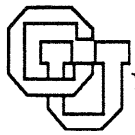
LOCATING THE RIGHT OBJECT IN A LARGE HARDWARE STORE: AN EMPIRICAL STUDY OF COOPERATIVE PROBLEM SOLVING AMONG HUMANS

Brent Neal Reeves

CU-CS-523-91



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

Locating the Right Object in a Large Hardware Store: An Empirical Study of Cooperative Problem Solving among Humans

Brent Neal Reeves

CU-CS-523-91 Spring 1991

Department of Computer Science, Institute of Cognitive Science
University of Colorado at Boulder
Boulder, Colorado 80309-0430
brentr@cs.colorado.edu

This research was supported in part by grants from the Colorado Institute of Artificial Intelligence (CIAI) and the National Science Foundation grant CDA 8420944.

Locating the Right Object in a Large Hardware Store: An Empirical Study of Cooperative Problem Solving among Humans

Brent Neal Reeves
Department of Computer Science, Institute of Cognitive Science
University of Colorado at Boulder
Boulder, CO 80309-0430
brentr@cs.colorado.edu

Abstract

Our goal is to build Cooperative Problem Solving Systems that serve the human user. Similarities between High Functionality Computing Systems and McGuckin Hardware led us to study customer-salesperson interactions there. This paper analyzes several conversations and uses the findings to evaluate computer systems built by the Human Computer Communication (HCC) group at the University of Colorado at Boulder. A critique of those systems results in new challenges for cooperative problem solving systems.

1 Introduction

The long range goal of the research within which this study was carried out is to build a theoretical framework of truly cooperative problem solving systems. As a frame of reference for this paper, consider the three reasons given below why current systems fall short of being truly cooperative. This paper will relate interactions at McGuckin Hardware to these issues and to the systems built in the Human Computer Communication group (HCC) at the University of Colorado at Boulder. For various reasons, current computer systems fall short of being cooperative:

- Systems are written from the designers point of view and are thus conceptually removed from the user's point of view. There is a gulf [Norman1986] between what the user wants to accomplish and how the system allows the user to proceed; a conceptual distance between a person's "situation model" and a designer's "system model" [FischerKintsch1986].
- Systems are built based on communication models that impede, rather than enhance productive interaction. Interactions are constrained and system-driven [FischerStevens1987].
- Systems that address problem solving and information retrieval are written with the assumption that people will approach the system with fully articulated queries. Yet people rarely construct fully articulated queries that need no reformulation at all [Williams1984, FischerNieper-Lemke1989].

The HCC group has built several systems that address cooperative problem solving from slightly different vantage points, each instantiating a key concept such as human problem-domain communication (Framer), mixed initiative dialogues (System's Assistant), incremental specification (Helgon), or other issues pertinent to human-computer interaction and problem solving. The similarities between high-functionality computing systems and McGuckin Hardware, described in the next section, motivated us to study problem solving interactions between customers and salespeople. We hoped that by studying those interactions, we might encounter similar approaches to the above mentioned shortcomings and establish better criteria against which to judge the successes of cooperative problem solving systems. The purpose was to confirm, in an empirical way, certain specific approaches instantiated in our systems, but also to challenge these same approaches by being open to new insights. This study shows how McGuckin Hardware store has overcome obstacles inherent in systems that are large in size and complexity and sheds light on

shortcomings of current systems, pointing out areas in which current systems should be improved to approach truly cooperative problem solving with users.

2 Appropriateness of the Study to our Research

Advances in computer hardware have made it possible to build software systems with a broad range of capabilities. Yet this power has come at the price of comprehensibility - the more powerful the systems have become, the more difficult they have become to master. Along with advances in technology, the cognitive costs to use these systems have increased. This trend has brought us to a time when there are fewer and fewer "experts": people who know an entire system. Thus various users will know different parts, but no one will know the whole system. As an example of a high functionality computing system, consider the Symbolics Lisp Machine: it has over 30,000 Lisp functions and over 3300 classes available in the object system. The documentation takes up over 4400 pages. Users don't complain about the power of such a workstation, indeed such power is welcome, yet they, like users of any platforms of this complexity and rich functionality have common problems:

- Users do not know about *the existence* of specific tools: If you don't know that a certain tool exists, you won't look for it, and you probably won't accidentally find it.
- Users do not know *how to access* tools: As the information space becomes larger and more complex, it becomes more and more difficult to find items, even if you know for sure that they do exist.
- Users do not know *when to use* tools: There are times when a tool intended for one use can be put to another use altogether. Yet this is not always obvious, so a person can be staring right at an item that could actually help them, and not know it.
- Users do not *understand the results* that certain tools produce for them: A mere description of a tool is not always enough to decide how a given item would work under certain new circumstances.
- Users cannot *combine or adapt tools* for special uses: Even though people are good at adapting an example, designing by reuse, it is not always clear just how to combine things to achieve a given effect.

These problems are remarkably similar to those encountered by customers who go to McGuckin to solve a problem. The problems associated with complex information spaces are true for McGuckin, which carries over 350,000 different line items on 33,000 square feet of retail sales space. Discovering items that you don't know exist, searching for items you think exist, knowing when a certain tool is appropriate, understanding the results of certain tools, and knowing how to combine several tools are common problems in a store as large as McGuckin Hardware. The following dialogues serve to illustrate the above mentioned problems.

Users don't know about the existence of specific tools

In the dialogue shown in Figure 1, the customer is unsure about how to attach a sign to a metal pole. Even if we assume a complete understanding of the problem, this is not enough to guarantee the knowledge of the best tool for the problem. Here the customer ends up buying a fastener that is introduced and explained by the salesperson.

---

1. C: I'm looking for a small fastener maybe 1/16.[1]
2. S: Okay. Plastic? Metal?
3. C: Well, what I've got is to fasten a sign on to a square pole. I've got a hole in the top and it fits fine and I got to get one on the bottom.

---

[1] C: is customer, S: is salesperson, I: is interviewer. Comments are in this font.

After looking at several fasteners, and asking a few more questions, the salesperson suggest a certain type of fastener.

4. S: How about a self-tapping bolt?
   Picks one up and shows it
5. C: Well, what uhh, well, this would probably do it, what about, would it come back out?
6. S: Oh sure. It'd come back out.
7. C: But once it's in?
8. S: As long as the hole is smaller than this thing, you can thread it in and out.

Figure 1: Attaching a Sign to a metal Pole

The customer does not know of self-tapping bolts and can therefore not ask for them. The salesperson analyzes the problem and determines that this kind of fastener would work, suggests it, then follows up with an explanation of how it works.

Users don't know how to access tools

The next dialogue shows that it can be difficult just to find items you know exist.

---

1. C: I need clips for tools where you shove it up in them and it holds
2. S: Yeah .
3. C: I mean not just a single clip, a bunch of them. We tried in housewares, the cheap little ones, tools only have like funny kind of ones. Where else could they be?
4. S: Garden center, for rakes and shovels and things like that
5. C: Would it be there?
6. S: Yeah.
7. C: Okay, I know where that is, thanks.

Figure 2: Finding Tool Clips

The customer is specific about the wanted item and even seems to know the store fairly well, but still cannot find the item.

Users don't know when to use tools

The interaction shown in Figure 3 involves a search for scales to weigh small animals and illustrates the concept of *applicability conditions*: the conditions under which an item can be used, especially for unintended purposes. The salesperson is able to ferret out the crucial element: namely that there be a platform large enough to hold something of a certain approximate size and weight. The fact that a scale is intended for food is less important than those features.

---

1. C: I'm looking for some scales and I saw some little tiny ones over here, but I need something that has a large platform on it, to weigh small animals on.
   Holds hands about 18 inches apart.
2. S: I would think something in our housewares department, for weighing food and things like that. Go on down to the last isle on the left.
3. C: Okay.

Figure 3: Scales for Small Animals

The customer uses an example item (the "little tiny ones over here") to differentially describe the intended solution. The salesperson extracts the crucial information and suggest an item intended for a different domain, yet useful for accomplishing the described task.

Users don't understand the results that certain tools produce for them

Figure 4 illustrates how people can know certain tools and still overlook issues that are crucial to the problem at hand. The customer wants strength, but the salesperson has to point out a crucial feature of that strength: that it comes at the expense of brittleness.

---

1. C: So if I were going to hook something would this be the best thing? What I'm going to have is I'm going to drill into the cement and have it sticking out.
2. S: You going to have this sticking out, just the shaft of the bolt ?
3. C: Right.
4. S: Hmm. Interesting problem.
5. C: A hardened bolt would give me more...
6. S: Yeah, but it'll shear, they're more brittle. I don't know if you'd be any better off with a hardened.

Figure 4: Hardened Bolts

The customer is unaware of how the strength provided by the hardening process introduces an important potential weakness: brittleness.

Users can't combine or adapt tools for special uses

Though the combination in Figure 5 is simple, it does illustrate how tools can be combined in various ways both for similar and different reasons.

---

1. S: After deciding that a 3/16 inch wire is to be looped around a 1/2 inch bolt, which is mounted in cement.
   You want a small enough loop, put it between two washers.
2. C: Small enough loop.
3. S: Yeah.
4. C: Why between two washers, so it won't rub?
5. S: Yeah, so it won't slide off. Probably won't.

Figure 5: Combining simple tools

The customer doesn't know why the salesperson suggests a certain combination of tools, but ventures a guess. The salesperson allows the suggestion, but then states his reason.

To summarize, I've argued that McGuckin hardware deserves study since customers who go there face problems that are similar in nature to those faced by users of high functionality computer systems. The next section describes how observations of human-human cooperative problem solving sheds light on the shortcomings of current computer systems mentioned in the introduction.

3 Framework of this Study

This study takes place in the context of several years of work on cooperative problem solving systems research. For each of the three major challenges mentioned in the introduction, I briefly review an HCC system that addresses the challenge and show how interactions at McGuckin illustrate key concepts.

Situation vs. System Model

This distinction between a person's perception of the problem at hand, and a computer system's approach to solving a problem of that class serves to illustrate the importance that mental models

play in human-computer interaction. When a user approaches a computer system to solve a specific problem, there exist a "gulf of execution" [Norman1986]. This gulf is the difference between how a user perceives a specific problem, and how the system addresses problems of that type. The specific problem to be solved must be presented to the system in such a way that the system can be of help in solving it. Typically, this task of mapping the specific situation onto the model that the computer system assumes or supports is a difficult one. The challenge is to produce systems that assist in bridging this gap. Though there is value in struggling to state a problem clearly (if only it helps one to more clearly understand the problem), the major portion of the interaction with such a system should be in working the problem itself, not in discovering just how to present the problem to the system so that the "real work" can then begin.

The computer system should interact in such a way as to leverage the user's model of the problem and problem-solving expertise - not introduce a new model and force the user to re-map his situation to one that conforms to the system.

Direct manipulation interfaces are a step in this direction [Shneiderman1983]. Objects are depicted on-screen that model the real world . Behavior is attributed to these objects and a pointing device is used to trigger various semantics associated with the screen-object. Specifically, to click and hold the mouse key down, then to move the mouse, might mean that the indicated object is to be moved on the screen. Connecting two objects is perhaps accomplished by clicking on them in succession, etc. Instead of the system typing out a suggestion such as "There is too much friction between the wire and the bolt": that wire could be hi-lighted on the screen, or the depicted bolt could change color indicating a rise in temperature caused by the alleged friction. The point is that direct manipulation interfaces can be better than command-driven ones if only because they can be used to narrow the gap between system and situation model and use visual representations of familiar objects.

Interactions at McGuckin bear out that a direct-manipulation interface is a good way to communicate about problems. In the interaction shown in Figure 6 (A portion of which was introduced briefly in Figure 5 above), both the customer and salesperson holds or points directly to an item and uses it to illustrate and amplify the spoken dialogue.

---

1. C: I'm going to drill this into cement and I need a bolt to be able to put a cable on it. And I need something that will not bend, the strongest bolt. I'm going to drill a hole and bury it in there with that cement. I don't know, I don't know, I don't want it too big.
2. S: Going clear through?
3. C: No, just into the cement about this far.
   Holding hands about 4 inches apart.
4. S: Aha.
5. C: Do you have any with a hook on?
6. S: Not hardened.
7. C: So if I were going to hook something would this be the best thing? What I'm going to have is I'm going to drill into the cement and have it sticking out.
   Holding a bolt by the threads so that the shaft is visible for about an inch.
8. S: You going to have this sticking out, just the shaft of the bolt?
9. C: Right.
10. S: Hmm. Interesting problem.
11. C: A hardened bolt would give me more...
12. S: Yeah, but it'll shear, they're more brittle. I don't know if you'd be any better off with a hardened... You going to put a cable over it or a chain?
13. C: Cable, I guess it's going to take three sixteenths.
14. S: Three sixteenths.
15. C: I seem to have pretty good luck with three sixteenths and double.
16. S: Hmhm.
17. C: So I'm talking a thousand pounds.
18. S: Okay well this should do it, should be plenty strong.
19. C: This what you would, rather than a hook?

20. S: Hmhm.
21. C: Because I did this once before and I got a hook and now the hook is bent.
22. S: Yeah, well the next one's going to do the same thing because we don't have any hardened ones.
23. C: So this one's going to be the best thing?
24. S: Yeah.
25. C: and then
26. S: You want a small enough loop; put it between two washers.
27. C: Small enough loop
28. S: Yeah
29. C: Why between two washers, so it won't rub?
Holding hands vertically, side by side.
30. S: Yeah, so it won't slide off. Probably won't.
Sliding two fingers along the shaft of the bolt and "catching" on the head.
31. C: So this is what you would, so this'd be strong enough. I could go one bigger but then I'd...
Picking up the next size bolt, "weighing" it in his hand.
32. S: That's a little bigger.
33. C: Would you go one bigger? Would that be...
34. S: To a cement wall?
35. C: Cement wall foundation on the house
36. S: Poured cement?
37. C: Yeah, think that's enough? Or should...
38. S: I don't know. As big as I thought I could.
39. C: How to put around those hooks? Going to go around that about what size?
40. S: Probably half, overkill is better than not enough.
41. C: I have to drill - so half inch drill?
42. S: Yeah, half inch masonry drill
43. C: I had so far one of them break off and then I had this bolt bend like this.
Moving the head of the bolt in a horizontal arc.
44. S: Not good luck, yet huh?
45. C: So a half inch masonry drill, do I want it longer than that?
Holding the bolt by the head and pointing to the beginning of the threads.
46. S: I don't think longer than that .
47. C: Six inches. I could go in more.
48. S: Don't want to leave any more exposed than you have to.

Figure 6: Direct Manipulation

Both people manipulate the objects themselves to illustrate ideas and questions.

A project which has paid particular attention to issues of direct manipulation is Framer [Lemke1989]. It is a design environment in which users build user interfaces intended for the Symbolics Lisp Machine. Framer has taken direct manipulation a step further by supporting what is called Human Problem-Domain Communication. The idea is that people want to work within the "vocabulary" that is suitable to the problem. If the user wants to move a display pane up in a tiled window layout, he shouldn't be forced to communicate at the programming language level; rather the system should display and represent the mock display pane in a way that resembles the finished product. This communicates to the user in the "language" of the problem domain. Direct Manipulation of this sort still falls short in an important way: though users might be able to easily create an artifact, that artifact is in no way guaranteed to work well [Budge 1983]. Framer addresses this problem by adding a critiquing component which finds sub optimal designs and thus helps users build artifacts that function well.

Figure 7 illustrates that in large, complex information stores, it is important to communicate using terms that are familiar to the individual user.

1. C: Do you have any oils?
2. S: Oils?
3. C: Oil paints.
4. S: Yeah.
   Pointing to oil paints in the next isle.

Figure 7: Human Problem Domain Communication

The salesperson knew where oil paints were kept, but was unfamiliar with the term "oils."

Mixed Initiative Dialogue

Most expert systems interfaces have been built on a communication model described as the "consultation model." The idea is that the user enters a problem specification after which the system takes control of the interaction and asks questions or assigns tasks until it reaches a solution. This is not how people naturally interact. It is important that the system allow the user to volunteer unasked-for information and takes that information into account in subsequent interactions. The HCC system which addressed this as the key issue is System's Assistant [FischerStevens1987]. The groundwork for this system was laid in the Rebooter system. The task for which that knowledge-based system had been constructed was to reboot a Pyramid 90x computer after it had crashed. A crucial shortcoming of the Rebooter system was that frequently a user had information which the system should have been able to use, but for which the system was not currently asking: the interaction was entirely system controlled. The answer to this shortcoming was to allow a user to volunteer information at any time, to make the interaction less system-driven. Interactions at McGuckin hardware confirm that cooperative systems are based on interaction styles which allow either participant to guide the discussion. In human-human interactions people frequently and easily switch speaking and listening roles. Switching roles appears to be a prerequisite to successful interaction, rather than a hindrance or exception. One of the following interactions contains an expanded version of Figure 1.

However, many computer systems in general, and expert systems specifically, are constructed in a way that discourages or prevents the user switching from listening to speaking at the user's discretion. There are times when a user could save time if only the system would allow partial or incomplete input before beginning a 20-question type dialogue. At other times, a user may remember something useful at a time when the system is expecting an answer to a specific question. To force the person to wait until the current situation is over before any information can be volunteered is unnatural and goes against the grain of typical human interaction.

Incremental Specifications

The traditional waterfall model of software development places emphasis on identifying precisely and exhaustively the requirements of the system. This assumes that it is even possible, let alone advisable, to specify up front anything of any complexity. The waterfall model assumes that people can and will specify up front exactly what they need. Interactions at McGuckin hardware illustrate that a specification is less of a noun and more of a verb: it is an active process and at best approaches being finished without ever reaching that ideal state. At some point a satisficing solution will typically be reached, but the purchase of an item may not mean that the specifications are even yet complete. Figure 9 shows the specification taking place in more gradual increments.

Notice that the specification sometimes evolves into an entirely different problem altogether. In Figure 10, the salesperson is describing a story in which he helped the customer rephrase the problem from "adding heat" to "retaining heat." The HCC system which addressed the issue of incremental specifications is Helgon [FischerNieper-Lemke1989].

1. C: I'm looking for a small fastener maybe one sixteenth.
2. S: Okay. Plastic? Metal?
3. C: Well, what I've got is to fasten a sign on to a square pole. I've got a hole in the top and it fits fine and I got to get one on the bottom.
   Ignores the question and volunteers extra information.
4. S: Poles got holes in it?
5. C: Yeah, I had a 1/8 bolt but it's too big. Need something smaller than that.
   Follows the answer with additional information.
6. C: We've got a car rack for a van and it was put on with sheet-metal screws.
7. S: Right.
8. C: And they aren't holding very tightly. We'd like to get it a little bit more secure. I'm not sure what size of screws. Getting ideas right now. And...
9. S: Going into the roof?
10. C: Going into the roof, right. They screwed it in, and actually it's been replaced once. So they had to take them out and put them back in.
    Adds more problem description.

Figure 8: Examples of Information Volunteering

Information volunteering occurs quite frequently. Participants in the discussion accept unasked-for information readily and shape their subsequent dialogue by it.

1. C: I need a cover for a barbecue
2. S: Okay... what have we got here... chaise, chair barbecue grill cover... Does that look kind of like what you got? Similar? No?
   Pointing to a series of grills set up.
3. C: No
4. S: Take any measurements?
5. C: No
6. S: That's a good guess there.
   Pointing to a specific grill.
7. C: It's a double burner one
8. S: 52 inches that's the total length it'll cover.
9. C: Yeah. I know its not that big at all.
10. S: You saying about 18 by 18. Well, this is 27, it'll cover up to... here.
    Using measuring tape and pointing.
11. C: I need two.
12. S: A couple... in that brand, that's all I have. Here are these Weber ones, thicker material and all that. Here's some smaller ones.
13. C: I'll take this one
14. S: We'll be getting more of these pretty soon.
15. C: You'll have them by Christmas?
16. S: Hopefully Thursday.

Figure 9: Problem Specification is a Process

Specifications are given incrementally. 1) a cover for a barbecue 2) dissimilar to certain examples 3) double burner 4) not that big 5) I need two 6) by Christmas.

The domain for which it was implemented is information retrieval and the system has a database of bibliographic entries. Helgon is based on the Rabbit system, which introduced Retrieval by

Reformulation [Tou etal. 1982], but extends that approach by incorporating multiple specification techniques. There is more than one way of incrementally specifying the criteria that applies to a search. The importance of multiple specification techniques, in addition to incremental ones, is born out by customers at McGuckin.

To summarize, interactions at McGuckin hardware confirmed the importance of the various approaches to short-comings of current computer systems. During the study, several other issues also arose and these are described in the next section.

---

1. S: A guy wanted to get a couple of heaters to insulate a insulate a downstairs hallway... Picking his brain, "What are you doing? What are you trying to heat? Is it insulated? How tall are the ceilings?" We figured out that these two [heaters] would work. Then he told me that the reason it gets so cold is that right at the end of the hallway is where the stairs are and the stairs just go up to this great big cathedral ceiling. Well maybe the problem isn't that you're not getting enough heat downstairs, may your problem is that you're not keeping the heat downstairs. "Do you have a door across the stairs?" "No." "Well that's your problem." He goes "Yeah! Well I didn't really want to put a door there." "Well how about a curtain or a hollow-core door?" It was like wow, that is his problem. I said two things. You can put a ceiling fan and blow the hot air back down, or cover it up with some kind of door.

Figure 10: Redefining the Problem

The salesperson suggests solving a different problem than the initial one.

In search of beaded chain

1. C: Four foot of this?
   Holding and pointing to six inches of beaded chain.
2. S: Beaded chain, beaded chain, thirty-five cents a foot
   Notice here also Information Volunteering.

In search of an item pictured in a catalog

1. C: You have that?
   Pointing to an item in a catalog the customer has brought with her.

Figure 11: Multiple Specification Techniques

Customers use multiple means of communicating specifications: example items, catalog pictures.

---

4 Other Issues

This section describes the following:

- Natural Communication: how people use verbal and non verbal resources to communicate.
- Managing Trouble: how communicative troubles seem to be the rule and not the exception.
- Delegating: how the importance of the "situation talking back" influences the extent to which one can ever delegate design decisions.
- User Models: how salespeople learn to anticipate and structure their interaction more efficiently and effectively.

- Applicability Conditions: how people are able to generate creative solutions by knowing what attributes of an item apply in various circumstances.
- Sales agents: how roamers and isle workers combine to make a cooperative problem solving system.

Natural Communication

A large amount of effort within research in Artificial Intelligence and User Interfaces has focused on building Natural Language interfaces. The assumption has been that the closer we get to natural language, the better the interface will be. Surprisingly, in most interactions observed at McGuckin, the role of natural language was not central: rather, people chose many communication channels and non-verbal resources to communicate in what we call Natural Communication. That research in natural language is important is not disputed. The point is that a system which can parse written English sentences is not sufficient: ideally we should support all the interaction styles observed, both verbal and non-verbal. The work on Human Problem-Domain Communication is important in this light, since the point is not so much on parsing sentences, but rather on representing abstractions with which people naturally interact.

Managing Trouble

If we consider McGuckin to be a cooperative system, then it is surprising just how often communicative troubles arise. At first one might assume that a cooperative system would be characterized by flawless, misunderstandingless, communication. But breakdowns appear to be the rule and not the exception. The important thing to notice is that tremendous knowledge is constantly brought to bear in order to notice breakdowns and repair them. Our goal need not be to build an environment in which breakdowns never happen, but rather that as these communicative troubles naturally arise, the system have the resources to recognize and repair them. Suchman writes, "The problem is not that communicative trouble arises that does not arise in human-to-human communication, but rather that when these inevitable troubles do arise, there are not the same resources available for their detection and repair." [Suchman1987, p185] How troubles are detected and repaired also depends on the communication paradigm in use. Consider the relationship between dialogue based on the model of Advising and that of Critiquing.

Comparing Cooperative Paradigms

	Critiquing	Advising
Breakdowns	Happen	Are Avoided
Response	People act	People need not act
Result	People learn to elicit Tacit knowledge	People don't build the skills to creatively handle breakdowns

The communication paradigm affects what individuals are able to learn. Customers and sales agents both use critiquing as one of their interaction styles. The person who owns the problem is allowed freedom to explore alternatives and when sub optimal features are noticed, or the person reaches a stand-still, then the critiquing agent is able to provide additional information which helps the owner of the problem continue. Advising takes a more active and controlling role by allowing a lesser degree of freedom to the owner of the problem. An advisor usually proactively avoids false starts and dead ends, freeing the problem owner from the attendant frustrations, but disallowing the learning that takes place in detecting and repairing design breakdowns.

Delegating

The issue of delegation was alluded to in the discussion on Incremental Problem Specification and deserves more discussion. In the traditional systems development life-cycle, a user would communicate requirements to a systems analyst, who would then communicate to programmers, who in turn would communicate with systems experts and the programmers would then write the system. One way to view this is that the user "delegated" the design task to a person, who then delegated parts to others, who in turn delegated parts and performed some of the work. If users typically had all requirements perfectly in mind and nothing ever got lost in miscommunication, this plan actually might work. But recent work on situated action has indicated that no users ever know the requirements specification - that the requirements themselves are a moving target [Lave1988, Suchman1987]. Defining the problem is the problem. Ideally, a user could interact with the product environment itself and incrementally define the system, changing the specifications as time went on and it became apparent what certain design decisions implied. For this to become a reality, this new design environment in which users write their own systems must do what all the previous delegation accomplished - namely help the user define the problem and design the solution at the same time. So the "delegation principle" states: A cooperative system must have the resources that were previously represented by the people to whom tasks were delegated. Delegating is not a bad thing, since it allows the user to more clearly identify and compartmentalize the problem. The point is that in order to remove the people that have traditionally come between the user and the system, these new cooperative systems must accomplish the same roles that the people "in the middle" previously accomplished. There are several reasons to delegate any one task. The reasons we focus on here involve lack of expertise, the inability to do what one wants done. Traditionally users lacked the expertise to program systems themselves and were thus forced to delegate that task to experts. In this study the concern is not how delegation affects good use of time, but how it affects defining and solving problems. Schoen has recently stressed the importance of the "situation talking back," the design environment allowing the designer to reflect on the design, to engage in a conversation with the emergent design and so discover crucial attributes that need to be addressed [Schoen1983]. Herein lies the crucial difficulty with delegating anything of complexity: the designer (the person with the real problem) delegates the design to people who then engage in "reflective action," i.e. the designer interacts only indirectly with the emergent design and so is not able to foresee implications that certain specifications and assumptions are having on the final solution. He is freed from the tedium of working out intermediate solutions and compromises, but this very freedom prohibits him from "seeing" the often subtle consequences of intermediate tradeoffs. The work on situated action referred to above has pointed out how important it is that we study people not in fake laboratory situations, but in normal, everyday situations.

To study the delegation principle in action, we invited people who were already on their way to McGuckin to state the problem specification to another person, then to proceed as planned. They were solving their own problems. Having people state the problem to another person, attempting to provide that person with enough information to solve the problem on their own, did not lead to complete problem specifications. Though some omissions seem trivial, and easily resolvable, they nevertheless serve to illustrate how people really solve problems: namely incrementally, while the specification and the solution evolve in parallel. The following specification mentions most of what later turned out to be important except the price range of a potential solution.

Later on, when this customer went to McGuckin, he found a specialized tool that inserted threaded pop-rivets. This tool was ideal for this purpose, but it cost $140, too much, the customer decided, and went on to discover a $3 solution. In deciding that the tool cost too much, he had access to knowledge that was not reflected in the specification. A more subtle omission was illustrated by a customer who described a problem in great detail but who later discovered that he had misunderstood a crucial item. He thus illustrated the delegation principle: it turned out that he himself was not the end-user of the product in question. He had had the problem delegated to him, and had he, in turn, delegated it to another person, that person would have been unable to catch the problem.

1. C: We've got a rack on our car, or van, and people that mounted it just mounted it with screws
2. I: Sheet metal screws
3. C: And it had to be replaced once. So those are kind of loose. Now I'd like to be able to get it tighter. Now one, partial solution domain side: It would be nice if it was something you could, a nice rigid thread. And I wonder if they make something like these pop-rivets. That once you popped them in, they have a thread in them.
4. I: Umhm.
5. C: So that's an example of a solution that might work. Because I could pop that into the hole that's there and have a thread in it. It'd be a nice solid thread. That would be a lot better attachment. It seems like I've seen something like that.
6. I: If you were to send me to fix that, what would be the guidelines for you?
7. C: Well, the main thing is that I can get that thing attached in a more solid manner.
8. I: Okay.
9. C: The screws right now wobble loose. You know from vibration. They're not really holding it that tight. So that's the face requirement.

Figure 12: Multiple Specification Techniques

Customers use multiple means of communicating specifications: example items, catalog pictures. A specification which omits one key issue: how much the solution may cost.

The misunderstanding came to light as he was using pieces of wood to simulate parts of the device that was to be built. The situation "talked back" and alerted his partner (the person with the real problem) to the misunderstanding. Had his partner not been there, the event would have gone unnoticed. The point here is simply that as more tasks get delegated, the misconceptions can increase and will get caught much later than they would have if the person with the problem had been able to directly participate in designing a solution.

User Models

Since the number of people who visit McGuckin Hardware is large, you would not expect salespeople to form mental models of many individual people and retain those over time. Yet salespeople do use a variety of information to help them serve customers. For instance, a salesperson must decide whether to grant a specific request, or question it and suggest possible alternatives or query further about the intended use. The following segment from an interview with a salesperson shows how he makes decisions of this nature. When asked how they decide what terms and analogies to use, one salesperson replied:

> If somebody really has it narrowed down to a specific thing, I almost never question it or if I did at all, it would be to open the drawer of the specific thing they asked for or to have it in my hand and say, "Will this do the job for you?" You can show the guy exactly what he asked for and it happens a lot of times someone asks me for 3/4 inch screws and you open the drawer and immediately they're looking at 1 1/4, because they don't really have any concept in their mind. Sometimes by the clothes they've got on... but some of them... don't know the names for some of the things that are fairly common. Some of them are sent in by other people to pick things up. They don't know what it's for or how they're going to use it. And sometimes people will take 2 or 3 sizes when they're not sure.

and another replied:

> Sometimes I'm making a decision for the person just by looking at who it is. If they're buying it for somebody else. If they just need... especially around campus you got a lot of students... A lot has to do with seasons. We can't always read their minds as to what they're looking for. A lot of times dress or just appearance can give some things away.

Applicability Conditions

The dialogue in Figure 3 above was used to illustrate that customers do not always know when to use a certain item. In that dialogue the salesperson was able to apply the crucial features of the specification to his knowledge of the store and find "food scales." Computer systems that can do this will need to have items indexed by use as well as features. Perhaps the above item could be found by a query such as: "Find the items that can weigh things, have a platform, and take up less than 2 square feet of space." The features of items will also need to be worded with words that come from the various problem-domains and not words that are chosen from the perspective of people who manage inventories. Creating systems that have this sort of capability would represent a sizable step towards the goal of building truly cooperative knowledge-based systems. Knowing when certain constraints are crucial takes knowledge that is similar to distinguishing between good analogies and bad ones. The fit is not a yes/no, but rather lies on a scale from unacceptable through barely acceptable on to highly acceptable.

Sales Agents: The key to Knowledge-based Cooperation

Sales agents are the interface, the intelligent agent that makes the system work by:

- mapping a user's problem over to the system's solution, narrowing the gap between the situation and the system model;
- using appropriate communication techniques, allowing the owner of the problem to explore the solution space and critiquing sub optimal designs;
- helping people to incrementally refine their specifications: they are the crucial part of the "design environment" that McGuckin is.

Though they spend much of their time answering "simple queries," such as seen in Figure 13, the task they face of mastering the information space is a difficult one.

---

1. C: I need a little piece of black Velcro
2. S: Isle nine to your left

Figure 13: Simple Queries

A frequent task: directing people to items in the store.

The store is so large that a salesperson is not expected to be able to master it all. In fact, certain salespeople have a special responsibility and are called "roamers." These salespeople are very familiar with at least one department, but in addition to that, roam the whole store and provide a global view of the store to customers. They analyze requests and direct customers to appropriate locations in the store. Roamers require a tremendous amount of knowledge and experience. The roamer that spoke to about these issues was the "rookie," and he had been working at McGuckin for eleven years. What kind of training program can prepare a newly hired salesperson to learn a system that is so complex that the junior roamer requires eleven years of service? None. McGuckin has found that the best training program is:

> a name tag, a measuring tape and a green apron. Get out there and start helping people. Hopefully people have some kind of experience... We must be doing something right. We have no training program at all.

Strong evidence that learning really is in doing [Lave1988]. This only works when people learn from breakdowns, as argued in Figure 14.

1. S: Propane used to scare me. People would come with parts from their camper to hook up a propane refrigerator and a propane stove. I consider myself an expert at that stuff. I just mastered it over the years. I'd go get Glenn, in the beginning, Glenn was getting like, "Hey, you're going to have to learn this." I'm still a little hesitant with electrical things, too. Accessories, no problem. You want to do some technical rewiring I'll go get John and pick his brain and hope I don't have to ask him next time.

Figure 14: Learning over Time

The system works because sales agents adapt and learn over time.

5 Summary

McGuckin Hardware has overcome several inherent difficulties of High Functionality Systems. They key ingredient is the combination of roamer and salesperson. One to provide the global view, the other to provide the detailed domain specific expertise. Computing systems that attempt to approach this level of cooperative problem solving system will need to embody both domain specific and more general knowledge about the world, and be built on principles of communication illustrated above.

Acknowledgments

The idea for this study originated with Gerhard Fischer. A special thanks to McGuckin Hardware for allowing us to observe customer-salesperson interactions and to the anonymous customers who participated in this study. Thanks also to the members of the HCC group for helping execute the study and analyze the results, especially Thomas Mastaglio, Hal Eden, David Redmiles, Curt Stevens, and Tammy Sumner. This research was supported in part by grants from the Colorado Institute of Artificial Intelligence (CIAI) and the National Science Foundation grant CDA 8420944.

References

[Budge 1983]
B. Budge, *Pinball Construction Set (Computer Program)*, Electronic Arts, San Mateo, CA, 1983.

[FischerKintsch1986]
G. Fischer, W. Kintsch: *Theories, Methods and Tools for the Design of User-Centered Systems*, Technical Report, Department of Computer Science, University of Colorado, Boulder, CO, 1986.

[FischerNieper-Lemke1989]
G. Fischer, H. Nieper-Lemke: *HELGON: Extending the Retrieval by Reformulation Paradigm*, Human Factors in Computing Systems, CHI'89 Conference Proceedings (Austin, TX), ACM, New York, 1989.

[FischerStevens1987]
G. Fischer, C. Stevens: *Volunteering Information -- Enhancing the Communication Capabilities of Knowledge-Based Systems,* H.-J. Bullinger, B. Shackel, Proceedings of INTERACT'87, 2nd IFIP Conference on Human-Computer Interaction (Stuttgart, FRG), North-Holland, Amsterdam, 1987.

[Lave1988]
J. Lave: *Cognition in Pracice*, Cambridge University Press, Cambridge, UK, 1988.

[Lemke1989]
A.C. Lemke: *Design Environments for High-Functionality Computer Systems*, Department of Computer Science, University of Colorado, Boulder, CO, 1989.

[Norman1986]
D.A. Norman: *Cognitive Engineering* in User Centered System Design, New Perspectives on Human-Computer Interaction, eds D.A. Norman, S.W. Draper, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986, pp. 31-62.

[Schoen1983]
D.A. Schoen: *The Reflective Practitioner*, Basic Books, New York, NY, 1983.

[Shneiderman1983]
B. Shneiderman: *Direct Manipulation: A Step Beyond Programming Languages*, IEEE Computer", Vol. 16, Num. 8, August, 1983, pp. 57-69.

[Suchman1987]
L.A. Suchman: *Plans and Situated Actions*, Cambridge University Press, Cambridge, UK, 1987.

[Tou etal. 1982]
F.N. Tou, M.D. Williams, R.E. Fikes, A. Henderson, T.W. Malone: *RABBIT: An Intelligent Database Assistant*, Proceedings of AAAI-82, Second National Conference on Artificial Intelligence (Pittsburgh, PA), Morgan Kaufmann, Los Altos, CA, August, 1982, pp.314-318.

[Williams1984]
M.D. Williams: *What makes RABBIT run?*, International Journal of Man-Machine Studies, Vol. 21, 1984, pp. 333-352.