Conversational Analysis and Human-Computer Interaction Design

Sarah A. Douglas

CIS-TR-93-29
December 1993

Department of Computer and Information Science
University of Oregon

Conversational Analysis and Human-Computer Interaction Design

Sarah A. Douglas
Computer and Information Science Dept.
University of Oregon
Eugene, OR 97403
(503) 346-3974
douglas@cs.uoregon.edu

To be included as a chapter in: SOCIAL AND INTERACTIONAL DIMENSIONS OF HUMAN-COMPUTER INTERFACES, Peter Thomas, Editor, Cambridge University Press, 1994.

ABSTRACT

One area of concern common to all designers of interactive systems is pinpointing where the design fails for the first-time user and how to improve it. Many design methods are analytic--good at tearing apart problems but unable to synthesize improvements. In this paper I will describe research I have done using one method derived from Conversational Analysis which not only aids in analysis but also provides the necessary synthetic element. The paper discusses the method in detail and its integration with standard software prototyping techniques and usability studies. A detailed example drawn from a three year project which developed a cardio-vascular simulation for use in teaching biology illustrates the method. Finally, an informal confirmation of the learnability, practicality and universality of the method was made through several years of teaching it to programmers in user interface design classes.

INTRODUCTION

From Idealism to Anarchy: The real world of user interface design

How do we create successful designs for interactive systems? For ten years this rather complex question was answered by a narrow definition of the word "successful" in early and persistent efforts by Card, Moran and Newell (Card, Moran et al. 1983; Newell and Card 1985; Newell and Card 1986) to create an engineering science of human-computer interaction (HCI) based on a cognitive science of the individual user. In their Tayloresque world "successful" meant reduction of the design to observable and measurable user behavior: achieving minimal speed of routinized performance, achieving minimal speed of learning, elimination of user "errors" which were regarded as wasteful of time, and a completeness of functionality. The designer performed a task analysis involving the specification of the design as a set of user goals accomplished by sequences of user mental or physical actions such as typing on the keyboard. For an elaborated example of this approach to the design of text editors, see (Roberts and Moran 1983).

During the past ten years the HCI community, including both researchers and designers, has failed to accept this approach to design. In particular, the Card-Moran-Newell approach has not been able to deal well with non-routinized human behaviors of managing trouble, problem solving and learning. For example, although a cornerstone of the theory depends on the concept of goal-directed behavior, the designer does not actually determine real human intentions but instead "discovers" a Socratic set of ideal and complete goals based on an abstract analysis of what work is to be accomplished, i.e. its functionality as a task environment. No attempt is made to analyse the context of work such as what the users already know or problems they have with existing products. This makes it devilishly difficult to predict where the user will have trouble or make "errors." The entire design effort focuses on the creation of a perfect, i.e. error-free, design rather than a design which can robustly respond to the infinite possibilities of failure.

The design process is typically characterized by initial designs created by programmers and then possibly reviewed by an expert in human factors or psychology. The standard design aides are the prior experience of the designer, expert designer heuristics published as user interface guidelines, task analysis tools such as the keystroke model, and simulated user behavior such as the cognitive walkthrough. There is no contact with real users in real environments of use, in other words, this approach to design is context-free.

Criticism of this context-free methodology has come from both practitioners who have studied the effectiveness of different design tools as well as from researchers with more theoretical interests. A study by (Jeffries, Miller et al. 1991) compared three methods of evaluating designs without actually involving the human user with a technique called usability testing. The non-empirical methods were human expert evaluation(Nielsen and Molich 1990), guidelines(Smith and Mosier 1986; Apple Computer 1987; Mayhew 1992), and the cognitive walkthrough(Lewis, Polson et al. 1990). Usability testing takes a working version of the software and tries it out on a user. Observations as to usability are made by the designers and used to make modifications to the design. The technique seems to encompass any method of direct observation of user performance with the purpose of discovering design flaws. The Jeffries et al. study found that usability testing uncovers the most serious interface failures, is very good at finding recurring and general problems, avoids low-priority problems, and does not produce any false problems. As a design tool it was preferable to the other three (Jeffries and Desurvire 1992). It is clear that the effectiveness of usability testing is that it attempts to create some kind of context when evaluating the design.

Continuing the call for adding more context into the development of software, Clayton Lewis, a leader in the HCI research community, elaborates on the failure of the Card-Moran-Newell approach in his paper "A research agenda for the nineties in human-computer interaction" (Lewis 1990). He cites three major reasons: the failure to accommodate the context of use, lack of iterative design process and an inability of this type of theory to modify itself from actual design experience. For Lewis these failures, which he calls *context*, *process*, and *systems*, should become the major issues for HCI design in the nineties. Due to their importance, brief review of what Lewis means by context, process and systems is in order.

Context. Primary criticisms about lack of context have been mainly ethnomethodological arguments (Suchman 1987; Whiteside, Bennett et al. 1988). Lewis does not define context in his paper other than to characterize Suchman as saying that the "theory is inside out: that the social context in which cognition occurs shapes behavior to such an extent that what is needed is a theory of social interaction modified to account for cognition not the other way around." ((Lewis 1990) p. 129) In short the Card-Moran-Newell approach fails to take into account *interaction*. Design is based solely on an individual psychology of the user ignoring the concept of interaction which requires at a minimum a pair of individuals, be they human-human or human-computer. In some profound sense the users seem like robots with routine procedures whose only contact with their environment is some sort of triggering event that launches them into an execution of pre-planned behavior.

Process. The second criticism of an engineering science approach involves the process of design itself. The central idea is that it is more important to worry about how to obtain and act on appropriate feedback on the effectiveness of a design than to worry about the perfection of the initial design. This is called an *open-loop* approach (Gould 1988).

Systems. The final failure cited by Lewis is the inability of design driven by engineering science to reincorporate the lessons of implementation back into the theory (Carroll and Campbell 1986; Carroll and Kellogg 1989). Despite ten years of trying to work through designs based on the Card-Moran-Newell model new designs are mostly shaped by the systems that have already been built and the experiences with them. Graphical user interfaces are an example of this.

In some sense, context, process and systems are very related to each other and point out the complexity or even futility of any science of "applied" human behavior. Providing a context of use to continuously inform the design creates the possibility of improving design through feedback from cumulative systems experience of both designers and users. This sounds like anarchy to those schooled in the careful task analysis of engineering science, but it emphasizes the design *process* rather than the design *product.* It also requires that the focus of design shift from creating error-free efficient human performance to creating consistency with users' expectations and intentions and a support of the users' own processes of repair.

In the remainder of this chapter, I will discuss ideas based on conversational analysis that I have been developing over a five year period that attempt to create a systematic method of design which seriously incorporates context, process and systems concerns.

METHOD
What does talk have to do with it?

In this section I describe this design method in detail. Briefly, a technique called constructive interaction generates context information about the usability of the design by first-time users. The choice of first-time users is made because the trouble they encounter discloses the work required to understand the system's behavior that is masked by the experienced user. Then using conversational analysis as both an analytic as well as a synthetic technique, the designer detects the trouble and decides whether it is a design failure. Further conversation analysis points to repair of the trouble and ultimately offers suggestions for software design changes. Finally, constructive interaction and conversational analysis are placed within an overall design process.

Constructive interaction.

Constructive interaction uses two participants who are given a set of problems to solve or tasks to perform. The participants' activities are recorded using both video, audio and possibly computer generated records such as keystrokes and pointing actions. It is helpful to compare constructive interaction to another method called *protocol analysis*, or "thinking aloud" (Lewis 1982; Ericsson and Simon 1984) which has been used extensively in the HCI community. Protocol analysis uses only one participant. This participant is given a set of problems to solve or tasks to perform and encouraged to "think aloud." The participant is recorded in a similar manner to that of constructive interaction. Both constructive interaction and protocol analysis may be repeated with other participants.

The primary goal of protocol analysis is to uncover the underlying mental life of the participant. Protocol analysis is useful in design studies because it provides a method for identifying "problems" the user may have with the design, some elaboration as to why and perhaps ideas for improvement. Unfortunately this method has one major limitation familiar to ethnomethodologists: it fails to recognize language itself as a social process. The presence of the observer and the demand to talk may cause participants to "make up stories". Also, it is highly improbable that reports of a person's own mental processes are scientifically valid(Nisbett 1977). Despite these limitations, protocol analysis provides the designer with two critical pieces of information: Where the design fails to achieve what the designer expects, and some insight into why.

Constructive interaction attempts to elicit verbal information about problems in a more naturally occurring conversation. By using two participants a situation of collaborative problem solving is created whereby each participant must inform the other in an explicit verbal record about problems, causes, and solutions. Although there is no doubt that the technique derives from early studies of conversation by Sacks et al. in the early 1970s, the method was first described in published material in the mid-1980s by Miyake (Miyake 1986) who used it to study human problem solving during the repair of a sewing machine. The person most directly responsible for the development of this technique and its use in the analysis of human-computer interaction is (Suchman 1987) who did extensive work on an computer-based expert help system for a Xerox copier.

Conversational analysis.

Once the constructive interaction is completed the designer is still left with the problem of systematic analysis of a primarily verbal record. One method is Conversational Analysis (CA). Unlike researchers like (Norman and Thomas 1990) who have used CA and its "rules" to create design guidelines, this use of CA centers entirely on the interpretation of human communication (including so-called non-verbal language) at a very detailed level with a focus on the detection of trouble and analysis of repair.

(Suchman 1987) pioneered the use of CA to analyse human-computer interaction. Her primary interest in doing this type of analysis was to compare the behavior of human and machine as an interactive exchange modeled on human-human communication. Suchman was interested in where and why that exchange failed in the broadest sense and the fundamental philosophical issue of whether it is possible to create a software system that could know and explain its own behavior. Although she borrowed heavily from the interactive emphasis of earlier conversational analysts, she radicalized CA by including the computer as one of the "social" participants. In a sense, the program of the machine "stands in" for the human designer-programmer in the interactive exchange. This requires that the programmer attempt to anticipate all possible intentions of the other participant (the user) and states of interactive communication.

Consequently Suchman reduced her original videotapes to a simple framework of interaction between the user and the machine:

THE USERS		THE MACHINE	
I Actions not available to the machine	II Actions available to the machine	III Effects available to the user	IV Design rationale
E1			
E2			
E3			

Table 1: Suchman's Analytic Framework

The general process of Suchman's analysis was to use this framework to create a transcript of the major episodes of trouble, denoted "E1," "E2," and "E3" above. Included in column I, "Actions not available to the machine," were all verbal and gestural events of the users. Column II recorded the users actions at the semantic level of the interface program such as "Selects START button." The third column contained the presentation state of the interface program in either visual or audio messages to the user or other physical actions that represented behavior of the machine, such as the sound of initiating a copying process. The final column gave the designer's rationale to the effects in column III. Suchman found this a very practical framework since the center columns, II and III, were essentially the "interface," while the two outer columns, I and IV, represented the interpretations of the user and the designer in response to events in II and III.

Suchman found that the coherence of users' actions was largely unavailable to the computer. If the software designer had anticipated a particular sequence of detectable user actions (for example, a button press) and linked them correctly to the user's intent, then the system would respond correctly in spite of the lack of access to other user actions (for example, gaze). At other times there would be a lack of coherence between user intent and system response. These failures could be categorized as follows:

-- false alarms, i.e. users assumed that things had gone wrong when they hadn't, and
-- garden paths, i.e. users didn't know that or when things had gone wrong, when in fact they had.

Suchman's analysis was primarily theoretical and basically ended here. However, this use of conversational analysis can be extended to more detailed analysis of design failure. The major insight from the failure of the engineering science approach was the inability to understand a context of use which is primarily interactive and communicative. Suchman's constructive interaction method combined with conversational analysis links user expectations and intentions to actions and in turn to system response. As stated earlier, user expectations and intentions are revealed during breakdowns. Breakdowns are signaled by repetitions and restarts of a sequence of computer-based actions as well as talk about difficulties. However, the record may or may not yield an expressed intent. If it does not, the designer as an observer of patterns of talk and actions and as a member of the cultural community must infer intention.

Although Suchman was interested in complete breakdown—false alarms and garden paths—any breakdown yields valuable information for the designer that can potentially be used for redesign. Additionally, from the transcript the users own attempts to repair the breakdown by using the interface as a resource can suggest to the designer appropriate changes. For example, looking for instructions in a help subsystem, trying out a particular interface item whose behavior suggests a repair ("undo" menu item or an icon whose graphics suggest the appropriate function), restructuring the task, rewording instructions, etc. This synthetic use of conversational analysis has not been exploited in previous research. An extended example demonstrates this in a later section.

Since designers are primarily interested in points of breakdown, an entire transcript of the videotapes need not be made. After a brief review of the videotapes selected areas may need to be transcribed. A transcription is produced of both linguistic and physical actions and possibly descriptions of the environment. This transcript may be coded using conventions of CA which include pauses, breaths, overlaps in talk, stress and pitch changes, gestures including pointing, gaze, etc. and computer input and output events. The transcription conventions found in (Luff, Nigel et al. 1990) or (Suchman 1987) may be used. An issue as to level of detail exists and designers will need to trade off the time and expense of transcription with importance. From my experience the transcript may reveal previously unappreciated issues of users expectations and intentions.

When I first began to explore the use of CA in the analysis of human-computer interaction design, I found it very difficult to understand exactly what could be used. There did not appear to be a right way to do CA. Ultimately, I found these themes of the most value:

-- Concept of episodes
-- Concept of sequentiality (turns, overlapping, etc.)
-- Concept of non-verbal communication (gesture, gaze, pointing, etc.)
-- Concept of breakdown and repair
-- Concept of retrospective intentionality
-- Concept of communicative resources

The first three themes, that of episodes, sequentiality, and non-verbal communication, helped me to form a sense of how to create order out of an otherwise almost indecipherable transcript of verbal utterances. These elements can be found in any introduction to CA such as that by Robin Wooffitt (Wooffitt 1990). The concept of using a machine as one of the communication participants as Suchman does or even as a form of communicative medium such as the telephone may require some additions to the normal analysis. For example, Suchman uses the designer's rationale as the intentionality of the computer and Hopper when examining turn-taking in telephone conversations (Hopper 1992)extends the Sacks et al. model to encompass issues of syntax and prosody.

The second three themes, breakdown and repair, retrospective intentionality and communicative resources, helped me to interpret what was really going on in the interactions. Breakdown is critical in that it reveals through retrospective intentionality the structure of human action. And repair, through participants' use of communicative resources, demonstrates other aspects of the structure of interaction. For example, because of the style of user interface most commonly used today user intentions in interaction are expressed by the selection of objects and actions. Breakdown occurs when the users can't find expected objects or they select the wrong objects (wrong in the sense that the designer did not intend those objects to be an expression of that particular intention.) Breakdown also occurs when users procedures for accomplishing actions do not match the designer's composition of subsequences or termination conditions. Repair is expressed by searching for possible resources of objects and actions available in the designed interface. The usefulness of these concepts will be more apparent in the details of an example offered in the next section.

The overall design process.

The information available from the prior analysis of context of use should inform the design from the earliest stages and continuously during development in an iterative model of design. From the programming standpoint the rationale is that the greatest flexibility in design comes earliest when it is much easier to change things on whiteboards than in code. Designs can be implemented in paper storyboards and given to users for a very crude simulation of interaction. Although paper simulations allow earlier feedback on the success of the design, they cannot substitute for the complexity of true interactive computing.

Rapid prototyping (Budde, Kuhlenkamp et al. 1984) and high-level programming languages specially designed for user interfaces (Douglas, Doerry et al. 1992) attempt to finesse this problem with automation support designed for flexibility and incompleteness of a design. The difficulty with a rapid prototyping tool is that it may not support all the interface actions desired for the final product, generate efficient code or generate code in a language that will interface to an application such as a data base manager. All prototyping methods require a highly modular implementation strategy for the software. Choosing that modularity may be critical to the success of the prototyping approach.

Based on several years of experience with this method, I have come to use three pairs of participants at each cycle of design analysis. My reasoning is that it is the minimum number to differentiate universal problems from those which are more unique to individuals. The design team, including users in a participatory design approach, can review the video tapes selecting problem areas for transcription. After the team discovers all the problem areas and generates design solutions, these design changes are prioritized trading off both severity and the complexity of programming. A new prototype is produced and tested again.

AN EXTENDED EXAMPLE
Using CA to design the Cardio-Vascular Construction Kit

During one extensive software development project beginning in 1987 and extending over a three year period, I experimented with the use of constructive interaction and conversational analysis in informing design. Previously I had used CA methods to analyze human-human tutoring (Douglas 1991) in order to determine the feasibility of an AI tutor for teaching Japanese. The success of that approach convinced me to use it more extensively on development of a simulation program called the Cardio-Vascular Construction Kit (CVCK) for use in teaching biology labs at the university level. In the particular environment of the biology labs students often shared computers in a collaborative way which made constructive interaction very compatible with their actual context of use. We were also charged with developing workbooks for the biology lab exercises that used the software and thus were intimately involved with the actual use of the software in a learning environment. This software is now available nationally as a software product in the BIOQUEST biology lab package (University of Maryland, 1992) and is thus not a "toy" example. This software has also had almost no complaints about the design of the interface or usability.

The design team consisted of myself as the primary designer, two professional programmers and one biology teacher. Our use of constructive interaction involved bringing two same gender, similar age and ethnicity first-year biology students to our HCI laboratory. (We discovered that we would get much more cooperative discussion if we paid attention to potential social power asymmetries.) For each phase of the design we would typically bring in three pairs of participants. Very early in the design process, prior to software implementation, the participants worked with paper prototypes. Later, while using rapid prototyping tools (Douglas, Doerry et al. 1992), we worked with software implementations until the final program was completed.

The two participants were left alone in our lab and instructed to follow the instructions in the workbook. They were given a buzzer they could use to summon help. Our HCI laboratory was arranged to have three video outputs: one from a camera situated at a high elevation and acute angle behind the pair in such a way that we could see the CRT monitor and where they were pointing on it, a second camera at a side angle so that we could see the upper half of the participants' bodies with mutual gazes and talking, and a third direct NTSC output from the CRT monitor which was generated by the computer system running the software. (See Figure 1.) We also recorded from lavaliere mikes on each person to each stereo channel on the second camera output.

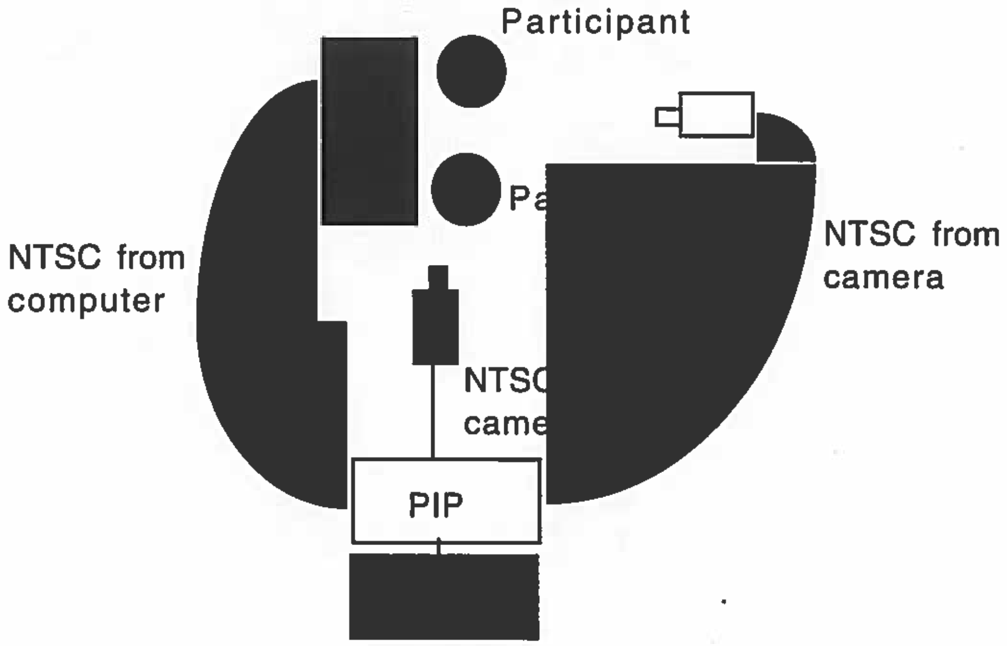

Figure 1: Laboratory Setup

After some experimentation we found that the first camera output from the monitor was quite adequate for analysis and thus dropped the third direct NTSC monitor output. We created a final videotape with a picture-in-picture processor with integrated images from the first and second camera. The image of the computer monitor with participants pointing occupied the major portion of the final picture and a reduced version of the long distance image of the two participants was placed in the upper right hand corner. This tape was then viewed by the whole team freely discussing possible problems, interpretations and alternative designs. For some events which we could not interpret we produced a written transcript which we would later individually analyze. This transcript was coded using standard CA techniques for verbal data enhanced with coding for participant pointing and gaze directed toward the computer monitor, participant computer input actions and any significant computer-generated user interface events using the transcription conventions found in (Luff, Nigel et al. 1990). After the team had discovered all the problem areas and had generated design solutions, we prioritized the fixes trading off both severity and the complexity of programming. We then produced a new prototype and repeated the constructive interaction method.

Our experience with the constructive interaction method is a valuable source of information about the use of ethnographics in detailed software design. Three primary aspects stand out. First, we were usually able to easily detect events where trouble occurred. Second, although we were able to usually detect trouble, we had difficulty diagnosing the cause. Third, ascribing a cause to the breakdown was often a key to the design solution. In addition, the repairs by the participants often provided us with possible alternative design solutions.

Detecting trouble with the design as revealed by participants' behavior is etecting trouble with the design as revealed by participants' behavior is fairly straightforward since the talk will be laced with expressions easily recognizable to the designers: "Oh my! What happened?", "Wait, wait..." "I don't understand this." What Suchman calls garden paths, i.e. users didn't know that or when things had gone wrong, when in fact they had, were infrequent but more difficult to detect. At a more structural level of analysis we found that we could detect trouble signaled by repetitions and restarts of a sequence of computer-based actions. In fact it was often difficult to pinpoint exactly where the problem started and a great deal of reviewing of the videotape was often necessary.

We found that diagnosing the cause of trouble was often problematic for the design team and dependent upon the ability of the design team to recognize the user's intent and the failure of the system to respond as expected. At times intent was not verbally expressed and even when expressed its intelligibility depended greatly upon the analysts' ability to place themselves in the context of the participants' work as social actors. This requires recreating rich visual and aural images from many sources, not just the already coded transcript. (Although we did not code the whole videotape because of cost and time constraints.) The group analysis of intent frequently required argumentation based on empirical evidence. A particular theory had to be convincing to the other members of the team. The evidence which would disclose an underlying intent was often not found in the verbal record, but in a complex interweaving of a history of talk, computer monitor state changes and user input actions. Participants frequently pointed to areas on the screen and used indexicals for reference resolution both for the efficiency of communication as well as a lack of lexicon. Again, the full video-tape and not a transcript was critical in the analysis. The designers were often able to compare what they expected the users intent to be, that is the design rationale, with the actual intent. This underlines the domain knowledge necessary to really understand what is going on. Showing the videotape to persons not members of the design team and not HCI experts familiar with the domain, provided no aid in analysis.

The following is an example of the richness available from this type of analysis. In this particular episode after the participants have read a brief tutorial description of how the software works, they are asked as the first task to construct a model system that replicates Figure 2. The reader can see that the CVCK interface is a standard tool palette with a central workspace for model construction. The user selects model components from the left edge of the palette and drags a copy into the workspace to the appropriate position. Because components are used in four different rotational positions, a design decision was made to have only one orientation of each component and use a "rotate" icon to create the correct orientation.

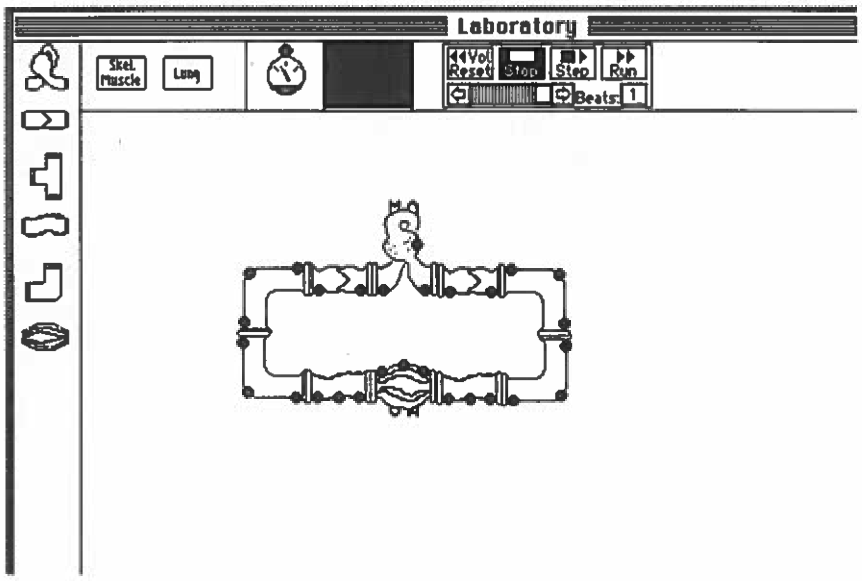

Figure 2: CVCK construction for study

The transcript begins after the participants have already placed one "elbow," i.e. right-angled vessel, component in the workspace. (For readability, the transcript has been somewhat edited.).

THE USERS		THE MACHINE	
I Actions not available to the machine	II Actions available to the machine	III Effects available to the user	IV Design rationale
L: OK. OK, what do we need (L & V read instructions)			
V: We need another (points to elbow icon on palette)			
	L clicks elbow	palette elbow highlights	add component
one = we need two three more	L drags elbow from palette	elbow added to workspace	place in model (selected object = elbow)

The second elbow is successfully added to the workspace and placed into position. They immediately discover that it is the incorrect orientation and attempt to correct that.

L: No, wait			
How do you //turn this thing around	L moves mouse in circles	elbow moves in circles	change position
V: //turn it			
L: Oh oh shoot			
V: That's right, just move it			
L: No it won't turn			
(twists mouse to turn it)		no effects	
V: OK.			

Here, the participants have attempted a direct mapping of their physical actions using the mouse to rotate the object. The first is to move the mouse around in a circle and the second is to actually rotate or twist the mouse by keeping it in one place. Neither works since in the first place the hardware recognizes moving the mouse in circles as a circular motion of an object on the screen and not a rotation of that object and in the second place, the hardware does not recognize twisting the mouse as a meaningful action at all. Thus, these two repair strategies fail. (And cannot be used by the designer as repairs to the design since they are outside the possibilities of design.) They try another approach:

L: Oh wait			
V: Put it over here			
L: Oh wait wait			
I have an idea			
maybe this works	L clicks on slider	elbow un-highlights	deselect component (no object selected)
	in control panel	slider highlights & moves	change simulation rate
this wait,			
where is that thing?			
here we go	L moves to rotate icon		
	clicks on right arrow	highlights rotate icon no effects	rotate component (no object selected)

This second strategy is to try to find an action that maps to a control icon that will rotate the elbow. Readers should note how the participants use the interface itself as a resource in their interaction. First the slider in the control panel is used and then L recognizes the "rotate" icon which is intended by the designer to resemble a compass. The designers intended that a component be selected, then rotated to the desired orientation by selecting one the four different points of the compass which represents that direction. Note that this is a compound procedure, first select the component, then select the compass point. Unfortunately L deselects the component accidentally by choosing the slider. She then recognizes this and attempts to redo the procedure:

V: OK now, put this one			
	L clicks on elbow drags it to left	highlights elbow	elbow selected change location
	L points to rotate icon, clicks just outside right arrow	unhighlights elbow	deselect component
L: Darn it		no effects	(no object selected)

L's failure here is interpreted by the designers as problem of designing the compass points too small. They are very tiny and the user is unable to "hit the target." A possible design change is to make them larger, but then they will exceed the maximum size of an icon in the interface software. L, however, attempts to follow the advice of her partner and abandon the task:

V: Turn it the opposite way			
L: It won't turn			
V: Then bring it back over here (points to elbow icon in palette)	L clicks on elbow drags it next to elbow icon in palette	highlights elbow elbow moves	elbow selected change position
	L clicks on elbow icon in palette	unhighlights elbow component	deselect component
		highlights elbow icon	add component
V: Where where'd that thing go?			

This is an unsuccessful attempt to put the elbow component from the workspace back into the palette and start a completely new problem. It can't be done and L tries one last time to get the rotate icon to work:

L: There	L clicks on elbow component	highlights elbow	elbow selected
Oh	L points to rotate icon, clicks right arrow	elbow rotates to right	rotate component
There we go			
What'd I do?			

Although L does finally complete the rotation problem, the designers have enough evidence at this point to attempt a new approach to rotation of components.

Once we, as designers, had determined the cause of a particular breakdown, it often suggested design solutions. However, and more crucial to this discussion, the repair efforts by the participants often suggested possible design alternatives. Participants repairs can be found by examining their detailed talk, and by their other actions in which they use the interface itself as a resource for furthering a repair. The first attempts by L & V to rotate the component, namely by direct physical actions with the mouse, were not possible design alternatives. However, we were impressed by L's three attempts to use an icon to rotate the component and thus remained committed to an icon command on the palette. (And did not choose another approach such as a menu based command.) In other words, the design rationale for accomplishing this action closely matched the participants' expectations and intentions. What failed was the actual details of the objects and actions.

We observed three problems with the existing design: a failure to recognize the rotate icon, a failure to select a component before selecting the rotate icon, and a failure get the cursor inside of a very small target. What we finally decided to do was change the rotate icon to a very simple type of icon button labeled with the word "rotate" and which when pressed, rotates a selected component. We also decided to add logic for trapping an attempt to use the rotate icon without a selected component. This trap then informs the user that a component must be first selected before it is rotated.

We have found that constructive interaction and conversational analysis, as I have described our usage above, is an invaluable tool in doing design. Though the method takes time for analysis with the whole design team, it also provides an arena and indeed creative environment for them to mutually work out the design in the presence of real evidence about human behavior.

EVALUATION

You can lead designers to water, but can you make them drink?

The previous example, I hope, has been compelling in arguing for the usefulness of this method in interface design. It promises to provide for the first time, a real method to an otherwise vague area called usability testing.

The most difficult aspect of the method is that it is primarily qualitative and is best learned through an apprenticeship with experienced analysts. But any design method must prove its value in at least several different ways. First, it must be learnable by those who stand the most to gain from it. Second, it must be practical in that the method is not too costly or too time-consuming. Third, it must be universally valid, that is that designers are able to discover the same problems and derive the same interpretations of cause, although not necessarily the same solutions.

In an attempt to respond to the first two concerns I have taught this general method for several years to my senior/graduate computer science course in user interface design where the students must use it in their final project. Many students have told me that it was the most exciting part of the design process and were very enthusiastic about it. Some of my students are now working for commercial software development companies and have implemented it within their design process. Few of the methods I have taught have had this kind of acceptance.

I still felt unsatisfied with its success, particularly concerning the third issue—universal interpretation. In order to test this out, I took my advanced graduate seminar in HCI and gave them selected chapters from Suchman's book, namely, Chapter Six "Case and methods," and Chapter Seven "Human-machine communication" (Suchman 1987). I also had them read Wooffitt's chapter mentioned earlier (Wooffitt 1990). I then had them review the videotape from the prior design session of CVCK. First, I had them analyse a particular breakdown which I pointed out, and second, I asked them to find all the remaining breakdowns in one hour of tape. The format of the homework was the following questions:

1) In the videotape at approximately 1075, there is an example of interface failure. Please do the following.

   a) Using the framework presented in Chapter 7 (cf. 122) and the notation of Chapter 5 (p. 96-97), create a transcript of the users' breakdown with the system.

   b) Suchman has stated that ". . . instructions rely upon the recipient's ability to do the implicit work of anchoring descriptions to concrete objects and actions." (p. 101) Discuss where in this example the users seem unable to do this.

   c) We have often discussed the problem of relating intent to behavior. In this episode discuss the programmer's expectations of the user's intent and actions, the actual user's intent and actions, and why there is a failure to match here.

   d) In this episode the users are able to repair the trouble they are in. How did they do this? What evidence do you have to support your interpretation *from the videotape?*

   e) Given your analysis of parts b-d of this problem, how would you redesign the interface?

2) Find all other episodes in the tape where there is interface design failure and answer parts a-e of question 1 again.

All twelve students in the class independently came to the same identification of breakdowns and almost exactly the same interpretations of causes. They were quite capable of doing the transcriptions although there were minor variations in details, such as a mumbled word. They did complain about the tediousness of transcription but felt that the transcript was invaluable. The most difficult areas of breakdown for them were the true garden path problems. They had difficulty identifying them and difficulty interpreting them.

CONCLUSIONS

As a result of the informal studies conducted above, I have great confidence in the design method proposed here. Only more general utilization will test its ultimate value. However, in closing I would like to address a few criticisms.

CA is an ethnomethodological approach and by its very nature takes a very radical view of the basis of human action. Part of that view demands that all action is context-based or "situated." But anyone can see that the type of situation created in constructive interaction is not the real situation of use. The participants, while not actors, are recruited. They are given tasks or problems by the designers. The environment in which they work has video cameras and other controlled features. Thus, constructive interaction is "situated action," but not exactly the same one that the final software product will encounter. The receding horizon of interaction itself creates a context with its own unique history that will theoretically never be identical to any other. How then, can I assert that this method is valid as a method of creating designs using a context of use?

Ultimately, I am left with the conclusion that design itself is always imperfect. This particular method stands to provide more direct information about context of use since it focuses on actual interaction itself. And, hopefully, I have shown in my informal evaluations that different designers do share a common set of interpretations of action, because they share a common culture with the participants.

There are also those critics that will say that this is not a new method; it is already well-received in the HCI community as usability testing. For example, a recent survey (Nielsen 1993) supports the estimate that 4-6% of corporate and industrial software development budgets are spent on usability engineering. But the most thorough analysis of usability testing defines it in rather vague terms

> ...*usability testing*, in which the interface is studied under real-world or controlled conditions, with evaluators gathering data on problems that arise during its use. These tests can offer excellent opportunities for observing how well the situated interface supports the users' work environment. . . . .The

> usability tests were conducted by a human factors professional, for whom product usability testing is a regular part of his job.
> (Jeffries, Miller et al. 1991, pp. 119 & 120)

This decription to me suggests that usability testing is a black art. Using the method I have proposed provides a systematic approach to it that has been sorely lacking.

I have attempted to show in the beginning of this chapter the failure of earlier approaches to human-computer interaction design which were based on engineering science. Engineering science failed to provide information about context of use for the design, an open-loop process where feedback from context was integrated into improvements in the design, and a recognition that systems design experience rather than abstract scientific principles has created more successful design. I then proposed a method of design usability information collection based on a controlled context of use called constructive interaction. This information is then analysed using techniques of Conversational Analysis which can provide not only a record of breakdown of the interaction, but also user-based strategies of repair which can possibly be incorporated by the designer into design improvements. Finally, I discussed an in-depth example of the method's use in a long-term software development project and my success at teaching it to several classes in user interface design. The next step is for wider acceptance and use.

REFERENCES

Apple Computer, I. (1987). Human Interface guidelines: The Apple Desktop Interface. Reading, MA, Addison-Wesley.

Budde, R., K. Kuhlenkamp, et al., Ed. (1984). Approaches to Prototyping. Springer Verlag.

Card, S. K., T. P. Moran, et al. (1983). The psychology of human-computer interaction. Hillsdale, NJ, Erlbaum.

Carroll, J. M. and R. L. Campbell (1986). "Softening up hard science: Reply to Newell and Card." Human-Computer Interaction 2: 227-249.

Carroll, J. M. and W. A. Kellogg (1989). "Artifact as theory-nexus." CHI '89 Conference on Human Factors in Computing Systems, ACM.

Douglas, S. A. (1991). Tutoring as interaction: Detecting and repairing tutoring failures. Teaching knowledge and intelligent tutoring. Norwood, NJ, Ablex.

Douglas, S. A., E. Doerry, et al. (1992). "QUICK: A tool for graphical user-interface construction by non-programmers." The Visual Computer **8**: 117-133.

Ericsson, K. A. and H. A. Simon (1984). Protocol Analysis: Verbal reports as data. Cambridge, MA, MIT Press.

Gould, J. D. (1988). "How to design usable systems." Handbook of human-computer interaction. New York, North Holland. 757-789.

Hopper, R. (1992). Telephone conversation. Bloomington, IN, University of Indiana.

Jeffries, R. and H. Desurvire (1992). "Usability testing vs. heuristic evaluation: Was there a contest?" SIGCHI Bulletin. **24:** 39-41.

Jeffries, R., J. R. Miller, et al. (1991). "User interface evaluation in the real world: A comparison of four techniques." Human Factors in Computing Systems, CHI '91, New Orleans, ACM Press.

Lewis, C. (1982). Using the "Thinking-aloud" method in cognitive interface design. Technical Report RC 9265. Yorktown Heights, NY, IBM Thomas J. Watson Research Center.

Lewis, C., P. Polson, et al. (1990). "Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces." Human Factors in Computing Systems: CHI'90, Seattle, WA, ACM Press.

Lewis, C. H. (1990). "A research agenda for the nineties in human-computer interaction." Human-Computer Interaction **5**: 125-143.

Luff, P., G. Nigel, et al., Ed. (1990). Computers and Conversation. London, Academic Press.

Mayhew, D. J. (1992). Principles and guidelines in software user interface design. Englewood Cliffs, NJ, Prentice-Hall.

Miyake, N. (1986). "Constructive interaction and the iterative process of understanding." Cognitive Science **10**: 151-177.

Newell, A. and S. A. Card (1985). "The prospects for psychological science in human-computer interaction." Human-Computer Interaction **1**: 209-242.

Newell, A. and S. K. Card (1986). "Straightening out softening up: Response to Carroll and Campbell." Human-Computer Interaction **2**: 251-267.

Nielsen, J. (1993). Usability engineering. Boston, MA, Academic Press.

Nielsen, J. and R. Molich (1990). "Heuristic evaluation of user interfaces." Human Factors in Computing Systems: CHI'90, Seattle, WA, ACM.

Nisbett, R. E. &. W., T.D. (1977). "Telling more than we can know: Verbal reports on mental processes." Psychological Review **84**: 231-259.

Norman, M. and P. Thomas (1990). "The very idea: Informing HCI design from Conversation Analysis." Computers and Conversation. London, Academic Press. 51-66.

Roberts, T. L. and T. P. Moran (1983). "The evaluation of text editors: Methodology and empirical results." Communications of the ACM **26**(4): 265-283.

Smith, S. L. and J. N. Mosier (1986). Guidelines for designing user interface software. Technical Report # MTR-10090, MITRE Corp.

Suchman, L. (1987). Plans and situated actions: The problem of human machine communication. Cambridge, Cambridge University Press.

Whiteside, J., J. Bennett, et al. (1988). "Usability engineering: Our experience and evolution." Handbook of human-computer interaction. New York, North Holland. 791-817.

Wooffitt, R. (1990). "On the analysis of interaction: An introduction to Conversation Analysis." Computers and Conversation. London, Academic Press.