the trust and empathy between therapist and patient rather than a more concrete shared activity (Bickmore 2003). Working relationships, of course, have important social and emotional aspects, and Bickmore's model may be best for embodied conversational agents that attempt to simulate human interaction. But for metaphorically relational interactions the task-oriented approach has several advantages.

First, motivations differ in working relationships. Bickmore cites the models of social contract, rights and obligations, and fear of negative implications as motivations for partners to keep with relational expectations (Bickmore 2003). While not necessarily contradicting the contract and rights framework, Gabarro suggests that task

[5] I don't mean to suggest that relational agents would inevitably be used for this dystopian scenario, or that Bickmore's approach somehow suggests these uses. In fact, Bickmore addresses the issue and proposes an interesting application of the technology to help the socially challenged learn how to relate better to other people (Bickmore 2003).

accomplishment rather than social acceptance is the primary reward and motivation for working relationships (Gabarro 1987). If I want a relationship with my cell phone, then, it's not because I need a friend or I'm afraid of hurting its feelings, but because the relationship helps me to make calls more efficiently.

Second, the pragmatic imperative shows that social factors are secondary to competence and successful task performance in judging likableness and trust of working partners (Gabarro 1987). So while in an experimental condition users may prefer computers with personalities similar to their own (Reeves and Nass 1996), or place more trust in computers that use color and clip art (Kim & Moon 1997, cited in Bickmore 2003), in an actual working interaction we would expect this preference to be eclipsed by task competence.

Third, roles and expectations are more explicitly defined and structured for most working relationships. Roles often include asymmetries of power, and personal disclosure is expected to be less than task-related disclosure (Gabarro 1987). Because talk tends to be task focused and because of the pragmatic imperative described above, we would expect less of a need for social dialog in human-computer interactions based on a working relationship metaphor than a social relationship metaphor. This frees designers from having to worry about small talk with interactive devices. Furthermore, we can expect roles to remain relatively static and relationships to deviate less wildly from conventional norms. This allows designers to rely on expected relationship trajectories rather than a sophisticated model of relationship status based on an understanding of past interactions.

Shifting the focus from social relationships to working relationships allows designers to concentrate on the aspect of the relationship likely to matter most to users: working towards the users' goals. The design challenges then become how best to share responsibility and agency in achieving those goals, and how to enable an appropriate conversation between the two partners. Users can have productive working relationships with interactive products without investing a great deal of personal or emotional energy.

Relational interaction: an example

To illustrate the questions raised by using working relationships as a model for human-computer interaction, let's look at a common example: the personal digital assistant (PDA). PDAs have a human interaction metaphor built into their names: the assistant metaphor. Because of the name, both users and designers of PDAs might already see an explicit connection between the devices and real-world assistants. But let's extend the metaphor, using social science frameworks to push the idea to its limits. What does it really mean for a PDA to be an assistant? Since assistants are people, the human-interaction metaphor applies, as does the conversation metaphor. Designers could employ any of the strategies discussed earlier in this paper to make the interaction more conversational. But an assistant is also a particular type of person, defined by a specific working relationship. How might this relationship affect the design of a PDA?

First, let's examine how existing PDAs use the assistant metaphor. As a functionality metaphor, PDAs don't live up to the source domain. Real-world administrative assistants perform useful work such as screening calls, organizing files, reading mail, and creating documents and presentations. Real-world personal assistants might pick up dry cleaning, maintain correspondences, shop, and run errands. PDAs perform none of these functions, nor do they do any of the significant work of other types of assistants such as butlers, concierges, sales clerks, babysitters, gas station attendants, or crossing guards. In terms of interaction, PDAs borrow their conventions and metaphors from personal computers. As discussed above, these GUI interactions are partly based on the human interaction metaphor, but there is little about assistants as a specific category of people or as a working relationship that comes through in PDA interactions as opposed to personal computer interactions.

In what sense, then, are PDAs assistants? The first commercially successful PDAs focused on a few simple functions: a calendar, address book, calculator and note pad. Since some assistants help people manage their schedules and their correspondences, one could draw a correlation. However, the functionality, were we to make the

metaphor literal, would be equivalent to an assistant following you around holding a calendar and an address book without actually doing the work of a real assistant. Rather than structuring the functionality of the PDA, the assistant metaphor structures its topics. In conversation, people stick to certain acceptable topics, determined not only by the conversation itself (Grice 1975; Sacks, Schegloff, and Jefferson 1974), but also by the type of relationship the participants have (Gabarro 1987). I wouldn't ask my assistant to baby-sit my children, for example (or if I did it would be seen as an out-of-role personal request rather than part of our working relationship). The assistant metaphor, then, keeps PDAs focused on the topics of scheduling and correspondence without offering assistant functionality or an assistant-like interaction. This is a tenuous correlation, made even weaker by the addition of music, photo and game functions to the basic PDA topic set. As a result, using a PDA doesn't feel much like interacting with an assistant.

To make PDAs more relational, we could first ask how the device maintains continuity across interactions. A sophisticated computer could begin to learn the user's preferences over time, noticing repeating patterns and frequently accessed information. A less sophisticated computer could simply remember all of the things a user has done before and make it easier to repeat them again. Many web browsers already have an auto-fill feature that performs this function.

More interesting is the question of how the relationship changes over time. Unlike desktop computers, most PDAs have only a single user. Consequently, the device can make assumptions about how far along the current relationship is in the expected trajectory. Working relationships progress through four stages: orientation, exploration, testing, and stabilization (Gabarro 1987). During the orientation phase, the PDA could focus on letting the user know its capabilities and limitations as well as the logistics of how to accomplish tasks. Some software programs provide this kind of orientation, either with "tips" at startup or with an agent such as Microsoft Office's Clippy (Clippy, however, doesn't know when to stop orienting and move on to the other stages).

In an exploration stage, the PDA could feel out the boundaries of its relationship. If a user hasn't taken advantage of a particular feature, is it because he or she is disinterested or unaware of the feature? An exploratory question might be appropriate, but only after the device has established a certain level of trust and rapport with the user by demonstrating task competence. On the second day after purchasing a PDA, a question such as "Did you know I can automatically sync with your cell phone?" might seem pushy and annoying; during the second month of use it might be appreciated, especially if the device determines during those two months that the user does in fact have a cell phone.

These examples show only a few of the possible design considerations that could result from the relationship metaphor. Obviously they would still need to be balanced with other considerations as part of a larger design process. Would the orientation and exploration described above slow down or interrupt important tasks the user wants to accomplish with the PDA? Are they technically feasible? How, precisely, would these ideas be implemented in the interface? Although it cannot provide easy answers, the relationship metaphor raises important questions, and can widen the interaction designer's palette of options considerably.

Conclusion

We have seen a number of examples of designers and researchers who use human interaction metaphors for human-computer interaction. Some turn to social science for a structured approach to human social interaction, while others rely on an intuitive understanding of human conversation. So if projects like ELIZA and *You Don't Know Jack* can succeed without explicit reference to social science, why should interaction designers bother to learn about discourse analysis or relationships? After all, we converse and relate with other people every day; we should know how it works.

The question ignores a fundamental premise behind the practice of design. Just because someone knows how to drive a car doesn't mean he or she can design one. A car designer needs a detailed understanding of how cars work in order to make

a car that anyone can drive. Similarly, since interaction designers are in some sense designing conversations and relationships, they would benefit from a much more structured understanding of how conversations and relationships work. Even if human interaction is only being used as an invention metaphor, a broad knowledge of social science gives designers a wider palette of creative opportunities. In addition, by running through the different theories and approaches, the metaphor can be extended in new and interesting ways. Not all products need a sense of timing or should develop working relationships, of course, but thinking about these interactions metaphorically could lead to better design decisions.

Metaphor highlights certain aspects of the target domain while hiding others. As designers we should look at our design problems from as many different angles as possible in order to expose these hidden viewpoints. The human interaction metaphor is only one of many metaphors available, but it has the advantages of tapping into a fundamental human experience as well as providing a rich and structured source of knowledge about the source domain. In applying the human interaction metaphor, the goal is not to create conversational or relational interactions to the exclusion of other types of interactions, but rather to see the conversational and relational *aspects* of every interaction. We can then use our structured knowledge of human interaction to think through the implications of these aspects and arrive at better design decisions.

We are only at the beginning of this process. Pervasive computing demands new interfaces and interactions, and we are only starting to explore the human interaction metaphor's implications. This paper highlights only a few of the possibilities implied by the rich body of social science knowledge. Ideas will need to be tested and refined, building a new vocabulary for human-computer interaction, a new set of conventions for conversational and relational interfaces, and a new generation of conversational and relational products.

It's important to note that none of the projects discussed, not even the most sophisticated agents, reproduce the full range of human behavior. Like ELIZA they take advantage of cultural conventions which limit the knowledge one would expect them

to have, and which structure the form of the conversation. By understanding the social implications of the particular metaphor and frame chosen, as well as the context surrounding use of the product, we can design products that seem naturally conversational and relational within that context without waiting for HAL. Conversely, the human interaction metaphor could help drive technology research by highlighting areas where more sophisticated models are required, driven by real-world product requirements rather than an attempt to simulate all of human behavior.

References

Aesop. Aesop's Fables. Translated by G. F. Townsend, The Internet Classics Archive (classics.mit.edu).

Austin, J. L. 1975. How to do things with words. 2d ed, The William James lectures: 1955. Oxford [Eng.]: Clarendon Press.

Bellotti, Victoria, Maribeth Back, W. Keith Edwards, Rebecca E. Grinter, Austin Henderson, and Cristina Lopes. 2002. Making sense of sensing systems: five questions for designers and researchers. In Proceedings of the SIGCHI conference on human factors in computing systems: ACM Press. 415-422.

Bickmore, Timothy. 2003. Relational agents: effecting change through human-computer relationships. Ph.D. thesis, Massachusetts Institute of Technology.

Bickmore, Timothy, and Justine Cassell. 2001. Relational agents: a model and implementation of building user trust. In Proceedings of the SIGCHI conference on human factors in computing systems: ACM Press. 396-403.

Bowlby, John. 1979. The making & breaking of affectional bonds. London: Tavistock Publications.

Brennan, Susan E. 1990. Conversation as direct manipulation: an iconoclastic view. In The art of human-computer interface design, edited by B. Laurel and S. J. Mountford. Reading, Mass.: Addison-Wesley Pub. Co. 393-404.

———. 1998. The grounding problem in conversations with and through computers. In Social and cognitive approaches to interpersonal communication, edited by S. R. Fussell and R. J. Kreuz. Mahwah, N.J.: Lawrence Erlbaum Associates. 201-225.

Brown, Theodore L. 2003. Making truth: metaphor in science. Urbana: University of Illinois Press.

Cassell, Justine, Timothy Bickmore, Lee Campbell, Hannes Vilhjálmsson, and Hao Yan. 2000. Conversation as a system framework: designing embodied conversational agents. In Embodied conversational agents, edited by J. Cassell. Cambridge, Mass.: MIT Press. 29-63.

Clark, Herbert H., and Edward F. Schaefer. 1989. Contributing to discourse. Cognitive Science 13 (2):259-294.

Cooper, Alan. 1995. The myth of metaphor. Visual Basic programmer's journal, July, 127-128.

Duck, Steve. 1990. Relating to others. Pacific Grove, Calif.: Brooks/Cole Pub. Co.

Fineman, Benjamin. 2004. MetaMonitor: a system for patient monitoring in intensive care units. Masters Thesis Project, School of Design, Carnegie Mellon University, Pittsburgh.

Frohlich, David M. 1997. Direct manipulation and other lessons. In Handbook of human-computer interaction, edited by M. Helander. Amsterdam; New York: North-Holland. 463-488.

Gabarro, John J. 1987. The development of working relationships. In Handbook of organizational behavior, edited by J. W. Lorsch. Englewood Cliffs, NJ: Prentice-Hall. 172-189.

Goffman, Erving. 1959. The presentation of self in everyday life. Garden City, NY: Doubleday.

———. 1963. Behavior in public places: notes on the social organization of gatherings. New York: Free Press of Glencoe.

———. 1981. Replies and responses. In Forms of talk. Philadelphia: University of Pennsylvania Press. 5-77.

———. 1983. The interaction order: American Sociological Association, 1982 presidential address. American Sociological Review 48 (1):1-17.

Goodwin, Charles, and John Heritage. 1990. Conversation analysis. Annual Review of Anthropology 19:283-307.

Gottlieb, Harry. 1997. The Jack principles of the interactive conversation interface: Jellyvision Inc.

Grice, H. Paul. 1975. Logic and conversation. In Syntax and semantics: volume 3: speech acts, edited by P. Cole and J. L. Morgan. New York: Academic Press. 41-58.

Heckel, Paul. 1991. The elements of friendly software design. New ed. San Francisco: SYBEX.

Hutchins, Edwin. 1989. Metaphors for interface design. In The structure of multimodal dialogue, edited by M. M. Taylor, F. Néel and D. G. Bouwhuis. Amsterdam; New York: North-Holland. 11-28.

Kiesler, Sara, and Lee Sproull. 1997. 'Social' human-computer interaction. In Human values and the design of computer technology, edited by B. Friedman. Stanford, Calif.; Cambridge; New York: CSLI Publications; Cambridge University Press. 191-199.

Kim, Jinwoo, and Jae Yun Moon. 1997. Emotional usability of customer interfaces — focusing on cyber banking system interfaces. Paper read at CHI 97 conference on human factors in computing systems, at Atlanta.

Kubrick, Stanley. 1968. 2001: a space odyssey. New York, NY: MGM/United Artists Entertainment.

Lakoff, George. 1993. The contemporary theory of metaphor. In Metaphor and thought, edited by A. Ortony. Cambridge; New York: Cambridge University Press. 202-251.

Lakoff, George, and Mark Johnson. 1980. Metaphors we live by. Chicago: University of Chicago Press.

Laurel, Brenda. 1990. Interface agents: metaphors with character. In The art of human-computer interface design, edited by B. Laurel and S. J. Mountford. Reading, Mass.: Addison-Wesley Pub. Co. 355-365.

———. 1993. Computers as theatre. Reading, Mass.: Addison-Wesley Pub. Co.

Lester, James C., Sharolyn A. Converse, Susan E. Kahler, S. Todd Barlow, Brian A. Stone, and Ravinder S. Bhogal. 1997. The persona effect: affective impact of animated pedagogical agents. In Proceedings of the SIGCHI conference on human factors in computing systems: ACM Press. 359-366.

Markkanen, Raija, and Hartmut Schröder. 1997. Hedging: a challenge for pragmatics and discourse analysis. In Hedging and discourse: approaches to the analysis of a pragmatic phenomenon in academic texts, edited by R. Markkanen and H. Schröder. Berlin; New York: de Gruyter. 3-18.

McTear, Michael F. 2002. Spoken dialogue technology: enabling the conversational user interface. ACM Comput. Surv. 34 (1):90-169.

Mead, George Herbert, and Charles W. Morris. 1934. Mind, self & society from the standpoint of a social behaviorist. Chicago, Ill.: The University of Chicago press.

Miller, Christopher A., and Raymond Larson. 1992. An explanatory and "argumentative" interface for a model-based diagnostic system. In Proceedings of the 5th annual ACM symposium on user interface software and technology: ACM Press. 43-52.

Miner, Cameron S., Denise M. Chan, and Christopher Campbell. 2001. Digital jewelry: wearable technology for everyday life. In CHI '01 extended abstracts on Human factors in computer systems: ACM Press. 45-46.

Neale, Dennis C., and John M. Carroll. 1997. The role of metaphors in user interface design. In Handbook of human-computer interaction, edited by M. Helander. Amsterdam; New York: North-Holland. 463-488.

Norman, Donald A. 1990. The design of everyday things. 1st Doubleday/Currency ed. New York: Doubleday.

———. 1998. The invisible computer: why good products can fail, the personal computer is so complex, and information appliances are the solution. Cambridge, Mass.: MIT Press.

Reeves, Byron, and Clifford Ivar Nass. 1996. The media equation: how people treat computers, television, and new media like real people and places. Stanford, Calif.; New York: CSLI Publications; Cambridge University Press.

Sacks, Harvey, Emanuel A. Schegloff, and Gail Jefferson. 1974. A simplest systematics for the organization of turn-taking for conversation. Language 50 (4, Part 1):696-735.

Schegloff, Emanuel A. 1992. Repair after next turn: the last structurally provided defense of intersubjectivity in conversation. American Journal of Sociology 97 (5):1295-1345.

Schmandt, Chris. 1990. Illusion in the interface. In The art of human-computer interface design, edited by B. Laurel and S. J. Mountford. Reading, Mass.: Addison-Wesley Pub. Co. 335-343.

Schön, Donald A. 1993. Generative metaphor: a perspective on problem-setting in social policy. In Metaphor and thought, edited by A. Ortony. Cambridge [England]; New York, NY: Cambridge University Press. 137-163.

Searle, John R. 1969. Speech acts: an essay in the philosophy of language. London: Cambridge U.P.

Shechtman, Nicole, and Leonard M. Horowitz. 2003. Media inequality in conversation: how people behave differently when interacting with computers and people. In Proceedings of the conference on human factors in computing systems: ACM Press. 281-288.

Slembrouck, Stef. 2003. What is meant by "discourse analysis?" [website], Nov. 19 2003 [cited Nov. 22 2003]. Available from http://bank.rug.ac.be/da/da.htm.

The Stanford HCI design learning space 2003. [website] [cited Sep. 12 2003]. Available from http://hci.stanford.edu/hcils/concepts/metaphor.html (no longer exists).

Stubbs, Michael. 1983. Discourse analysis: the sociolinguistic analysis of natural language. Chicago; Oxford, [Oxfordshire]: University of Chicago Press; B. Blackwell.

Taylor, M. M., F. Néel, and Don G. Bouwhuis. 2000. The structure of multimodal dialogue II. Philadelphia: J. Benjamins Pub. Co.

Thórisson, Kristinn R. 1995. Multimodal interface agents and the architecture of psychosocial dialogue skills. Thesis proposal for PhD, Media Laboratory, Massachusetts Institute of Technology.

Thórisson, Kristinn R. 1999. A mind model for multimodal communicative creatures and humanoids. Applied Artificial Intelligence 13 (4):449-486.

Weizenbaum, Joseph. 1966. ELIZA — a computer program for the study of natural language communication between man and machine. Communications of the ACM 9 (1):36-45.