Principles of Collaborative Coordination: An Agenda for Furthering Interactive Computational Intelligence

Derek Brock

Naval Research Laboratory
4555 Overlook Ave., S.W.
Washington, DC 20375
brock@itd.nrl.navy.mil

Abstract

In the past two decades, the underlying interaction model for most software use has arguably remained unchanged and is little more than an expedient design based on certain superficial features of face-to-face communication that not only fails to accommodate an important range of users' native interaction skills, but also devotes few computational resources to a useable artificial understanding of the process, progress, and products of the implied collaboration. This short paper examines how principles at work in people's collaborative activities with each other play out in software use and takes the position that computational implementation of these fundamental human interaction concepts continues to be a relevant agenda for the artificial intelligence and human-computer interaction communities.

Introduction

Human interaction with computational systems is aptly characterized by Susan Brennan as "a kind of coordinated action that bears many similarities to conversational interaction" (Brennan 1998). In making this comparison, Brennan goes on to say that the kind of coordinated action she is referring to is collaborative action, or more specifically, purposeful coordination between two or more actors. Between people, this simply means doing things together. In many respects, conversation is an epitome of this process, but so too are ostensibly nonverbal interactions such as shaking hands, passing food around a table, or executing a double play. In each of these undertakings, a core set of set of principles can be seen to be at work that make it possible for each party in the collaboration to coordinate his or her part in accomplishing their social purposes. Ideally, software user interfaces are conceived as designs for the coordination of information processing, and so, are implicitly collaborative in interactive settings in the rudimentary sense that they make it possible for users to engage in purposeful activities with them that facilitate goals that might otherwise be difficult for people to achieve on their own. But while the means and conventions for interacting with computational systems may be verbal or nonverbal or some mix of the two, the interaction model underlying a given interface is rarely more than an expedient, *ad hoc* design that not only fails to accommodate an important range of users' native interaction skills, but also devotes little or no computational resources to a usable artificial understanding of the process, progress, and products of the collaboration. The goal of this short position paper, then, is to examine how the principles at work in people's collaborations with each other function in software use and continue to be a relevant agenda for the artificial intelligence and human-computer interaction communities.

People ordinarily take it for granted that they are essentially alike in their abilities to perceive, recall, and reason about activities they participate in alone or together. As a consequence, when they set out to do something together, they rely on an important set of expectations about these abilities. They presume that each other's participatory experiences in their collaboration will be largely the same in terms of what each attends to and what each understands or believes has transpired. They also count on each other to possess and coordinate a range of complementary interaction skills that is made up of procedures for orienting each other's focus and attention in various ways. Underlying these skills are the complementary notions of speaker's meaning and addressee's understanding, which in turn rely on common ground, a term of art for the collective body of knowledge that speakers and addressees take to be justifiably shared. People rely on each other to express their intentions with means that are easily worked out and, in turn, depend on each other to provide evidence that their intended meaning has either been understood or needs further elaboration. This process is called grounding, and without each other's functional cooperation, the coordination of content and process in their social interactions would present insurmountable difficulties and ultimately foil their collaborative purposes.

These and other conceptualizations of the interrelated factors that play a role in the mechanics of people's interactions with each other have, in recent years, become increasingly well-articulated, particularly through the work of Herbert Clark and his colleagues. Although Clark's work has grown out of what he calls an "action tradition" in the study of language he traces to the philosophical

work of Austin, Grice, and Searle (Clark 1992), he has come to the conclusion that language use in its conventional, linguistic sense is simply a class or form of a larger and more basic social phenomenon, namely, jointly coordinated actions between people that involve meaning and understanding through the use of signals (Clark 1996). Taking the position, then, that all forms of functional signaling between people can be construed as language use, Clark's framework provides a range of useful insights about people's collaborative skills and the coordination of meaning.

As a result, much has been made of the value of models of language use for the design of human-computer interaction. Brennan, for instance, has focused on what she calls the grounding problem in computer user interfaces and makes the point that "electronic contexts are often impoverished" in terms of what they interactively convey to users (Brennan 1998). The idea is that interfaces should always be able to indicate to users whether or not their inputs are accomplishing what they are intended to do. She argues that the current dominance of so-called direct manipulation interfaces is not as much due to their graphical idiom and use of familiar workplace metaphors as to the way, in certain respects, they represent and provide verifiable evidence of the effects of users' actions. Her concern, though, turns on the fact that interface designs generally lack an underlying, structured model of the grounding process, and so, fail in unanticipated ways to reliably indicate their processing state in response to command interactions. In this view, user inputs should be modeled as jointly grounded contributions to what is functionally a dialogue. Just as human addressees indicate the strength and state of their understanding of what has been said to them with their responses, so too should an interface coordinate its acceptance and uptake of a user's input by responding with appropriately positive or negative evidence of its state of understanding, which can vary from attending or not attending through acting on an input and reporting the result. Similarly, according to this model, an interface that is capable of treating the user as an addressee should allow the user to coordinate with it on the degree his or her understanding through a range of response options. Not only does this provide users with better information for diagnosing when their actions have unintended effects the underlying system is not designed to evaluate, but it also, when properly implemented, gives users of mixed initiative systems better opportunities to correct misunderstandings and more fully direct the course and purpose of the interaction.

Viewing Human-Computer Interaction as an Instance of Language Use Between People

In another example of how models of language use are relevant to the design of human-computer interaction, Brock argues that software use must be viewed as a true instance of language use, not because of any similarity between its interaction model and human discourse, but because user interfaces are design artifacts that function as a means for coordinating a particular class of collaborative activities between people (Brock 2002). In this view, the real participants are not the user and the machine, but people acting as themselves in the roles of designer and user.

Several important insights are made possible by this perspective. First, working with computer software in this view is properly characterized as an activity with at least two conceptual layers of action. In language use, layers are conceptual domains of meaning and action that arise above, outside of, or beyond the immediate domain of activity. All interactions involve a primary layer in which the parties involved coordinate the direct business of their activity as themselves. In the primary layer of software use, the designer and the user proceed by grounding the designer's meaning on the basis of the presentation, arrangement, and actions of artifacts and other signals in the interface. Secondary layers and beyond are identified by references to entities, settings, and meanings that are not necessarily understood to be present in the layer or layers that lie below. In human-computer interaction, a secondary layer is always present in the sense that users take part in a collaborative activity that is designed to imply that the computer, rather than the designer, is the user's real counterpart.

Another issue that arises when software use is viewed as an instance of language use is the notion of interaction settings. The setting or circumstance in which a particular interaction takes place has substantial consequences for how its participants coordinate meaning and understanding and any related actions. In particular, the kind of access a setting, along with its medium and other constraints, affords people to each other largely determines the sorts of communication skills they will need and how difficult it will be to accomplish their ends. For people, face-to-face settings are generally viewed as the most basic and written settings as possibly the most demanding. Software use might seem at first to be an activity that takes place in a face-to-face setting, but this, in fact, is only the case in its second layer of activity, not in its first. Instead, the primary layer of software use must be understood to take place in a written setting because the interface design has been entirely worked out and produced in advance of its presentation, which users then take up at a later time. In most respects, this is no different than the design of a web page or, more to the point, the writing of an essay or a book. Just as printed material requires a reader's collaboration to be understood and to achieve its social purpose, so too, a user interface design requires a user's collaboration for it to be engaged properly, its computational functions exercised, and the social result its designer intended—facilitating the user's computational goals—achieved.

If human-computer interaction, then, is a collaboration between users and designers, albeit in a written setting, it should be possible to examine how a range of language use principles function in the joint coordination of this activity. In Clark's framework, the coordination of meaning and

understanding is idealized as a coordination problem that one person poses for another and then both set out together to solve. To do this, and do it efficiently, they require certain ingredients and certain principles. The ingredients are signals and the pair's common ground, and the principles apply to how these ingredients are used. Common ground is taken to be built in part through shared experience and in part through the process of grounding knowledge and meaning in people's interactions. As a consequence, between collaborators, it can be roughly conceptualized as a three-part, mutual idea of their activity together: their common ground at the start, the state of their activity now, and what has openly happened so far. People expect each other to maintain an awareness of this mutual knowledge so they can justify and refer to it with signals, and so, use it to introduce and coordinate what they decide to do together next. In posing coordination problems for each other, they look for premises to be met that will help them quickly converge on viable solutions and reduce their combined effort. In particular, an idea of the solution should already be in mind given what is common ground, all of the information needed to reach the solution should be provided (with the signals involved functioning as a basis or device for indicating the coordination itself), and the solution in this context should be conspicuous and readily apparent. In written settings, meeting these premises of solvability, sufficiency, and joint salience is even more important because addressees come to the discourse after its presentation has been designed and have little recourse if something proves to be incomprehensible. Writers intuitively know this and work hard to find the right devices to coordinate the grounding of their message. Capitalizing on devices that are likely to be salient and familiar to addressees is one of the most important considerations in this process since the inherent immediacy of this type of information substantially reduces the effort needed to work out intended meanings. Two kinds of devices that readily serve this purpose, and are often found together, are external representations and conventions. Making use of external representations involves the placement or appropriation of relevant items or artifacts in a physical setting to indicate how coordination should proceed. Conventions have an analogous procedural function. They instantly signal how to move forward because they are, in effect, settled and reliable courses of action for commonly occurring coordination problems in people's dealings with each other. Written settings epitomize the use of these devices. Books and magazines are entirely composed of external representations, which in turn, are mostly organized as systematic, linear streams of conventional signals. An important strength of this organizational convention is its straightforward correspondence to the common ground notion of what has happened so far, which makes the knowledge established through this medium easy to review when misunderstandings or other problems occur.

Most of the features of language use outlined above can be readily found in the first layer of software use. Like the media of any written setting, its message is predominantly coordinated with the user through set of external representations and conventions that function as coordination devices. The user grounds the designer's meaning through actions that he or she decides to coordinate on the basis of the presentation, and, as is the case in most written settings involving specialized representations and procedures, the activity presupposes a certain degree of literacy, which, in this case, concerns computers and means for interacting with them. In coordinating the software's use, both the designer and the user rely heavily on each other's language use skills. Designers strive to create interfaces whose functions are self-evident on their face, and users expect to be able to converge on viable coordination strategies for their projects with little or no effort beyond what they would need in any other written setting.

A significant difficulty for the smooth function of this collaborative process, though, lies in how the organization of information in first layer of software use differs from that of conventional written media. Almost all user interface designs provide users with a large number of interaction starting points and are consequently organized in a fundamentally nonlinear manner. This first layer content presentation idiom arguably has its basis in the second layer ideal of face-to-face communication, but it has unintended consequences for the process of meeting the collaborative premises of solvability, sufficiency, and joint salience. Since this organizational scheme intentionally gives opportunistic control of the activity to the user, the designer must relinquish his or her control over the process of building a structured body of common ground in the careful manner of a linear presentation. The result is that users, through no fault of their own, are inevitably led into coordination problems they have been given insufficient information to solve due to a contingent branching of *their* focus that few if any nontrivial interface designs can fully anticipate. The problem is further compounded for users by this written medium's essential lack of an intuitive representational correspondence with the notion of what has openly happened so far in the collaboration. Evidence of interactions is often underrepresented, evanescent, or, far more often, simply put away as the work moves forward, and there are generally no provisions for indexing or reviewing a linear history of the activity.

The picture that emerges in the first layer of software use, then, is one of inherently incomplete common ground between designers and users. However, the knowledge that is successfully grounded in the first layer of this collaboration is wholly relevant to the second layer's notion of face-to-face collaboration with the computer. Without it, the purposes involved in software use would be incomprehensible and interactions with computers would be opaque exercises—as, in fact, some are. Difficulties that arise from incomplete common ground are seldom show-stoppers, but they nevertheless have a withering effect on collaborative strategies users might wish to employ in the

second layer. In particular, it can be readily argued that users ordinarily keep track of a running idea of their interaction with the computer that corresponds in all respects to the rough, three-part characterization of common ground given earlier, albeit with the important understanding that their implicit collaborator is an information processing machine. However, since there is generally no provision for the computer to maintain a corresponding representation of this knowledge and, further, since there are few if any affordances for the user to participate in the coordination of such a representation, users are left to justify their own notion of events and some of their most important face-to-face language use skills go largely unused.

Interactive Coordination of Artificial Understanding in Software Use

As the preceding material has attempted to suggest, many of the difficulties that arise in human-computer interaction are due to presentation idioms that only attempt to honor certain surface features of face-to-face collaborations such as straightforward turn taking and the use of numerous starting points for interaction. Although these conventions of design are well-intentioned, they necessarily lead to violations of language use principles that are essential for meeting users' collaborative expectations in the coordination of software information processing tasks. At a minimum, giving users access to a contextual record of task events that have transpired so far would alleviate one confound of nonlinear designs by making it possible for users to reconcile their own conception of the activity with what in fact has happened. A more sophisticated implementation of a representation such as this would also allow users to index and selectively return to, or make forward use of, past computational products and states. However, even remedies of this sort still leave the burden of comprehension on the user, as it is for addressees in all written settings. Hence, the broader intention here is to emphasize the essential role that the technical notion of common ground must play in the design of software user interaction if collaborative activities with computers are to move to a new and sustained level of utility.

Representation of common ground in human-computer interaction is an important paradigm for current research in both software user interface design and artificial intelligence for at least two reasons. First, it is an inherently hard problem. People's notion of their common ground is far more than a body of shared conventions and a mutual record of events. For it to provide genuine collaborative utility in the second layer of software use, it must also entail a substantial range of computational adjuncts whose ancillary roles include common sense reasoning, functional awareness of how grounding events relate to the purpose and goals of activities in the task domain, the conception, bounding, revision, and generalization of representations, and providing access to the products of this artificial cognition in ways that appropriately meet the expectations and needs of users. All of these considerations involve issues that are prominent themes in artificial intelligence and, indeed, all of the collaborative principles of language use Clark describes have become central concerns in computational linguistics. Arguably, though, very little has changed in the basic interaction model of software use in the past two decades, and the advent of intelligent user interfaces appears to be waiting for a unified approach. Such an approach could begin with a standard contextual representation, such as the record of events proposed above, that would then serve as a basis for the application of computational techniques designed to reason about this information and provide access to it in ways that are intended match the expectations of users in the normal exercise of their face-to-face interaction skills. In the near term, implementation of artificial understanding is likely to require an integrated mosaic of computational techniques whose products can be reconciled across a range of representations, such as the approach developed in Polyscheme (Cassimatis 2002). Managing the process of grounding (e.g., Cahn and Brennan 1999, Traum 1994) and the coordination of representations are likely to be two of the most useful augmentations to the collaborative process for users. Concepts that are relevant in grounding include access to earlier products, salience and other issues for coordinating meaning and action, justification of common ground, and repairs. Coordinating representations is also a dominant part of grounding. Concepts that are relevant to this process include collaboration in referring (e.g., Heeman and Hirst 1995) and the use of conventions (e.g., Alterman and Garland 2001).

A second motivation for pursuing computational strategies for the maintenance of common ground is to share the user's cognitive load. This is a long range goal that ultimately will augment the range of human concerns computers are likely to be useful for. A brief, but hardly speculative, list of domains in which such a capacity would be invaluable include situation awareness and decision support in military environments and interactive robotic assistance in any number of critical settings. A great strength of social interaction for people is its capacity to corroborate and revise their perceptions and to expand their knowledge and experience. Persistent computational representation of collaborative activities between humans and computers and the development of appropriate interaction strategies and techniques for its coordination and use will prove in the end to be an indispensable requisite if people are to benefit from the many advantages robust machine cognition is certain to bring to human endeavors.

References

Alterman, R. and Garland, A. 2001. Convention in Joint Activity. *Cognitive Science.* 25(4): 611-657.

Brennan, S. E. 1998. The Grounding Problem in Conversations With and Through Computers. In S. R. Fussell and R. J. Kreuz eds. *Social and Cognitive Psychological Approaches to Interpersonal Communication,* 201-225. Hillsdale, NJ: Lawrence Erlbaum.

Brock, D. 2002. A Language Use Perspective on the Design of Human-Computer Interaction. *Proceedings: Office of Naval Research TC3 Workshop: Cognitive Elements of Effective Collaboration*. University of San Diego. CA.

Cahn, J. E. and Brennan, S. E. 1999. A Psychological Model of Grounding and Repair in Dialog. *Proceedings, AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems,* 25-33. North Falmouth, MA: American Association for Artificial Intelligence.

Cassimatis, N. L. 2002. Polyscheme: A Cognitive Architecture for Integrating Multiple Representation and Inference Schemes. Ph.D. diss., Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Clark, H. H. 1992. *Arenas of Language Use.* Chicago, IL: The University of Chicago Press.

Clark, H. H. 1996. *Using Language.* Cambridge, England: Cambridge University Press.

Heeman, P. and Hirst, G. 1995. Collaborating on referring expressions. *Computational Linguistics*, 21(3): 351-382.

Traum, D. R. 1999. Computational Models of Grounding in Collaborative Systems. *Working Notes of AAAI Fall Symposium on Psychological Models of Communication*. 124-131. Menlo Park, CA: American Association for Artificial Intelligence.