INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700  800/521-0600
RICE UNIVERSITY

TROUBLESHOOTING BY COMPUTER ADVISORS: A DESCRIPTIVE FIELD STUDY

by

JAN C. PANERO

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE MASTER OF ARTS

APPROVED, THESIS COMMITTEE:

David M. Lane, Director
Associate Professor of Psychology

Robert Dipboye
Professor of Psychology

Kathleen O'Conno
Assistant Professor of Psychology

Houston, Texas

May, 1997
ABSTRACT

Troubleshooting by Computer Advisors: A Descriptive Field Study

by

JAN C. PANERO

Telephone conversations between software technical support advisors and their clients were recorded and analyzed. The roles the advisors took in the conversations influenced how much control each conversant had and the kind of contributions they made. When the advisors took the role of solving a problem, they had more control and asked more questions than in other roles. The conversations where the advisors acted as problem solvers were analyzed qualitatively in light of the problem-solving theories such as information processing theory, Gestalt theory, and schema/frame theory. Most technical support problem solving was explainable using Gestalt and frame theory, but some behavior was displayed that was not predicted by these theories. A model describing the prototypical technical support problem-solving case is presented, along with descriptive findings about flexible behavior in non-prototypical cases.
ACKNOWLEDGEMENTS

This project would not have been possible without the generous support of the technical support agents who gave me their time and let me watch over their shoulders as they performed what can be a stressful job. Special thanks to all of them, and to the manager who orchestrated my visits and was extremely generous with his time.

I would also like to thank my research assistants, Tonyamas Charmsangavej and William Sean Ritzie who gave up hours of their summer to tirelessly code and transcribe audio tapes. I appreciate their hard and careful work.

Thank you to my parents and to Page Benway, all of whom supported me through this long project. I couldn't have done it without you.

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Terminology</td>
<td>1</td>
</tr>
<tr>
<td>Background</td>
<td>2</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>3</td>
</tr>
<tr>
<td>Context-free problems</td>
<td>3</td>
</tr>
<tr>
<td>Frame Heuristics</td>
<td>5</td>
</tr>
<tr>
<td>General Heuristics</td>
<td>7</td>
</tr>
<tr>
<td>Avoiding Einstellung: The &quot;Simplest-First&quot; Heuristic</td>
<td>9</td>
</tr>
<tr>
<td>Social Influence on Probability Judgments</td>
<td>13</td>
</tr>
<tr>
<td>Social Interaction</td>
<td>14</td>
</tr>
<tr>
<td>Stating the Goal</td>
<td>15</td>
</tr>
<tr>
<td>Asking the Question</td>
<td>16</td>
</tr>
<tr>
<td>Negotiating a Solution</td>
<td>18</td>
</tr>
<tr>
<td>Client Participation: Verification Requests</td>
<td>19</td>
</tr>
<tr>
<td>Other Potential Technical Support Failures</td>
<td>20</td>
</tr>
<tr>
<td>The Red Herring Problem</td>
<td>20</td>
</tr>
<tr>
<td>Missing Simple Solutions</td>
<td>21</td>
</tr>
<tr>
<td>Expertise of the Client</td>
<td>23</td>
</tr>
<tr>
<td>Communicating the Solution</td>
<td>24</td>
</tr>
<tr>
<td>Problem Solving</td>
<td>25</td>
</tr>
<tr>
<td>Descriptive Field Study</td>
<td>34</td>
</tr>
<tr>
<td>Studies Using Related Methodologies</td>
<td>35</td>
</tr>
<tr>
<td>Method</td>
<td>39</td>
</tr>
<tr>
<td>Quantitative Analysis</td>
<td>40</td>
</tr>
<tr>
<td>Overall Categorization of the Queries</td>
<td>40</td>
</tr>
<tr>
<td>Advisor Role</td>
<td>41</td>
</tr>
</tbody>
</table>
Conclusion of Query .......................................................... 43
Content Coding in Previous Technical Support Research .......... 43
Development of the Coding Scheme ...................................... 44
Category Definitions: Client .............................................. 46
Category Definitions: Advisor ............................................ 49
Coding Procedure .......................................................... 49
Inter-coder Agreement ...................................................... 50
Method of Analysis ........................................................ 51
Qualitative Analysis ........................................................ 53
Results and Discussion ..................................................... 54
Quantitative Analysis ....................................................... 54
   Advisor Expertise ......................................................... 54
   Overall Categorization of Each Query .............................. 54
   Content Coding .......................................................... 57
      Concerns About Significance Testing .......................... 57
Comparison with Previous Research ................................... 58
   Balance of Control ..................................................... 58
   Client Participation .................................................... 60
Comparison to Previous Research: Conclusions .................... 64
Differences as a Function of Advisor Role ........................... 66
Advisor Role: Conclusions .............................................. 72
Qualitative Analysis ........................................................ 74
   Fitting Problem-Solving Theories to Technical Support ........ 75
Recognition of a Familiar Problem Based on Symptoms ............ 86
Working Directly from Surface Features: Use of System
   Knowledge ............................................................... 87
Extending the Models ..................................................... 88
Technical Support Strategies.........................................................95
Create More Surface Features.......................................................96
Eliminate High-Level Frames.........................................................96
Other Heuristics ............................................................................97
Social Influence.............................................................................98
Did Advisors Test Simple Solutions First?....................................103
Einstellung ......................................................................................105
Summary of the Qualitative Analysis ..............................................109
Fitting the Model to Non-Prototypical Cases...............................112
Potential Technical Support Failures.............................................113
General Conclusion.........................................................................115
References.......................................................................................118
Appendix A: Individual Differences Among Advisors.....................124
LIST OF TABLES

Table 1: Each query was categorized by the role the agent took in the interaction and the conclusion of the query. ..............................41

Table 2: A summary of each coding category used in the present study and its closest precedent in previous research. .........................45

Table 3: Number and relative frequency of queries for each advisor role that were resolved, indeterminate, or unresolved. .................56

Table 4: Comparison of client responses in previous research to responses in the present study..................................................61

Table 5: Types of verifications made by clients in previous research compared to those made in the present study............................62

Table 6: Verifications that are new ideas as a function of queries about sophisticated or less-sophisticated topics..................................64

Table 7: Relative frequency of each type of verification request as a function of the advisor role....................................................67

Table 8: Edited excerpt from a query. Information gathering and search.......................................................................................75

Table 9: Edited excerpt from a query. An involved information-gathering process.............................................................................78

Table 10: Edited excerpt from a query. Unexpected answer during information gathering......................................................................79

Table 11: Edited excerpt from a query. Advisor explains hypotheses, narrows hypothesis frame.........................................................82

Table 12: A hypothesis frame for a computer problem......................................................................................................................85

Table 13: Edited excerpt from a query. Immediate invocation of a familiar frame................................................................................87
Table 14: Edited excerpt from a query. Advisor moves through several frames.

Table 15: Edited example from a query. Advisor keeps two familiar frames in mind simultaneously.

Table 16: Edited example from a query. New information causes advisor to switch domains.

Table 17: Four edited excerpts from a query. Client provides hypotheses.

Table 18: Edited excerpt from a query. Client's hypotheses fit easily into the advisor's problem-solving process.

Table 19: Two edited excerpts from two different queries. Advisors check for simple hypotheses immediately.
LIST OF FIGURES

Figure 1: A context-free troubleshooting task..................................................4

Figure 2: Combination of information processing perspective and
schema-driven perspective of problem solving taken from
Gick (1986)...........................................................................................................29

Figure 3: Combination of information processing perspective and
Gestalt perspective of problem solving as described by
Ohlsson (1984b)..................................................................................................30

Figure 4: Model of how the symptom knowledge base and system
knowledge base contribute to the selection of frames.........................34

Figure 5: The distribution of the lengths of the calls.................................55

Figure 6: The distribution of the lengths of the structuring queries........56

Figure 7: Portion of description, verification, and question items
made by clients as a function of advisor role in the query..................69

Figure 8: Portion of instruction, question, and statement items made
by advisors as a function of their roles in the query.........................70

Figure 9: Portion of each speaker's utterances that are controlling, as
a function of agent role in the query..............................................................72

Figure 10: Govindaraj and Su's model of fault diagnosis........................86

Figure 11: A model of prototypical technical support problem solving........111

Figure 12: The use of the category "instruction" across roles by the
advisors............................................................................................................124

Figure 13: The use of the category "question" across roles by the
advisors............................................................................................................125

Figure 14: The use of the category "statement" across roles by the
advisors............................................................................................................125
Figure 15: The use of controlling utterances across roles by the advisors.
Introduction

Technical support is one of the fastest-growing occupations in the United States (Lohr, 1996) and yet little has been written about how computer advisors, or technical support agents, solve their clients' problems. In many ways, technical support interactions are like doctor-patient or mechanic-driver interactions. The client is an expert in his or her own computer use: in what he is trying to accomplish, and the symptoms of what went wrong. The advisor is an expert in the tool: how the hardware and software are supposed to work, and potential reasons that they might not. Since technical support is usually performed remotely over the telephone, the agent is not an independent troubleshooter and therefore must interact with the client to identify the problem space and goal. The advisor is the one who solves the problem, but is constrained by the information provided by the client, including the veracity of the details, the client's level of expertise, and the client's ability to carry out instructions as the advisor intends. This paper discusses (a) the heuristics that advisors use and how they are affected by the human-human interaction between themselves and clients, (b) how well existing theories of problem solving and troubleshooting explain what advisors do, and (c) the factors that make advisors adjust their tactics.

Terminology

Advisor: The technical support agent who provides help to the client.

Client: Customer, caller, advisee or user who is asking for help.

Query: What the client needs help with. This term is used instead of "question" because queries are not always posed in the form of a question. Also, a client and advisor may each ask many questions in the process of resolving the query. A query is a single-topic conversation between a client and an advisor.
The advisor is referred to as "she" and the client as "he" for ease of reading.

**Background**

Early research on the interactions that occur in technical support focused on what clients wanted from the technical support organization, their subjective evaluations of the help they got (Alty & Coombs, 1980; Lang, Auld & Lang, 1982; Lang, Lang & Auld, 1981), and the structure and control of the conversations (Coombs & Alty, 1980). Later studies described advisory interactions in the pursuit of the development of expert systems to provide automated interactive support to clients (Aaronson & Carroll, 1987a; Hill, 1993; Kidd, 1985; McKendree & Carroll, 1986; Pollack, 1985; Senjen & Austin, 1993). This research has not succeeded in its goal of providing knowledge for the development of expert systems. Hill (1993) concluded from his studies of advisor-client interactions that the giving and following of advice is so full of contingencies, presuppositions about clients, social contexts, and constant refinements of assumptions that creation of an intelligent advisory system would be highly impractical. Still, human advisory services have been found to be the most important source of guidance in computer use (Alty & Coombs, 1980) and so any improvement that can be made to them should be beneficial to clients and computer companies.

In order to help a client, an advisor must learn the client's problem and goal in the context of the client's work, formulate a solution acceptable to the client, and explain the solution so that the client can implement it. At the base of this interaction is the advisor's ability to find the problem – her troubleshooting experience and skill. Therefore, I begin with a review of the issues involved in individual troubleshooting and then continue with the influence of the social context of technical support on troubleshooting
heuristics. Finally, I discuss how technical support troubleshooting relates to more general problem-solving theories.

**Troubleshooting**

**Context-free problems**

Technical support of computer systems is a type of troubleshooting or fault diagnosis, on which there is a large literature (Bisant & Groninger, 1993, for a review; Morris & Rouse, 1985). Rouse (1978) introduced a context-free task for analyzing troubleshooting strategies. The problem space used in this and many follow-up studies was an abstract network of nodes. In a network task each node takes one or more inputs and operates by this rule: if all inputs are 1 and the node had not failed, the node produces the output "1", otherwise, it produces the output "0." Subjects are given a series of networks where the final outputs of the nodes would be a pattern of 1's and 0's. The task is to find the failed node. In Figure 1, all the nodes in the first four columns pass the value "1" to the right. Node 31 is the failed node, so it passes "0" to the right. Every node that comes after it (37, 38, 44, and so on) also passes 0. This results in the pattern of 1's and 0's on the right side. The subject's task is to identify node 31 as the failed node.
Rouse determined that the network problems could be diagnosed with a brute-force algorithm, tracing back from any 0 node, either testing nodes randomly or, with more cognitive effort, determining the faulty node without any tests. However, the optimal strategy was found to be the split-half algorithm. Using the split-half algorithm, the subject analyzes the pattern of outputs to select a test that eliminates half of the possible solutions. The faulty node is then relegated to one half of the remaining network and the algorithm is repeated until the faulty node is found. Rouse discovered, however, that human problem solvers tended to use strategies more similar to brute force than split-half, although eliminating time constraints and providing computer aids to help with bookkeeping tended to increase the optimality of the strategies. Rouse also found a confirmation bias in that subjects tended not to use information that indicated that a particular node was working.
Frame Heuristics

The network task is useful for examining some troubleshooting strategies. However, the context-free nature of the task limits the variety of strategies researchers can observe. Using richer tasks, several researchers have suggested that experienced troubleshooters use "frames" to organize their knowledge about the faulty system and determine a strategy of hypothesis testing. A frame is a data structure represented as a network of nodes and relations that organizes information about a stereotyped situation (Minsky, 1975).

The top level of the network is composed of constants – information that is always true. For example, a computer network administrator might have a "network problem" frame. When clients report problems such as being unable to print their documents, unable to send email, or similar errors, the administrator checks to see whether the problem fits the network error frame. The administrator would check for symptoms that are usually indicative of network problems. These symptoms are the constants at the top of the frame. Are several clients having problems, or is it just one person? Do the problems involve several network-related functions, or just printing?

The lower levels of the frame are made of many slots that can be filled with specific instances or data, and usually contain default information. The slots can also specify conditional rules for what can fill them. The assignments themselves can become subframes that are invoked in given conditions. A subframe might be email server problems. A conditional slot would contain a rule. For example, "If all clients are having email problems, the email server may be broken." The administrator would look to see if everyone is having email problems, or just some people. If it is only a few people, the email server slot is filled with "no." Similar conditional testing
can eliminate other possibilities and narrow the number of hypotheses. So, a troubleshooter chooses a particular frame given the current situation, and fills in each slot based on what is known and his or her current goals.

Govindaraj and Su (1988) identified five heuristics for choosing a frame or a potential cause of a symptom. These were:

1. Choose frames of higher perceived likelihood of being the cause, based on previous experience with how often a component fails or the degree of dependency between components.
2. Choose rules of higher specificity. For instance, subjects used combinations of information, choosing a different hypothesis for the cause of a symptom depending on the context.

If the above heuristics didn't work, subjects would choose from the next 3:

3. Search through subsystems or components physically close to the one that was hypothesized to have failed.
4. Choose the frame with the hypothesized symptoms that best matches the observed symptoms.
5. Choose the frame that contains subsystems or components that are related to the symptoms on the surface. This heuristic ignores structural, topographic, and functional knowledge.

Smith, Giffin, Rockwell, and Thomas (1986) looked at the process of filling the "causes" and "expectations" slots in fault diagnosis frame. They found that successful experts appeared to use knowledge structures and managed to overcome some common cognitive decision-making biases. For example, they did not have a confirmation bias. Given a set of symptoms, the subjects would create hypotheses, filling the "causes" slots of the frame. They would then attempt to reject each hypothesis by testing the "expectations" slot. In the computer network example given above, a causes slot would be
"email server down" and its associated expectation is "no one has email." As in other types of tasks (Mynatt, Doherty & Tweney, 1977), these subjects did not consider alternative hypotheses, but they did attempt to nullify their current hypothesis. It may be that, unlike in Rouse's context-free task, where a confirmation bias was observed, when troubleshooters were able to organize their knowledge into richer relations they were more likely to try to eliminate possibilities than to focus on proving a single hypothesis.

General Heuristics

Katz and Anderson (1988) looked at the process of locating bugs in computer programs. They identified three general strategies:

1. *Simple mapping* is the strategy of matching the program's behavior directly to the bug. For example, if a variable name is given in an error message, a strategy would be to look through the program for occurrences of that variable name. This strategy doesn't require a very deep comprehension of the program.

2. *Hand simulation* is the process of tracing the lines of the program, executing it as the computer would, and looking for discrepancies between expected and actual results.

3. *Causal reasoning* is the process of examining the output and determining logically, based on one's understanding of the program, what the error might be.

Katz and Anderson (1988) noted that their novice subjects did not consider some causes until after they had disproved their first set. Katz and Anderson reasoned that this might be a function of familiarity. For example, the actual cause of a bug might be the failure to close a parenthesis, a very simple syntactic error. However, the subjects in this study had not written very complex programs before, so such errors had been rare, and, therefore,
were not considered until late in the debugging session. The authors asserted that an experienced programmer who had made many parenthesis errors in the past might be likely to check that hypothesis first, regardless of whether or not the program output directly suggested that cause (a variation on the simple mapping heuristic). Despite its inefficient-sounding description, simple mapping may be a very efficient strategy when used by experienced debuggers who know which simple solutions to test.

This phenomenon of failing to consider simple hypotheses has been noted before in experiments using concept attainment tasks. In these tasks, a subject sees a series of pairs of cards that vary on a number of dimensions: left vs. right, circle vs. triangle, blue vs. green, large vs. small, etc. The subject chooses (randomly the first time) one card from each pair and is told "right" or "wrong," depending on the experimenter's predetermined rules. Levine (1971) found that if subjects sampled from the incorrect set of hypotheses, they would never solve the problem. For example, possible solutions could be "choose in the pattern of left, left, right, left, right," "always choose the green card" or "choose the circle if it is green or large, otherwise choose the triangle." Levine found that if a subject was sampling from the set of hypotheses of "some pattern of rights and lefts" but the actual solution was "always choose green," the subject might never solve the problem, even though the solution seems very obvious. In fact, subjects didn't seem to learn anything about this alternative hypothesis (Fingerman & Levine, 1974).

Levine (1974) developed this finding into the transfer hypothesis. The transfer hypothesis states that the universe of possible solutions is broken into domains of solution, such as, for the card selection task, simple solutions, sequence solutions, conjunctive solutions, and so on. When a subject solves a number of problems, he or she infers from the first
problems the domain from which the solution to the \( n + 1 \)st problem will come. The subject samples from that domain until it is exhausted before moving to another, just as Katz and Anderson noted in their debugging task. In the event that the chosen domain is infinite, the problem might never be solved. This failure to "see" the simple solution is a form of Einstellung, the classic Gestalt phenomenon of "psychic blindness" first described by Luchins (1942; Luchins & Luchins, 1959).

**Avoiding Einstellung: The "Simplest-First" Heuristic**

As Wilson (1991; 1994) noted from his experience as a technical support advisor, the expert troubleshooter has to learn to break out of the current domain and to try other solutions. Wilson suggested trying quick fixes as a first approach: check the cables, reboot the machine, etc. These solutions are very quick and cognitively simple. Mager (1982), another author speaking from extensive troubleshooting experience, suggested quick fixes as well. The idea is that even though a particular simple solution may not be the *most* likely cause of the trouble, it takes little time or energy to test, and thus introduce little cost to the procedure. If the simple solution is indeed correct, a great deal of time is saved from not testing more complicated hypotheses.

Stolurow, Bergum, Hodgson, and Silva (1955) noted that the "optimal" split-half heuristic is in fact suboptimal when the range of possible hypotheses varies in probability of correctness and in the time it takes to test them. Rouse found his network task to be solved optimally with the split-half heuristic, but in this task all tests took the same amount of time and there was no contextual experience that affected the probability that a particular node might fail more than another. Stolurow *et al.* determined that the best algorithm that accounted for probability and time "cost" was to tabulate all possible permutations of orders to test the components and to compute the
total average time each permutation should take. The optimal order was the one with the least average time. This is a rather cumbersome algorithm, however, so the researchers determined that a simple approximation was to test the components in increasing order of time divided by probability.

A hypothetical example is that if a client attempts to print a document and nothing is produced by the printer, this problem is caused by a broken link in the network 95% of the time. Therefore, someone having this problem may be tempted to trace the whole network, which takes 2 1/2 minutes (150 seconds). However, 5% of the time, the problem is caused by an unplugged printer, a possibility which takes 2 seconds to confirm or eliminate. The "unplugged" ratio is 40 (2 + .05), and the "network link" ratio is 158 (150 + .95), so the algorithm dictates that the troubleshooter should check first that the printer is plugged in. If the plug is the problem, the time to solution is 2 seconds. If the plug is the problem but the network is checked first, the time to solution is 152 seconds. The average time to find the correct solution across many incidents is 144.5 seconds when the simple solution is checked first, and 150.1 seconds when the complex solution is tested first.

Levine's work suggests that if complicated problems are the cause of errors most of the time, troubleshooters sample from a domain of complex solutions until it is exhausted, and only then switch to other domains, including the domain of simple solutions. Ashby (1988) asked subjects to find an error in one of five algebra problems. The error was equally likely to exist in any of the problems, but some problems took longer to check than the others. Eventually, most subjects learned the strategy of testing the simplest (quickest) problems first. However, the subjects were less likely to use this strategy when there was more than one error to find. Later researchers (Dammon, 1993; Dammon & Lane, 1993) found that if subjects were first:
"trained" on a series of single-fault algebra problems and came to use the simplest-first strategy, they were able to transfer that strategy to multiple-fault algebra problems as well as to single-fault problems in the more enriched domain of checkbook balancing.

However, the probability that an error will be found in a particular location is not usually equal across all possibilities, and so if a troubleshooter is to apply Stolurow et al.'s (1955) time + probability heuristic, he or she has to be able to judge the probability, as well as the testing time of a particular potential failure. In their original study, Stolurow et al. found that expert troubleshooters did not agree on probability or time ratings for the cause of various symptoms. In fact, it is a well-known phenomenon that people, even experts, are biased in judging probabilities (Tversky & Kahneman, 1974). Other researchers have found that people who generate lists of hypotheses of the cause of trouble tend to believe them to be fairly exhaustive (Mehle, 1982) and that even after new hypotheses are introduced, people don't adjust the probabilities of the alternative hypotheses to allow for the new idea (Van Wallendael, 1989). So, even if troubleshooters used Stolurow's simple heuristic, it would be difficult to detect, since it would be dependent on each individual's judgment of relative probabilities and time-to-test.

The problem hasn't been solved in medical diagnosis either. Clinical severity in medicine is analogous to time-to-test in technical troubleshooting. Severe diseases are analogous to simple solutions in that they have low probabilities, but if they are missed, the consequences are negative. In technical support, the consequence is wasted time, in medicine, of course, the consequences are much worse. However, medical text books are unclear on how to avoid missing rare but severe diseases, and it is not easy to know when to look for them (Weber, Bockenholt, Hilton & Wallace, 1993).
Suppose a technical support organization could measure carefully both the ultimate solutions to clients' problems on a single product over a period of time and the average time to test for each of these solutions. Then, advisors would have base-rate and time information available to them, and could apply Stolurow et al.'s (1955) algorithm. Would this increase the efficiency of the overall organization? Probably not, because the clients who need help come from a variety of populations, and the probability that a particular solution is correct may vary among the different subpopulations. It is likely that a novice computer client calling a technical support line has not checked for simple solutions: cables loose, software settings incorrect, devices unplugged, and so on. It is also possible that a novice may check for these possibilities and incorrectly conclude that all is in working order. An expert client is less likely to first, not to have checked for simple solutions, and second, to have checked and incorrectly dismissed them. So, an advisor speaking to clients who seem to be experts or novices might correctly judge the relative probabilities of the possible causes differently for each type of client.

In many cases, adjusting the base rates for subpopulations may be the right thing to do. Some researchers question Tversky and Kahneman's (1974) finding that using the availability heuristic is an error. Nisbett and Ross (1980) note that the practitioner is interested in characterizing individual cases rather than populations, and thus preconceptions may be appropriate to make judgments of probability when the data about the case are ambiguous. Einhorn and Hograth (1981) noted that there is no accepted way of defining the relevant population for a particular case. They gave the example of estimating an individual's propensity to heart disease. In this case, there is no way to be certain which population is most relevant: the age group of the
person? Geographical area? Einhorn and Smith concluded that there was no way of determining whether people attended to base rates in naturally occurring phenomenon because the relevant population was ambiguous.

Determining base rates for problem solutions for a product may not be useful to technical support agents. Overall ratings would have limited usefulness because the subpopulation membership of individual clients would vary widely. Perhaps experience really is the best way for agents to make accurate judgments. Weber, et al. (1993) found that doctors used probability effectively when they had previous experience with the particular problem. Memory of previous similar cases increased the likelihood that doctors would be sensitive to base-rate information.

Medical patients with known risk factors for rare diseases are more likely to have them, and doctors are less likely to miss them. For technical support, Levine's transfer hypothesis (1974) implies that when expert clients report a problem caused by a simple error, the advisor is unlikely to find it quickly, first sampling from the "most likely" domain of complex solutions. Conversely, advisors might take longer to begin testing complex hypotheses for problems presented by novice clients, because the advisors may first exhaust the set of simple solutions. The latter is less likely to cause problems than the former, since it shouldn't take much time to test the simple hypotheses. Also, Lane, McDaniel, Bleichfeld, and Rabinowitz (1976) found that troubleshooters move more easily from simple domains to complex ones than vice versa.

Social Influence on Probability Judgments

There is some evidence that troubleshooters vary their diagnoses depending on who's reporting the problem. In a study of initial psychiatric evaluations, it was found that clinicians tended to give more complex
diagnoses to men over women, and, to a lesser extent, to blacks over whites (Fabrega, Mezzich & Ulrich, 1989). Another study found that when an experimenter asked for street directions in Boston, he got longer, more detailed explanations when he gave explicit ("I'm from out of town") or implicit (southern accent) clues that he was a Boston "novice" (Kingsbury, 1968, as cited in Krauss & Fussell, 1991). Mehle (1982) studied subjects who were asked to generate hypotheses for causes of some symptoms of car trouble. In each case, the male subjects read scenarios of car trouble that supposedly came from a phone call from "your spouse." Although it was not the topic of the report, it is clear from a few of the quotations from protocols of the subjects that they were using the source information to help them decide on likely hypotheses. In one scenario, the car was difficult to start. One hypothesis began, "Let me see ... Flooded ... All the time; ... Like most of the girls do." In another scenario, the car stalled at stop signs. Another subject said, "It could be that the dumb wife does not know how to work the clutch...So, I think the clutch is a problem," and in fact, this subject eventually concluded that the clutch problem was the most likely cause (p 97).

Social Interaction

Early in a technical support interaction, the agent needs to infer the client's goal and question. Presumably, this occurs simultaneously with the agent's inference of the client's expertise and other attributes, such as level of emotional arousal. Subjectively, advisors report that there is "something about how they [clients] ask the question" that tells advisors whether the client knows what he or she is doing and whether he or she is angry or willing to interact patiently (P. Benway, personal communication, Feb. 29, 1996). This "something" in the client's initial question influences how the advisor directs the conversation when she has control. By phrasing their
questions in particular ways, clients communicate metamessages such as "I need a lot of explanation," "I'm upset and in a hurry so just solve my problem without too much chit-chat," "I know more than you do about this system so believe what I say and follow my suggestions," or "I think I know what I'm talking about but really don't." Demographic characteristics and competency (knowledge of terminology and procedures, for example) influence doctor-patient communication and doctor behavior (Fisher, 1983; Hooper, Comstock, Goodwin & Goodwin, 1982) in valid and invalid ways. For example, if a patient shows a lack of knowledge about what tests results mean, the doctor should give a more thorough explanation to that patient. On the other hand, attributes such as race should not influence whether the doctor provides opportunities for the patient to show competency, ask questions, or give background information. Similarly, client attributes can validly influence the advisor's behavior of adjusting probability judgments, using different levels of explanation, and following up on client suggestions.

**Stating the Goal**

A clue to clients' levels of expertise is whether they know what to ask. Miyake and Norman (1979) showed that people can only ask about topics they have a basic understanding of. However, the advisor must know the client's goal before addressing the problem. Hill (1993) found that the statement of goals was an implicit advice-seeking rule – clients always made sure that advisors knew their goals as best the clients could explain them. However, Wilson (1991) speaking not from research but from experience as an advisor, found that especially novices often initially asked questions that differed from their real goals. Pollack (1985) noted that since clients didn't always know exactly what information they needed to solve their problems, advisors needed to infer underlying goals. For example, a client was trying to use a
library database over a modem. Not knowing where to start, he called the library and asked, "What do I type to log in?" The advisor told him the command he needed to type on the log-in screen. This was the command a client types to actually log into the system, once the client has connected the computer by modem to the library system. The client had not connected to the system though. He was typing commands DOS, the operating system on his computer, which gave him an error message (J. Barney, personal communication, Jan. 15, 1996). The clients goal was to connect to the library system and then log in, but his question was about how to log in. Due to the client's misconception of the steps he would need to take, the advisor inferred the wrong goal. So, it may be that clients give their best efforts to provide their goals to their advisors, but advisors often have to infer true goals from bits of information given by the client. When the advisor infers the wrong goal, the interaction has to cycle back and forth between the solution stage and problem statement. In the log-in example, the client reported that he received an error message and this surprised the advisor. The advisor asked for more context – the exact error message. From this information, the advisor realized the client was at a different stage than she had inferred, and the advisor began the solution phase again, this time with the correct goal.

**Asking the Question**

The goal statement is wrapped up in the communication of the overall problem, context and background information that is needed to bring the advisor into the human-computer interaction. Coombs and Alty (1980) found that the quality of the query statement varied between clients. They noted that novice clients tended to give overly concrete or else long but imprecise descriptions to support their queries. Wilson (1991) also saw this as an area for
a potential communication breakdown. He noted that understanding the question fully is a prerequisite to being able to solve the problem efficiently. Advisors must be careful to pay full attention to their clients' descriptions, listen actively, and to be aware of the context surrounding the problem in order to understand the question. Metamessages in the question statement can give the advisor an orientation from which to approach the interaction. The advisor may receive clues to the client's level of knowledge from terminology that the caller uses (Hopper, Handy Bosma & Ward, 1992), explicit statements by the client ("this is the first time I've touched this machine"), or by evidence of inaccurate mental models of the system. Tannen and Wallat (1986; 1987) gave examples in the medical domain of inaccurate models by a patient's mother. Communication difficulties arose because the doctor's and mother's models didn't match. The patient, a child, had cerebral palsy (CP). When the doctor asked the mother about the child's general health, he meant problems besides the CP, such as bronchitis or ear infections. The mother indicated that the patient has no such problems but was not in good "general health" because she had motor deficiencies. The doctor's model of general health was different from the mother's and so the doctor got different answers to his questions than he expected, and the mother continued to show concern over the child's general health. In technical support, when a client and advisor have mismatched models, this may influence the advisor to ignore certain "out of model" facts provided by the client, or to give more explanation. The existence of such facts provides information to the advisor, but also creates the need for the advisor to decide which misunderstandings to address and which to ignore. Over-explaining can bog the interaction down, while ignoring too much can lead to communication problems later in the interaction.
Negotiating a Solution

Kidd (1985) described a complex negotiation between clients and advisors toward a final solution. In his study, conversations that were completely dominated by the advisor were rare and generally unsuccessful. He found that clients participated by constraining the solution space by saying that the solution must be "cheaper than X" or "must not affect Y." The client would then negotiate the solution by checking that the constraints were satisfied, rejecting solutions already tried, proposing their own solutions, and asking for further information about the solution.

On the contrary, Coombs and Alty (1980) found that advisors were mostly occupied with finding the solution rather than communicating their reasoning to clients. Advisors might have asked for pieces of background information but they didn't explain the solution process. Advisors stated that they believed clients were responding during this episode, but clients said they did not understand what the advisor was doing. The reason for this discrepancy was that clients said encouraging things in order to keep the advisor working on their problem. The client must make genuine contributions in order to stay involved in the interaction or he might be left behind as the advisor focuses on the cognitive task of solving the problem. The need for clients to stay involved in the interaction while solving the problem is clear. Whether clients actually do is less clear. Research on technical support is mixed on how much clients contribute. This may depend on how confident a client is, how skeptical, and how talkative (Wilson, 1991). It may also be influenced by the advisors: how empathic they sound, how many openings they leave for the client to participate, and so on. In doctor-patient communication, demographic variables seem to influence how much
control of the conversation the doctor gives to the patient (Fisher, 1983), and such variables may function similarly in technical support.

**Client Participation: Verification Requests**

When clients contribute in the solution phase, they tend to make verification requests (Aaronson & Carroll, 1987a). A verification request is a question asked by the client that provides a partial answer. The client asks (usually implicitly) whether his ideas are correct. Aaronson and Carroll asserted that verification requests were useful for a client to verify his understanding, and to get the advisor to address the issues from the client's point of view. This could help the client solicit the correct level and type of explanation from the advisor. Verification requests could also allow the client to introduce ideas and take some control without compromising the roles of the participants. Verification requests may help in improving the client's memory for the solution through rehearsal, and allow the client to pace the conversation so that he can process the information provided. Aaronson and Carroll found that most of the verification requests from expert clients contained new ideas that had not been introduced previously in the conversation. New ideas were contained in a minority (33%) of the verification requests from novice clients, indicating that novices more than experts might have been using these requests as memory aids or to pace the interaction. Experts might have used the requests to assert some control, but both groups of clients probably used verification requests to convey their level of understanding and to convey their points of view.

Tannen and Wallat (1987) showed that patient misconceptions can become clear through mid-exam questions and comment. The mother of the child with CP suggested connections that could not exist between the disease and symptoms. For example, she asked whether a rash behind the child's
right ear was connected with the CP since both were on the right side of the body. However, the same-side relationship was false, and the doctor knew the rash and CP were not connected. However, the doctor shifted his attention to the rash and away from what he was doing before the mother mentioned it. This question by the mother provided the doctor further information about the mother's understanding of CP, but if it had been different, it might also have suggested a hypothesis to the doctor that he might not otherwise have thought of. It is possible that technical support advisors could pay differential attention to such verification requests from clients depending on their displayed level of expertise. Following up a verification request requires a shifting of frames for the advisor, so she may be less likely to oblige the client if the request shows a misunderstanding. In some cases, this could be a mistake – the new idea brought by the client could be more valid than it seems, especially if it is offered in a mitigated fashion.

Other Potential Technical Support Failures

The Red Herring Problem

Wilson (1991) described a rule he called "beware the red herring," which is a risk when there is an obvious cause to the problem. For example, if the client mentioned there was a large storm yesterday, the failure of a printer could be due to a power surge. However, Wilson warned advisors not to focus exclusively on that cause because the problem could be a simple loose cable or a client error. In one case, an organization was having trouble with a virus that caused the characters on the monitor to "melt" to the bottom of the screen. One caller reported that her "screen saver was falling off the screen." An advisor was sent to her office to remove the virus, only to discover that a glare reduction screen that was attached to the monitor was slipping. The problem was solved with a piece of tape (J. Shaw, personal communication,
December 5, 1995). This particular problem stemmed from a terminology error (Hopper et al., 1992), but it was compounded by the virus red herring. If the entire interaction had taken place over the telephone, it might have taken a very long time to solve. Examples like this abound as funny stories about technical support, and they may stem from the common use of the availability heuristic (Tversky & Kahneman, 1974), in that people tend to inflate the probability of a particular idea due to recent or more extensive experience with it. In fact, strategies that often help in solving problems may exacerbate the situation in the case of a red herring. Wilson suggested to advisors to "search your memory" to possibly find the solution to the problem through previous interaction with the same symptoms. As with any heuristic, this could be helpful most of the time, but harmful if the problem isn't caused by the storm. This is why Wilson's first step of troubleshooting, "listen to the problem," is important. Careful understanding of the problem could avoid wasting time on solutions based on surface features that are similar to an earlier problem.

**Missing Simple Solutions**

In some cases, a simple solution could be tested early but ruled out in error. In one situation, an advisor asked a client to check that there was power to a component. The client noted that the LED lights on the back of the component were lit, and reported to the advisor that there was power. So, a more complex and difficult solution was decided on, and a system board was ordered. However, a few days passed and the advisor happened to look at the system and he noted that power was not in fact reaching the component. The LED lights were powered in a different way than the component was. So, a power supply had to be ordered instead of a system board, adding still more time onto the repair process (G. Dau, personal e-mail communication, Dec.
17, 1995). Confirming this phenomenon, Hill (1993) found, through an experimental study, that clients often did not manage to follow the prescriptive advice they are given by advisors. Also, Ashby (1988) found that when troubleshooters began with simple hypotheses, they were not likely to retest those hypotheses (i.e., double checking the component has power) before moving to the more complex domain. On the other hand, people who started in the complex domain tended to retest the complex hypotheses a number of times before shifting to possibilities in the simple domain.

In many cases, a simple solution may be missed simply because the advisor never considers it at all. Advisors probably do not think "maybe the cable's loose" and consciously decide not to test it. The failure to think of a simple solution may well be the most common cause of failing to test for them first.

A simple solution could be missed simply because the advisor never experienced the particular problem before. One reason a simple-to-test solution is found is that it has recently been the cause of a problem in another case. For example, if a computer advisor had recently worked with client A and discovered that his or her symptoms were caused by a loose printer cable, the advisor may immediately direct client B, who has a similar problem to client A, to check for loose printer cables. It is possible that this may even generalize to testing a variety of simple potential causes, especially if the conversation with client A took an inordinate amount of time before the loose cable was discovered. Therefore, an advisor might be motivated to check for simple solutions because of recent experience of "getting burned" by missing a simple solution.

Lack of experience with a particular simple cause may not be the only reason it is not tested. As Ashby (1988) found, in some situations people
wouldn't test for simple solutions even when all the possibilities were clearly laid out. One of these situations occurred when probability was manipulated. When a subject saw many errors from the complex solution domain, the subjective probability that the next error would also come from that domain increased, and the subject was unlikely to test from the domain of lower probability, even though low-probability items were simple to test for. So, it seems subjects either do not typically use Stolourow's heuristic to combine time and probability, using pure probability as a guide, or they judge the probability of the simple solution to be so low that its ratio is large.

**Expertise of the Client**

Another common feature of the advisory interactions is "verification requests," described earlier in this paper. These mostly seem to be helpful as the keep the conversation within the boundaries relevant to the client, keep the client involved in the interaction, provide feedback to the advisor, and provide possible hypotheses that may prove to be correct. However, there is the possibility that these verification requests could lead to overemphasis on a limited hypothesis domain. For example, the client may introduce a possible red herring: a recent lightning storm, a recent hard drive crash and so on. This is not to suggest that such pieces of information should be withheld or ignored, of course, but they may cause the advisor to begin attempting solutions from an overly complex domain.

Sundstrom and Salvador (1994) looked at how subjects diagnosed a malfunctioning network. Subjects received help from an automated system. The system made recommendations about potential problems to check and gave confidence ratings for those recommendations. When the automated system's recommendations were wrong and confidence levels were low, subjects tended to check for further information before determining a
diagnosis. However, when the automated system provided incorrect diagnoses with high confidence, subjects were more likely to accept the false diagnoses and less likely to check for further information. The machine's confidence might translate to perceived expertise in human-human interaction, or even directly to confidence expressed by the client in certain potential solutions, and advisors could be lead astray by confident, expert clients.

**Communicating the Solution**

Once the problem is solved, the advisor must communicate the solution to the client, who must then implement that solution. Coombs and Alty (1980) found that in the "statement of solution" phase, advisors rarely gave a review of the arguments leading to that solution or a conceptual overview to support the solution. Clients did not tend to ask for clarification of the solution even if they didn't understand it. With expert clients, the control of the interaction was more balanced, and, ironically, the advisors were more likely to support their solutions with explanation. In support of these findings, Alty and Coombs (1980) found that clients were typically satisfied in advisors' ability to solve their problems, but not with advisors' explanations of the solutions. In contrast, advisors generally stated that clients don't want explanations, but simply solutions. The reason for this discrepancy may be that clients tend to misunderstand how complicated a full explanation is. As one advisor stated, "They call up and expect me to solve their whole problem with a sentence or two answer, when I really have to go through their whole system with them" (J. Barney, personal communication, Jan. 15, 1996).

Kidd (1985) found that advisors always offered an explanation of the solution. However, this explanation usually focused on communicating the
solution space rather than on an account of the advisor's reasoning. Thus, the advisor gave conditional rules for when to apply the solution (first try X, if that doesn't work, try Y...), or explained how generalizable the advice would be. Sometimes the advisor justified the advice with a simple "because" clause, particularly if it ran counter to what the client expected. Or the advisor responded to particular queries from the client about parts of the solution. A similar phenomenon exists in doctor-patient communication. Doctors tend to underestimate patients' desire for information, and also for personally-relevant information. Patients generally want information such as, "will I recover?" "what is going to happen next?" whereas physicians tend to give technical information such as the type of disease and stage, and believe that they have adequately informed the patient (Ong, de Haes, Hoos & Lammes, 1995).

Some technical support calls are virtually all explanation. Especially in the cases of simple solutions, the problem frequently stems to a misunderstanding on the part of the client (Wilson, 1991). For example, a client was unable to open the program he wanted to use. The advisor determined quickly that there was a simple solution; the client had several programs open already and was out of memory. Therefore, a large part of the call involved an explanation of how the computer works, and how to close programs to free memory. Much of the technical support agent's job is teaching and explaining (Lohr, 1996) that also needs to be adjusted to the client's level.

Problem Solving

Troubleshooting is a subset of problem solving. It involves an initial state (broken) and a goal state (not broken) and a set of operators that depends on the domain: reboot the computer, test node 43, or click the third checkbox
in the "preferences" dialog box. Therefore, a study of technical support can help explain how people solve real-world problems. Problem-solving theory has three main perspectives: information processing perspective as described by Newell and Simon's (1972) General Problem Solver (GPS), the Gestalt perspective, that focuses on restructuring as the key to insight, and schema theory, that focuses on the organization of knowledge in the problem-solver's head.

These three theories were each developed on different kinds of tasks. The information-processing perspective focuses on knowledge-lean problems that take many steps to complete, such as Towers of Hanoi or the Missionaries and Cannibals problem. Gestalt theorists also focused on knowledge-lean problems, but these are problems where one particular step is key and solving the problem is entirely dependent on the insight to discover that step. Schema theory focuses on knowledge-rich tasks such as algebra or chess. All of these perspectives assume implicitly that there are two basic subprocesses to problem solving: understanding and search (VanLehn, 1989). Understanding is the process of collecting information about the problem. This is done by reading the problem description in paragraph form, by discovering the problem's existence on one's own (e.g., noticing that the car won't start), or by questioning someone else about the problem, as a computer advisor does with a client. Search is the process of looking for a solution. In information processing theory, search is the process of transforming the problem state (physically or in one's head) by applying operators and attempting to make the current state match the goal state. In Gestalt theory, search can be considered the process of trying out different solutions, or by trying to think of a way to restructure the problem. In schema theory, search is the process of searching for a schema that fits the information available.
Information and search need not occur sequentially; problem solvers often alternate between them many times, looking back at the instructions in a laboratory task, or asking the client for more information after making one or more attempts at the search process (VanLehn, 1989).

Newell and Simon's (1972) general problem solver has been found to apply to knowledge-poor, well-defined problems. According to this theory, the problem solver works in a problem space that includes the initial state, the goal state, and a set of operators that can be used to transform the current state to a new state. Observation of problem solvers who work problems that best fit this paradigm – Towers of Hanoi, the nine dot problem – has shown that they use several heuristics. The heuristic of means-ends analysis involves identifying the difference between the current and goal state, finding an operator that functions to reduce that difference, and then applying that operator. The heuristic usually involves setting subgoals because the desired operator cannot always be immediately applied. The subgoal becomes to figure out how to use the operator. Other heuristics identified by Newell and Simon include the difference reduction method (transforming the current state to resemble the solution state), working backward (transforming the initial goal into a set of subgoals and the working toward them), and problem solving by analogy (looking to solved examples for information to solve the problem).

Gestalt theorists focus on the function of restructuring in the problem-solver's mind to reach a solution (Ohlsson, 1984a, for a review). Restructuring occurs when the problem solver changes his or her mental representation of the problem. For example, in the two-string problem where the task is to tie two suspended strings together and the strings are too far apart to reach at the same time, the problem solver must restructure his or
her view of the hammer provided, and view it as a pendulum weight rather than something to drive nails. The Gestalt view of the understanding and search processes consists of comprehending the goal, searching for a way to achieve it, and then reaching an impasse. Then, a different type of search must occur, a search for a different representation of the problem. If the situation is successfully restructured, the problem solver realizes the hammer can be a pendulum weight, the problem is all but solved. The mechanism of restructuring is not known, nor do we know what triggers it. However, it does seem to occur in situations where problem solvers reach impasses.

Schema theory, like Gestalt theory, is based on a person's mental representation of the problem. Schema theory is usually applied to knowledge-rich domains, and each schema "contains" the knowledge needed to solve a particular problem. Schemas contain procedures that enable the problem solver to move from the current state to the solution state with a minimum of additional mental effort. So, once the correct schema is selected, the problem is effectively solved, just as in the Gestalt perspective once the correct restructuring is discovered, the problem is all but solved. Schema theorists don't usually use the term "search." The correct schema is "selected."

How do these different perspectives of problem solving fit together? Gick (1986) described a model that combines the information-processing and schema-driven perspectives. In this model, the problem solver initially constructs a representation of the problem space. The process of constructing this representation may activate a schema the problem solver has previously acquired. If a schema is activated, the problem solver simply implements the solution the schema dictates. If no schema is activated, the problem solver must then enter a search process like the one described in information
processing theory (see Figure 2). Thus, the two perspectives exist side-by-side, depending on the problem-solver’s previous experience with the particular problem type. In more familiar domains a schema is activated and search is avoided, and in less familiar domains no schema is activated and the problem solver invokes a search process.

![Flowchart](image)

*Figure 2: Combination of information processing perspective and schema-driven perspective of problem solving taken from Gick (1986).*
Ohlsson (1984b) described a similar complementarity of the information processing approach and the Gestalt perspective. In this model, the problem solver who is searching in the problem space proceeds through various problem states or knowledge states, as in the General Problem Solver of information processing theory. Then, at each problem state, a description space exists. Each potential description of a particular problem state is equivalent to a representation of the problem, and different operators apply depending on the description. According to Ohlsson, the problem solver spends most of his or her time searching the problem space, moving from problem state to problem state. However, he or she can at any time begin searching the description space for the current problem state. This is where Gestalt-style restructuring occurs. Ohlsson calls restructuring a "retrieval" process, wherein the problem-solver's preexisting knowledge is used to reinterpret the current state, and then that new interpretation is propagated to the rest of the problem space. This restructuring is then followed by further searching through the problem space if the solution has still not been reached. A representation of this model is shown in Figure 3.

Figure 3: Combination of information processing perspective and Gestalt perspective of problem solving as described by Ohlsson (1984b). "S" stands for "Problem State" and "D" stands for "Description."
Ohlsson proposed two types of search. Searching through the problem space is search in the traditional sense, and in the sense used by Newell and Simon to describe the process of applying operators in an attempt to get, step by step, closer to the solution state. The other type of search is the search for a new description in an effort to restructure the problem. Therefore, the problem solver sometimes searches for a procedure and other times searches for a representation of the problem. Gick (1986), in combining schema-driven and information processing approaches said that the problem solver *either* engages in search *or* invokes a schema. However, implicit in the model is a search process wherein the problem solver invokes a schema that is not successful in solving the problem. Then, the problem solver returns to either a search for a procedure, or constructs a new representation of the problem (see Figure 2). This cycle of attempting to apply a schema, failing, and returning to construct a new representation is congruent to Ohlsson's search through the description space. The problem solver attempts to match schema after schema to the current problem.

Ohlsson suggested several heuristics the problem solver might use to decide when restructuring (searching a description space) should be done. For example, the problem solver might use the "restructure-when-stuck" heuristic that dictates that the problem solver continue moving through the problem space until no more useful moves can be found, and then restructure. The "restructure-on-novelty" heuristic dictates that the problem solver restructure when new information is discovered. The "restructure-on-overload" heuristic dictates restructuring when it becomes difficult to keep all relevant information in mind at once.

Restructuring when stuck (Ohlsson) or changing to a new hypothesis domain after exhausting the current one (Levine) is similar to the process of
selecting a new schema when implementation of the first schema fails, as shown in Figure 2. In each model (Levine's domains, Ohlsson's Gestalt restructuring, and schema theory), the problem solver reaches an impasse and is forced to construct a new representation of the problem.

Larkin (1983) used the schema perspective to describe how expert physicists solved novel problems. The physicists selected representations of the problems in successive attempts. They would make a "rather quick and uncritical" selection of a physical representation, or schema, and then attempt to instantiate it. If they reached an impasse, or an inconsistency in attempting to instantiate a particular schema, they would drop that schema and select another. The expert physicists did not carry out any calculations until they found a schema with no inconsistencies, and then the calculations proceeded in a straight-forward manner. The "search" portion of problem-solving was over once the correct representation was found.

Schemas are highly generalizable to all types of problem solving. Schemas are representations of information. This can be any type of information, such what a house is like or what to do in a restaurant. The term "schema" is often considered synonymous with (or at least is a superset of ) "script" or "frame." In troubleshooting, researchers usually refer to "frames." I described frames in the context of frames briefly in the introduction. I return now to use frames to help clarify the application of knowledge representation to technical support.

Govindaraj and Su (1988) studied of fault diagnosis of a marine power plant and found that troubleshooters used two types of knowledge: symptom knowledge and system knowledge. Symptom knowledge, learned through experience, consists of a set of rules that associate symptoms with non-obvious symptoms or causes. System knowledge is the troubleshooter's
mental model of how the system works and how components are connected
together. Govindaraj and Su found that for marine power plants, system
knowledge was well defined. It was not identical in all subjects, but it was
hierarchical and organized at various levels of abstraction.

Hypothesis frames consist of several slots, and the slots that exist vary
depending on the domain. As described earlier, the generalized hypothesis
frame consists of "causes" and "expectations" (Smith et al., 1986). Govindaraj
and Su extended this and made it specific to their domain. They described
four slots: symptoms, components, inference, and flow. Symptoms are
indexed to the symptom knowledge base and components (a list of all system
components relevant to the particular failure the frame describes) are indexed
to the system knowledge base. The inference slot is filled with a set of
propositions that describes how each symptom arises from the failure. The
flow slot is highly specialized to the power plant domain, and indicates which
type of flow the problem is associated with: steam, feedwater, etc. Govindaraj
and Su described a model (Figure 4) for how each type of knowledge
contributes to the selection of the correct frame. As new symptoms are found,
they are used to search for new rules in the symptom knowledge base.
Discovering these rules elicits more specific frames. More specific frames give
rise to the need to check for more symptoms. The system knowledge base
provides information for the selection of frames. Frames are updated by
inquiring into the system knowledge base.
Figure 4: Model of how the symptom knowledge base and system knowledge base contribute to the selection of frames. From Govindaraj and Su (1988).

One goal of the present study is to evaluate problem solving theory in light of the field data collected. Therefore, I discuss examples of problem solving in technical support in the context of all these theoretical perspectives and attempt to describe a model that takes from all the perspectives but also describes the data as adequately as possible.

**Descriptive Field Study**

Problem solving by technical support agents is not a solitary activity, but involves a complex interaction between the client (an expert on his or her own particular goals and symptoms) and the advisor (an expert on the tool the client needs to reach those goals). The advisor needs to get information and constraints from the client in order to create hypotheses. This information may come implicitly or explicitly from the client. The client can participate in the negotiation of the solution, and depending on the context of the interaction, the client’s input may help or sidetrack the problem solving process or be ignored. This study focuses on how the two participants interact
— when they ask questions, how much they participate, and when they give information. The study also focuses on the problem-solving process. Can existing theories explain the process used in these real-life cases? What heuristics do advisors invoke, and how does the interaction with clients influence those heuristics?

The data in this study are descriptive and mostly case oriented. Descriptive field studies and case studies are a good way to identify important variables in new research areas (Stanovich, 1996). Case studies are common in computer usability research, and have been used to examine computer use in a social context (Nardi & Miller, 1990). They are also the primary method of psycholinguistic research into doctor-patient communication (e.g. Fisher, 1983; Tannen & Wallat, 1987), which is similar to technical support. Research on technical support problem solving is not well advanced. Only one published paper, to my knowledge, has used a commercial technical support agency in a field study, and it focused on organizational aspects rather than call content (Pentland, 1992).

Studies Using Related Methodologies

Suchman and Trigg's (1991) recommendations for sorting through video tapes of ethnographic data are easily transferable to audio tape. They suggested beginning with "issue-based" logging, carefully noting interesting events on the tape and recording them with chronological keys. Quantitative coding of the information can then be carefully recorded by research assistants. Suchman and Trigg indicated that simply listening and re-listening to certain cases is the most effective way to form theoretical insights. These insights can lead to formation of new themes and categories that can later be quantitatively coded for.
Frohlich, Drew, and Monk (1994) examined how pairs of computer users learned to use new software by videotaping their interactions. The preliminary data analysis was done through direct observation of the tape, with certain pertinent episodes transcribed for detailed analysis. This method allowed for more accurate evaluation of intent of statements and for subtle cues to be derived from intonation and timing. Frohlich, et al. used their selected episodes to describe different ways computer users recovered from errors, and discovered a "massive gap" between the ideal interaction methods that software designers intended computer operators to use and the actual sequences users went through. The researchers began with an open-ended question, "What methods or procedures do users employ to recover from error?" The descriptive data analysis method Frohlich, et al. used revealed interesting and useful insights such as systematic ways users responded to delays in the face of different kinds of feedback, how users responded after mistakenly asking the computer to do something, and so on.

The human-computer interaction literature provides many examples of studies that use descriptive analysis of ethnographic data to answer open-ended questions. Nardi and Miller (1990) studied how spreadsheet programs were used in cooperative work environments. They found that what they were interested in, the kinds of problems people use spreadsheets to solve as well as how users structure the problem-solving process, would be impossible to study through controlled methods that prescribe tasks to the participants. Instead, Nardi and Miller interviewed spreadsheet users and found that spreadsheet development was often distributed over several users, and that coworkers cooperated in a variety of ways. Some users specified a spreadsheet design and asked coworkers with more programming experience to implement it, some coworkers taught less experienced coworkers about
advanced features, and some coworkers passed files back and forth and added pieces over time.

Rogers (1992) studied how groups of computer users coordinated the troubleshooting of a shared network system by presenting a detailed analysis of a single case where a printer on a computer network had stopped working. She described the stages of the cooperative troubleshooting process: identification of the problem, making the problem public, widening transmission of the problem, initial hypotheses, secondary hypotheses, and so on. Rogers described complex ways that the problem transformed from belonging to one person to belonging to the group, how group members with different working hypotheses failed to communicate, and gave many examples where the distributed problem solving sequence was inefficient although ultimately successful. Rogers presented a single detailed example in her paper, but many of her insights came from observation of many such troubleshooting episodes. By observing a series of episodes, Rogers was able to determine which types of communication problems were typical, and presented examples from her single case.

Rieman (1993) studied exploratory learning by having subjects keep a diary of learning incidents during the work day. The diaries showed that the subjects could be easily grouped into “explorers” and “nonexplorers” based on the number of reports turned in. Rieman described the most common ways that learning occurred, such as trying things until something worked, reading a manual, or asking others for help. He found that these methods were often combined and that use of these methods were not equally distributed between explorers and nonexplorers.

Psycholinguistics uses ethnographic studies of doctor-patient communication to describe problem solving and diagnosis, the effects of
patient personality and demographics, and the negotiation of treatment options. By examining and interpreting individual doctor-patient interactions, Tannen and Wallat (1983; 1986; 1987) described how mismatched schemas between doctors and patients caused communication problems and increased cognitive loads for doctors. The existence of these mismatched schemas was determined by qualitative evaluation of the interaction by an impartial observer who noted, for instance, when patients asked questions that doctors believed they had already answered. Fisher (1983) followed 21 patients with abnormal pap smears and observed their interactions with doctors. As a result, she described a “negotiation” of treatment decisions between doctors and patients. This included how patients questioned doctors, how doctors questioned patients, and how doctors sometimes presented options to patients and suggested how the information should be interpreted, and other times persuade patients to consent to particular treatments and prescribed how they should interpret the information given to them. Maseide (1983) used a case study of a doctor-patient interaction to describe how theoretically-based models of clinical problem solving ignored social and communicative aspects of the medical interview. Maseide found that although doctors attempted to control the flow of information in the interview to follow the medical model of reasoning, mothers of patients also controlled the flow of information by introducing topics and qualifying their answers. This substantially changed the flow of information from the theoretical ideal.

In studies of dyadic requests for help and advice description of what happens, based on transcripts, is the norm (e.g. Hopper et al., 1992; Hutchby, 1995; Jefferson & Lee, 1981; Whalen & Zimmerman, 1990). In all these cases,
researchers began with the question "How do these interaction work?" and answered with detailed description based on sets of real cases.

As described earlier, previous studies of technical support provide some guidance in content to look for in these interactions. Coombs and Alty (1980) studied computer center employees who helped university computer users. This study focused on identifying conversational stages of the interaction, describing the type of explanations given by advisors, and determining the distribution of control over the interaction between the two participants. Two other studies of technical support followed up on the question of control over the interaction. Kidd (1985) noted that interactions where the locus of control resided too strongly with the advisor were not successful. The user often had contributions and ideas for solutions, or needed to question an advisor's proposed solution. Aaronson and Carroll (1987a) followed this up with a description of "verification requests," that are comments or questions users employ to demonstrate their level of knowledge, help pace the conversation, confirm their understanding, and to assume a more active role.

Method

SHSC Corporation\(^1\) sells a wide variety of software for personal computers, and has a well-respected technical support group of approximately 60 advisors who support the company's products over the telephone. Eleven advisors volunteered to participate in the study, and all were informed that I was observing how technical support advisors solve problems, and that no evaluation would be made of their performance. I sat in each advisor's cubicle during his or her shift, and monitored the tape recorder that was

\(^1\)The name of the company involved has been changed, as well as the names of all products, version numbers, and many of the commands within the products.
plugged directly into the advisor's telephone. The advisors asked each client, "This call is being monitored by a graduate student named Jan. Would it be okay for her to record this call for her research project?" The advisors told clients who asked for more information that I was studying how technical support advisors solve problems. All but one client gave permission to record the call.

One hundred eighty telephone conversations were recorded over a period of two weeks. Advisors were recorded for a maximum of one shift a day. Shifts ranged from one to two and a half hours. For scheduling reasons, some advisors were recorded more than others.

**Quantitative Analysis**

The calls were content-coded in an attempt to measure conversational control between the advisor and client. The coding scheme was based on coding methods of technical support conversations from several previous studies, but with substantial modification.

**Overall Categorization of the Queries**

Each query was categorized by the role the advisor took in the interaction and by how the query concluded. How the query concluded was determined by whether the goal was reached. Table 1 shows a summary of these categories.
Table 1: Each query was categorized by the role the agent took in the interaction and the conclusion of the query.

<table>
<thead>
<tr>
<th>Query Category</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advisor Role</td>
<td></td>
</tr>
<tr>
<td>Informing</td>
<td>Pre-sales (not analyzed further)</td>
</tr>
<tr>
<td>Defining</td>
<td>&quot;How does this work?&quot;</td>
</tr>
<tr>
<td>Indexing</td>
<td>&quot;How do I do this?&quot;</td>
</tr>
<tr>
<td>Structuring</td>
<td>&quot;This doesn't work.&quot;</td>
</tr>
<tr>
<td>Conclusion Type</td>
<td></td>
</tr>
<tr>
<td>Resolved</td>
<td>Goal reached</td>
</tr>
<tr>
<td>Indeterminate</td>
<td>Solution found, but client still needs to carry it out</td>
</tr>
<tr>
<td>Unresolved</td>
<td>No solution found — advisor will follow up</td>
</tr>
</tbody>
</table>

Advisor Role

Each query was evaluated to determine the advisor role. These role types were based on a classification system described by Carroll and colleagues (Aaronson & Carroll, 1987b; McKendree & Carroll, 1986). The judgments were made holistically, and were based on what the advisor spent the most time doing, as opposed to how the original query was posed. Some calls contained multiple queries, and these were each coded separately.

Advisors in the informing role answered requests for general information, especially "pre-sales" queries such as "If I buy your product, will it be compatible with my computer?" or "How can I find a consultant in my region that can help me with your product?" In informing queries, the advisor served as a convenient reference source and focused on general features. These conversations remained mostly hypothetical, and usually did not involve interaction with a computer. Informing-role queries were not analyzed in this study.

The defining role was defined as "How does this work?" or "What would happen if I did this?" and included questions that had to do with a particular part of a product, but were not associated with a problem or often
even a stated goal. For example, "Does the 'find' command always search all the records?" or "How does this database program handle memory?" In defining queries, no goal or problem was ever stated by the client, and the advisor mostly answered questions without probing the client for underlying goals or offering alternative methods.

The *indexing* role was defined as "How do I do this?" and included questions that usually could have been answered through use of the manual or on-line help. For example, "How do I change the fill color in my drawing?" or "How do I import a file that was created in another database program into my new database program?" A query that started out as a defining-role conversation often changed to an indexing conversation when the advisor asked the client for his goal. For example, the client could pose a defining query such as "What does the 'integer' function do?" If the advisor simply defined the term and did not pursue the topic, the query would be "defining." However, the advisor often would define the term and then ask what the client wanted to use that function for. This would introduce a goal into the interaction, and such queries were categorized as "indexing" because they evolved into the advisor giving suggestions and procedures for how to reach the goal.

The *structuring* role was defined as "This doesn't work" or "How do I fix this?" and included problems where the client had tried to accomplish something and got unexpected results or had gotten error messages from the program. In this role, the advisor went through a troubleshooting procedure. Such queries often included an indexing portion, where the advisor described how to implement the solution, but any query that included troubleshooting was classified as structuring despite brief periods where the advisor served another role.
Conclusion of Query

Each query was also categorized by its result. There were three possible outcomes: *resolved* (the goal was reached), *indeterminate* (the goal might have been reached), or *unresolved* (the goal was not reached). In resolved queries, the client's goal was reached during the conversation. For example, if the client wanted to know how to create a blue box, at the end of the query, the client had a blue box on the screen. In indeterminate queries, a solution was found but the client still had to carry out some steps after the call ended. For example, the client might have a set of instructions about how to draw a blue box, but he hadn't actually followed the instructions during the phone call. In unresolved queries, no solution was found, and the advisor intended to follow up on the query by calling the client back.

Content Coding in Previous Technical Support Research

Aaronson and Carroll (1987a) classified client responses as to whether they were verifications ("Is this idea correct?"), requests for more information, rejections, or passive acknowledgments. They further divided verification requests into "not a new idea," "repeat of an earlier solution," "restatement or trivial detail," and "new idea."

Coombs and Alty (1980) described three ways to identify conversational structure. In a statement-statement structure the first speaker notes a deficiency in the conversational partner's knowledge and makes a statement to provide the information. The second speaker acknowledges that. In a question-answer structure, the speaker notices a gap in his or her knowledge and seeks to fill it. The second speaker provides that information. In a command-action structure, the first speaker notices that some action needs to be performed and tells the other conversational participant to perform the action. The second speaker responds appropriately and acts.
Hill (1993) classified types of questions clients asked, coming up with 13 categories. These included "give plan to mentioned subgoal," "describe system object," "explain disappointed expectation," and "verify action as proper."

I developed a coding scheme based on all of these systems, with substantial modification.

**Development of the Coding Scheme**

The coding scheme was developed iteratively by the experimenter and three research assistants. Table 2 shows a summary of the codes developed next to the most similar categories used in previous research. The statement-statement, question-answer, and command-action structures were found to be extremely difficult to identify reliably. Statements were not always followed by statements; sometimes they were followed by responses ("Okay") or questions. Often statements did not come in pairs but in sets of three or four. In these cases, it was very difficult for two coders to agree above a chance level as to which speaker initiated the structure. Question-answer pairs were easier to identify, but even questions were not always followed by direct responses; sometimes they were followed by questions (asking for clarification) or statements that did not directly address the question. Command-action pairs were particularly hard to categorize with high inter-coder agreement. It was often impossible to tell whether the clients actually executed the actions, whether they noted them for later execution, or whether they even understood the commands.
Table 2: A summary of each coding category used in the present study and its closest precedent in previous research.

<table>
<thead>
<tr>
<th>Present Coding System</th>
<th>Previous Research</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client Codes</strong></td>
<td></td>
</tr>
<tr>
<td>Verification Requests</td>
<td></td>
</tr>
<tr>
<td>Procedural</td>
<td>Verify action as proper (Hill, 1993)</td>
</tr>
<tr>
<td>Repetition</td>
<td>Not a new idea</td>
</tr>
<tr>
<td></td>
<td>Restatement or trivial detail (Aaronson &amp; Carroll, 1987a)</td>
</tr>
<tr>
<td>New idea/concept</td>
<td>New idea (Aaronson &amp; Carroll, 1987a)</td>
</tr>
<tr>
<td>Questions</td>
<td>Question-response (Coombs &amp; Alty, 1980)</td>
</tr>
<tr>
<td>Why?</td>
<td>Request more information (Aaronson &amp; Carroll, 1987a)</td>
</tr>
<tr>
<td>Other</td>
<td>Describe system object</td>
</tr>
<tr>
<td>Description</td>
<td>Statement-statement (Coombs &amp; Alty, 1980)</td>
</tr>
<tr>
<td>Command</td>
<td>Command-action (Coombs &amp; Alty, 1980)</td>
</tr>
<tr>
<td>Rejection</td>
<td>Rejection (Aaronson &amp; Carroll, 1987a)</td>
</tr>
<tr>
<td><strong>Advisor Codes</strong></td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Statement-statement (Coombs &amp; Alty, 1980)</td>
</tr>
<tr>
<td>Question</td>
<td>Question-response (Coombs &amp; Alty, 1980)</td>
</tr>
<tr>
<td>Instruction</td>
<td>Command-action (Coombs &amp; Alty, 1980)</td>
</tr>
<tr>
<td>Response</td>
<td>Question-response (Coombs &amp; Alty, 1980)</td>
</tr>
</tbody>
</table>

I suspect that the previous studies that used coding schemes involving these conversational structures simply ignored cases that did not follow the structure. However, I decided to include all utterances in the coding scheme,
and so I unlinked the pairs and simply counted frequencies of statements, questions, responses, and commands. Actions were not counted because all attempts to count these reliably failed.

Aaronson and Carroll's method of counting verification requests was also adapted. The "not a new idea" category was eliminated because it was difficult to distinguish from "repeat of earlier solution." The restatement/trivial detail category was also eliminated and all unoriginal or trivial verifications were categorized as "repetition" verification requests. The "new idea" category was included and titled "new idea/concept." A new category was added and titled "procedural." The need for this category was identified because of the large number of incidents in which clients posed verifications of how to carry out instructions given by the advisors. Verifications were not further categorized as to the degree of certainty, as was done by Aaronson and Carroll, in order to keep the number of categories to a manageable size.

All other client utterances were divided into questions, description, commands, responses, and rejections. Questions were further divided into two categories. Attempts to categorize questions under Hill's 13 categories failed to reach acceptable levels of agreement, so only two categories were used.

**Category Definitions: Client**

Client utterances were divided into nine categories: verifications (three types), questions (two types), descriptions, commands, responses, and rejections.

**Verification – Procedural:** Client verified a step or a detail of how to carry out a step of an instruction.
Here are two examples, where the advisor ("A") gave an instruction and the client ("C") made a verification request.

- A: Throw that file away.
  C: Really throw it away?

- A: Let's restart.
  C: Is that under the special menu?

**Verification – Repetition:** Client verified something that had already been stated or was trivial.

This only applied to conceptual details. Repetitions of how to carry out a procedure were classified under procedural. Trivial verification also included completions of sentences.

- A: It has to be set to "share" in order to share it.
  C: So it boiled down to that the files were set to "don't share," not "share."

- A: The computer will go to that field, put in the data...
  C: ...And close the field.

**Verification – New Idea/Concept:** Client introduced a new step, idea, or possible solution.

- C: Maybe it's not working because I forgot to save what we did before?
- C: Do you think I should change that field to text?
- C: My hard drive crashed yesterday. Do you think that's related?

**Question – Why?:** Client asked for an explanation from the advisor.

The client requested that the advisor give more description of what is going on or why the client needed to follow a procedure. This corresponded to Aaronson and Carroll's "more information" category.

- C: What's the difference between X and Y?
- C: I'm not sure why we're going through all this.
**Question – Other:** All other questions asked by the client, besides “why” questions and verifications.

- C: *How do I write a script to do all my work?*
- C: *Can I do this without choosing X first?*

**Description:** Client told the advisor what was going on, described a problem, or gave information.

- C: *It’s finding those files when I’m running it from the server. It’s just that when I come over as a guest and try to run it, then it says that it can’t find those files.*
- C: *It’s opening the program now.*
- C: *I’ll click on that now and see.*
- C: *Nope, still didn’t work.*

**Command:** Client told the advisor to do something on the advisor’s computer.

- C: *Go under “file open” and see if it works on your computer.*

**Response:** Listener responded to speaker’s statement or question.

Responses could be very brief ("uh huh", "okay"). Longer responses could directly address a question or could affirm a statement.

**Rejection:** The client imposed constraints on the solution space.

In response to the advisor, the client said that he has tried that before, that the advisor didn’t understand the problem, the advisor’s statement was erroneous, or that what the advisor said to do was pointless.

- C: *Okay, this could take some time because it’s got to shut everything down and then move it and then it takes forever to start up.*
- C: *Yeah, but I don’t think that’s the case. It was on the machine at home and it worked, and this is a new folder I just made.*
Category Definitions: Advisor

The advisor's utterances were divided into four categories: statement, question, instruction and response.

**Statement:** The advisor gave information to the client.

- A: *That's the most likely thing that's gonna happen that the folder itself is either locked or it's damaged.*

**Question:** Any utterance by the advisor that asked the client for information, even if the tone of voice didn't indicate it was a question.

- A: *What version of the software do you have?*
- A: *Tell me what it says on that screen.*

**Instruction:** Advisor told the client to do something, or detailed a step the client should do.

- A: *Double click on that.*
- A: *Let's reboot.*

**Response:** Advisor responses were the same as client responses, described above.

Coding Procedure

The coding was done directly from the audio tapes of the phone calls. This method was preferable to coding from transcripts because the coder has access to tone of voice and timing of speech, which made it easier to correctly infer the speaker's intentions.

The unit used for coding was a turn in the conversations, or a change in category within one turn. For example, if a client spoke for several sentences describing his query, the whole turn was coded as one occurrence of "description." However, if the client described the problem and then made a verification request, the turn would receive two codes: description and then verification. All responses that occurred while the other person was talking
were combined into one code. So, if the client spoke several sentences of
description and the advisor said "uh huh" and "okay" three times during the
description, only one advisor response was recorded.

Utterances that were off the topic of conversation were not coded. For
example, if the client asked how the advisor liked her job, no code was
recorded. Also, queries that fell into the "informing" role category were not
coded. In the informing queries, the advisor functioned less as an advisor and
more as a salesperson. Since I was primarily interested in problem solving
and helping, the informing queries were the least interesting data, and so
were not coded. Also, queries about being transferred to the sales department
and where the clients asked to be transferred to a different advisor (whom
they had spoken with previously) were not coded.

**Inter-coder Agreement**

Each call was coded by two independent coders. In most cases the two
were the expert coder and one assistant coder. In a few cases, calls were coded
by two assistant coders. All disagreements were decided through a discussion
between the coders. The expert coder participated in each of these sessions,
including those that were originally coded by two assistants. Therefore, these
sessions served as both a means to obtain the most accurate set of codes, and
also as continual retraining for the assistant coders.

The biggest source of disagreement was not which code to apply to an
utterance, but the boundary of the utterance – whether one or two codes
should be applied. Therefore, many sophisticated methods of inter-coder
agreement were inappropriate for the data, since they count disagreements
but do not allow for one coder to provide a code where the other coder had
none. Therefore, the index of agreement was a simple percentage: for each
query the number of judgments on which the coders agreed was divided by
the total number of judgments. This agreement index is biased toward coding systems with a small number of categories. By chance alone expected agreement is higher with fewer categories. So, the observed agreement must be compared to the amount of agreement expected due to chance. However, the frequency of use of each category is likely to be uneven in that a large number of categories may be available but a few are used most heavily. A more conservative measure of expected agreement considers the relative frequency that each category is used (Scott, 1955). Scott suggested that the measured level of agreement be compared to \( P_e \), which is the expected level agreement purely by chance, taking frequency of use of each category into account. \( P_e = \sum p_i^2 \), where \( p_i \) is the portion of all items that fall into the \( i \)th category.

**Method of Analysis**

Since the number of coded utterances varied widely across queries, it was not possible to compare the raw frequency of a categories across queries. Aaronson and Carroll summed all the frequencies across the 30 queries they studied and presented the frequencies of certain categories as a function of selected other categories. By "summing across queries" I mean that instead of saying, for example, that clients made an average of 6.9 verification requests per query, they said that clients made 208 verification requests over all queries. Then, they presented the use of each category as a function of the other categories. So instead of saying that 5% of everything clients said were "new idea" verification requests, Aaronson and Carroll said that 63% of verification requests were new ideas, or 76% of responses were verification requests. Coombs and Alty also summed categories across queries, but they presented the relative frequency of use of a category by the advisor to the
frequency of use of that same category by the client. So they said 58% of all questions were asked by advisors.

Summing across all queries weights the analysis so that longer queries have more influence than shorter queries. It also means that all statistical analyses have to be nonparametric. Instead of summing, I use "query" as the unit of analysis, so that all queries are weighted equally and statistical tests such as ANOVA and t-test can be used to test for effects. "Query" is the random factor in the analysis. It functions like "subject" does in other psychology studies.

There is no way to escape reporting the use of one category as a percentage of other categories. However, some methods of accomplishing this are more biased than others. For example, reporting the percentage of verifications that are new ideas is strongly influenced by the total number of verifications. So, if in one query, a client makes three verifications and all are new ideas, 100% of all verifications are novel and the client appears to be contributing a lot to the solution of the problem. However, if another client in a query of similar length makes three new-idea verifications but also makes three procedural verifications, only 50% of his verifications are novel and he appears to be contributing less. To mitigate this bias, I report frequencies of the use of categories as a percentage of the total number of coded items made in the query. So, if a client utters 56 coded items in a query and 3 are new-idea verifications, the portion of new ideas is 5.3%, regardless of how many other verifications the client makes. Advisor category use is reported as a percentage of the number of coded utterances made by the advisor and client category use is reported as a percentage of coded utterances made by the client. This method still contains some bias because, for example, if a client utters many description items, the portion of new-idea verifications
will appear artificially lower. But by including all items in the comparison, the influence of the use of one category on another is lessened.

An alternative method would be to use time as a basis for comparison and compute the number of new-idea verifications per minute, for example. However, some of the queries contained long periods of silence while the computer rebooted or the advisor researched the answer to the query, and this method would artificially depress the use of all categories in queries that contain silence. Therefore, I report the use of each category as a percentage of the total number of items coded in each query.

So, for each query, the frequency of use of a category by a speaker (client or advisor) is divided by the total number of utterances made by that speaker, giving the portion of utterances made that fit that category. Reported means are the unweighted means of the percentages for each query. For example, if in query A the client made 100 utterances and 13 were description, descriptions were used 13% of the time. If in query B, the client made 50 utterances, 5 of which were description, descriptions were used 10% of the time. The mean use of description across the two queries, then, is 11.5%. I refer to this as the query-based method.

When comparing the present results to previous research, I used the same method of analysis as was used in that research. Otherwise, I use the query-based method.

Qualitative Analysis

The structuring-role queries were transcribed and analyzed in more detail. I used these queries as case studies to learn about patterns and methods of problem solving and troubleshooting used in technical support.
Results and Discussion

Quantitative Analysis

Advisor Expertise

The advisors studied were experienced and well trained. The eleven advisors' experience in technical support at SHSC ranged from one to nine years, and the average tenure at the time of the study was 3.7 years. Some advisors had previous technical support experience before working at SHSC. Including this experience, the mean amount of experience was 4.2 years. New advisors at SHSC are trained for two weeks before they begin speaking directly to clients. This initial training includes teaching of both technical information about products and good customer service techniques. Trainees also listen to many support calls by experienced advisors. New advisors usually support only one product as they start client interaction, and they add more products as they gain experience. Their interactions with clients are monitored by their mentors as part of the training process. All advisors are continually trained throughout their tenure, attending classes on technical topics, and listening to other advisors talk to clients.

Overall Categorization of Each Query

In the 180 calls recorded, clients posed approximately 233 queries. Queries in which the advisor took the informing role or which did not involve a query at all (asked to be transferred, set up a contract for further technical support) were not analyzed. These eliminated, 175 queries were analyzed. Thirty-seven queries fit the defining role of the advisor, 75 fit the indexing role, and 63 fit the structuring role.

The calls, which sometimes contained more than one query, averaged 12.91 (SD = 9.54) minutes long, but, as shown in Figure 5, the distribution was positively skewed. The median call time of 9.82 minutes was shorter than the
mean. As a point of interest, the longest 20% of these calls (those calls 20 minutes and longer) took 44% of the total time for all the calls recorded.

![Call Time Distribution](image)

**Figure 5:** The distribution of the lengths of the calls.

Since structuring queries were examined in more detail, these 63 individual queries were timed to determine the amount of time spent actually talking about the query rather than the total call time. As shown in Figure 6, the distribution of structuring query times was similar to call time, positively skewed with a mean time of 12.31 minutes ($SD = 8.95$) and a median of 9.00 minutes.
Figure 6: The distribution of the lengths of the structuring queries.

Table 3 shows the portion of queries that were resolved, indeterminate, or unresolved. More than half of the defining queries were resolved, slightly fewer indexing queries were resolved, and a third of structuring queries were resolved. The conclusion of the queries varied as a function of the advisor role, \(\chi^2(4, N = 175) = 21.02, p < .01\).

<table>
<thead>
<tr>
<th>Role</th>
<th>Resolved</th>
<th>Indeterminate</th>
<th>Unresolved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining</td>
<td>20 (54%)</td>
<td>15 (41%)</td>
<td>2 (5%)</td>
</tr>
<tr>
<td>Indexing</td>
<td>33 (44%)</td>
<td>42 (56%)</td>
<td>0</td>
</tr>
<tr>
<td>Structuring</td>
<td>21 (33%)</td>
<td>30 (48%)</td>
<td>12 (19%)</td>
</tr>
</tbody>
</table>

The reasons for indeterminate or unresolved conclusions varied. Indeterminate queries often involved clients who did not have access to their computers when they called. In these cases the advisors usually gave the clients several things to try on their own. In other cases, the solution took a long time to carry out. For instance, in several cases the solution was to download a new version of the software and install it. In these cases, the
advisors gave the clients instructions but did not remain on the phone while the clients carried out the instructions. In some cases the advisors mailed out disks with new software on it. In most indeterminate queries, the advisors seemed to consider the problem solved in that they believed that they had found the right solution and that the client could carry it out. When the query was indeterminate because the client was not at his computer, the advisor usually seemed less certain that the suggested solutions were correct. In unresolved queries no conclusive hypothesis was given and usually the advisor needed to do "further research" such as contacting the engineering department or attempting to replicate the client's exact system.

**Content Coding**

The inter-coder agreement was .792. The percent agreement expected by chance, taking into account the number of categories and the relative frequency of use of each category, was much lower, $P_e = .145$.

**Concerns About Significance Testing**

Below I report the results of the content coding and quantitative data, supporting the results with significance testing. It is important to keep in mind the nature of the data. The data points (individual queries) were not truly independent from one another because the same advisors participated in multiple queries. Different advisors had different styles. Some seemed to ask more questions, and some to describe more. An analysis of individual differences between advisors would be interesting but the current data makes it impractical. One way to compare advisors would be on their relative success or efficiency on similar queries. Unfortunately, there were very few queries on the same topic recorded, so the relative success of different styles was impossible to measure. Another method would be to use "advisor" as an independent factor in the analyses. Most of the analyses described below are
based on the independent variable of advisor role. Using both factors makes the cell sizes very small, sometimes one or zero.

There would be a problem if one advisor had many more structuring-role queries than other advisors did. If that advisor tended to ask many questions, there would be no way to be sure whether a finding that advisors asked more questions in structuring queries was caused by role differences or by advisor differences. However, it does not appear that advisors were badly confounded with advisor role. After eliminating four advisors who contributed on only a few queries (less than ten), the factor of advisor only predicted a small portion of the variance in advisor role, $\phi(N = 154) = .36$. Further, qualitatively, the factor of advisor did not seem to interact with the factor of role. As shown in Appendix A, although some advisors used certain categories more than others, the advisors almost always followed the same patterns of use across roles.

Notice that in general the standard deviations are often quite high, due to the large variability of field-collected data. This lowers the power of the statistical tests but makes the significant results that were found all that much more intriguing.

Comparison with Previous Research

The first goal of the quantitative analysis was to compare the results of this study to previously reported results. Direct comparison was difficult since different coding schemes were used, but I attempted to make comparisons where possible.

Balance of Control

Coombs and Alty (1980) were interested in the degree of client participation in the interaction and focused on the amount of relative control between the client and advisor. As described earlier, they counted occurrences
of adjacency pairs of statement-statement and question-response. Coombs and Alty assumed that the party that initialized the adjacency pair had control. They found that advisors initiated 64% of all statement-statement pairs, a significantly greater portion than the 36% initiated by clients. They also found that advisors initiated 58% of all question-response pairs. However, the difference between the two groups was not significant. Since in the current study I did not count occurrences of adjacency pairs, the statement-statement results could not be replicated, but the relative number of questions asked by each party could be compared. Combining the three types of verifications and two types of questions, clients asked 1407 questions, while advisors asked 1423. So, in the current study, advisors initiated 50.3% of all questions, a much smaller advantage than Coombs and Alty found. Using the query-based method of analysis, clients asked questions in a mean 21.81% \((SD = 14.4\%)\) of all their utterances, while advisors asked questions in an average of 19.85% \((SD = 12.6\%)\) of their utterances. This difference was not significant, paired-samples \(t(174) = 1.08, p = .284\).

Coombs and Alty found a bigger disparity in the portion of questions asked by advisors when queries involving expert and non-expert clients were analyzed separately. With expert clients, advisors asked 49% of the questions, and with novice clients, advisors asked 64% of the questions. In the latter case, the disparity of control was significant. I did not find this result in the current study. In the 50 non-database (less sophisticated) queries, advisors asked 50.005% of the questions, and this equality was also evident when the data were analyzed with the query-based method. Advisors asked questions in 21.37% \((SD = 15.3\%)\) of their utterances and clients ask questions in 21.53% \((SD = 10.7\%)\) of theirs, paired-samples \(t(49) = .05, p = .960\). In 125 database (more sophisticated) queries, advisors asked questions in 19.24% \((SD = 14.0\%)\)
of their utterances and clients ask questions in 21.91% (SD = 13.4%) of theirs, paired-samples t(124) = 1.22, p = .224.

A better gauge of who controlled the queries than who asked more questions or made more statements is simply who made more utterances. Because the basic unit of coded items was conversational turn (or change of category in one turn, but this happened infrequently), the portion of all coded items per query that were made by advisors (M = 49.5%, SD = 3.8%) was not significantly different from the portion made by clients (M = 50.5%, SD = 3.8%), paired samples t(174) = 1.78, p = .077. However, the most frequent category used for both advisors and clients was “response.” Responses were passive, and were always in direct reaction to the other party. All code categories except response represented some degree of conversational control taken by the speaker. Counting all items except responses, a greater portion of advisors’ utterances were controlling (M = 66.4%, SD = 13.7%) than were clients’ (M = 52.8%, SD = 14.0%), paired samples t(174) = 7.08, p < .01. So advisors appeared to take more conversational control than clients did in that advisors were more likely to make a statement or ask a question than to passively respond to the other party. Coombs and Alty, however, did not adequately prove that this difference was necessarily a bad thing. After all, the advisor was being ask to impart her knowledge to the client.

Client Participation

Aaronson and Carroll (1987a) found technical support interactions to be much more complex and varied than Alty and Coombs described them to be. To examine conversational control, Aaronson and Carroll focused on whether clients fully participated in negotiating the solution to their query, or whether they passively accepted what the advisor said. This method did not take into account what the advisor did, but it focused more on whether the
client made substantive contributions to the interaction rather than if they simply talked more or less than advisors.

Aaronson and Carroll counted the different ways that clients responded to advice after the presentation of the initial query. I did not differentiate between acknowledgments made during the initial query and the rest of the interaction, however, the initial query almost always consisted of client descriptions and advisor questions and not client verifications, questions, or rejections, the categories that interested Aaronson and Carroll.

Table 4 compares the relative frequencies of different kinds of client responses. These categories can be directly compared because they were defined the same way in both studies. The relative frequencies for verification requests and requests for more information were remarkably similar across the two studies. There were a few more rejections in the current study. That may be because the current study was over the telephone, and advisors had more difficulty understanding all of the clients' constraints since they couldn't see their computer screens.

<table>
<thead>
<tr>
<th>Response Type</th>
<th>n</th>
<th>%</th>
<th>Response Type</th>
<th>n</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verification</td>
<td>208</td>
<td>76</td>
<td>Verification</td>
<td>865</td>
<td>71</td>
</tr>
<tr>
<td>More information</td>
<td>33</td>
<td>12</td>
<td>Question Why?</td>
<td>111</td>
<td>9</td>
</tr>
<tr>
<td>Rejection</td>
<td>34</td>
<td>12</td>
<td>Rejection</td>
<td>242</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 4: Comparison of client responses in previous research to responses in the present study.

Aaronson and Carroll also counted verbal passive acknowledgments of advice to determine whether clients more often negotiated advice or simply accepted it. They counted 112 passive acknowledgments, just 40% of all acknowledgments. They believed this showed that clients were actively involved in the negotiation of the solution. In the present study, there were 3298 client responses, or passive acknowledgments, which is more than two
and half times the number of other types of acknowledgments shown in Table 4. However, I did not distinguish between responses to advice and responses to anything else, such as agent questions or statements of fact.

Aaronson and Carroll next analyzed the relative frequency of the use of different kinds of verification requests because they were interested in how often clients introduced novel ideas. Unlike the response types described in Table 4, these results cannot be easily compared across studies because of the modifications I made to the categorization of verification requests. Aaronson and Carroll did not have a category that corresponded to "procedural" verification requests, and the three categories they used for non-novel verifications were all combined into "repetition" in the present study.

Still, Aaronson and Carroll were most interested in the relative frequency of new ideas as compared to non-new ideas, and the new idea category did exist in both coding schemes. However, as shown in Table 5, Aaronson and Carroll found a much larger rate of new ideas (63%) than I did in this study (25%). On the other hand, Aaronson and Carroll's three non-new idea categories account for 37% of the verification requests, which was almost identical to the portion of repetitions found in the current study.

*Table 5: Types of verifications made by clients in previous research compared to those made in the present study.*

<table>
<thead>
<tr>
<th>Verification Type</th>
<th>n</th>
<th>%</th>
<th>Verification Type</th>
<th>n</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not a new idea</td>
<td>16</td>
<td>8</td>
<td>Repetition</td>
<td>312</td>
<td>36</td>
</tr>
<tr>
<td>Repeat of earlier</td>
<td>13</td>
<td>6</td>
<td>Procedural</td>
<td>338</td>
<td>39</td>
</tr>
<tr>
<td>Restatement</td>
<td>48</td>
<td>23</td>
<td>New idea/concept</td>
<td>215</td>
<td>25</td>
</tr>
<tr>
<td>New idea</td>
<td>131</td>
<td>63</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

One explanation for the disparity of new ideas is that when procedural verifications occurred in Aaronson and Carroll's queries, they were counted as new ideas. I don't believe this is an accurate explanation though, because in the examples Aaronson and Carroll gave, it appeared that they categorized
questions about procedural details as restatement/trivial detail, and that new
idea was only given for verifications that were conceptual or suggested
solutions, just as in the current study. An alternative explanation is that the
clients in the current study were simply less sophisticated and or less willing
to introduce new ideas. Or, the reason for the lower rate of novel verification
requests in the current study could again be an artifact of the conversions
being over the telephone. Aaronson and Carroll pointed out that
verifications could be useful for the client to pace the conversation and elicit
the desired level of explanation from the advisor. These needs might be
greater over the phone, since the advisor can't see from the client's face
whether the pace is too fast or slow, or whether the client understands. Thus,
the client has to continually give this information through non-novel
verification requests.

Aaronson and Carroll wondered if the number of new ideas put forth
by clients would vary as a function of the sophistication of the client. They
studied queries from two advisors, one who helped with questions about
personal computers and one who helped with text processing. The personal
computer queries tended to be more sophisticated, and as shown in Table 6,
clients who made sophisticated queries also made a greater portion of new-
idea verification requests. The current study had a similar division, in that
71% of the queries were about a sophisticated database program, and the rest
of the queries were in reference to simpler software programs. As shown in
Table 6, the database clients made a slightly greater portion of new-idea
verifications, although the difference was by no means as dramatic as
Aaronson and Carroll's results.
Table 6: Verifications that are new ideas as a function of queries about sophisticated or less-sophisticated topics.

<table>
<thead>
<tr>
<th>Aaronson &amp; Carroll, 1987a</th>
<th>Current Study</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sophisticated (PC)</td>
<td>Sophisticated (Database)</td>
</tr>
<tr>
<td>72%</td>
<td>27%</td>
</tr>
<tr>
<td>Less Sophisticated (Text)</td>
<td>Less Sophisticated (Non-database)</td>
</tr>
<tr>
<td>33%</td>
<td>20%</td>
</tr>
</tbody>
</table>

Using the query-based method of analysis and including only queries where there were verification requests (eliminating 28 queries), the difference between database and non-database queries was larger. New ideas accounted for 31.5% ($SD = 33.5\%$) of verifications in database queries, and 21.0% ($SD = 26.0\%$) of non-database queries. A paired-samples t-test with unequal variances showed that this difference was significant, $t(113.7) = 2.08, p = .040$. However, when new ideas were counted as a percentage of all client items instead of verifications, the difference was no longer significant. New ideas occurred in an average of 4.22% ($SD = 4.0\%$) of client utterances in database queries where there were verifications and 2.86% ($SD = 5.5\%$) in non-database queries. This difference was not significant in a paired samples t-test with equality of variance, $t(145) = 1.53, p = .129$.

Comparison to Previous Research: Conclusions

Coombs and Alty (1980) concluded that advisors took a large degree of control in technical support conversations, and that especially for novice clients, advisors were not particularly supportive of client needs. Based on the data that I was able to compare to Coombs and Alty, I did not replicate this finding. Advisors and clients asked about the same number of questions, and this did not change when taking more and less sophisticated clients separately. However, advisors made a greater portion of substantive (non-response) utterances than clients, so on the most general level, there was some similarity between the present and Coombs and Alty's findings.
Aaronson and Carroll concluded from their study that clients actively participated in the negotiation of solutions to their problems. Verification requests helped clients influence advisors to modify their advice according to the clients' ideas, levels of knowledge, needs, and understanding of the advisors' explanations. Aaronson and Carroll found that clients made many more verification requests than rejections or direct requests for information without expressing an idea of what the information would be (the category of "more information"), and that result was replicated in the current study. Aaronson and Carroll also found that clients made many more verification requests that contained new ideas than other types of verification requests, and this result was not replicated. In addition, they found that sophisticated clients made a greater portion of new-idea verifications than less sophisticated clients did. In the current study, I found a slight tendency in that direction, but the result was not nearly as conclusive as was found in the previous work.

The possible reasons for the different results between the current study and both Coombs and Alty and Aaronson and Carroll are numerous. There were methodological differences in the coding of each of the studies, and the context of the interactions were different in each case. The previous work was based on face-to-face interactions rather than telephone conversations. Aaronson and Carroll found substantial differences between the types of verification requests made by more- and less-sophisticated clients, but client sophistication was completely confounded with advisor in that study. One of the two advisors answered sophisticated client questions and the other answered less-sophisticated client questions. Perhaps the second advisor seemed less open to client ideas.
Also, since Aaronson and Carroll's study was published in 1987, and especially since Coombs and Alty's paper in 1980, software and software users have both changed considerably. Software has become easier to use but also more powerful and complex. More people have become computer users, and so it is possible that the "average" user is less sophisticated, accounting for the lower portion of new-idea verification requests in the current study. On the other hand, with easier-to-use software, there may now be fewer differences in help-requesting behavior of more- and less-sophisticated clients, which would explain the relative lack of a difference between those two subpopulations in the current study. Or, the criteria for more vs. less sophisticated clients in the current study was simply too gross to show the differences.

The data in the current study cannot be used to determine the reason for the differences between the two sets of results. However, the purpose of the current study is not to identify differences in technical support interactions across decades or context of interaction, but instead to investigate the problem-solving process. What the current study replicated from the earlier work is that verification requests were common in the technical support interactions, and that clients were more likely to attempt to verify what they know than to ask open-ended questions. Verification requests not only tell the advisor what the client needs to know, but they also give information about what the client already knows.

**Differences as a Function of Advisor Role**

Interesting differences occurred as a function of the query type. For example, Table 7 shows the relative frequency of each type of verification for each role type. The use of each category varied as a function of role type, $\chi(4, N = 865) = 64.53, p < .01$. 
Table 7: Relative frequency of each type of verification request as a function of the advisor role.

<table>
<thead>
<tr>
<th>Advisor Role</th>
<th>New idea</th>
<th>Procedural</th>
<th>Repetition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining</td>
<td>34 (30%)</td>
<td>12 (11%)</td>
<td>67 (59%)</td>
</tr>
<tr>
<td>(37 queries)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indexing</td>
<td>73 (19%)</td>
<td>162 (42%)</td>
<td>149 (39%)</td>
</tr>
<tr>
<td>(75 queries)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structuring</td>
<td>108 (29%)</td>
<td>164 (45%)</td>
<td>96 (26%)</td>
</tr>
<tr>
<td>(63 queries)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As shown in Table 7, most of the verification requests made in defining-role queries were repetitions. In these queries, the advisor explained how something worked, so there were few step-by-step instructions. The advisor focused instead on explaining a concept. Therefore, since the client's goal was to gain understanding rather than carry out a procedure, he often repeated the advisor's explanation to confirm his understanding. Most of the new idea verifications in defining queries occurred when the client was showing his understanding by extending what the advisor had explained. In indexing queries, there were fewer new ideas, mostly because the client focused on following the advisor's instructions. He didn't usually introduce many new hypotheses, but verified how to carry out the steps the advisor gave and also used repetition verification requests to verify the explanations that she gave for what each step would do.

Structuring queries had the lowest proportion of repetitions. In these queries, the clients often focused less on gaining understanding than on generating hypotheses on how to fix the problems they were having. Therefore, they introduced some new ideas, often hypotheses of causes to their problems, and they verified the steps the advisors asked them to take to test out hypotheses or to carry out solutions, but they repeated advice less often. One of the functions of verifications that Aaronson and Carroll gave was to rehearse facts to increase memory retention. Logically, if clients were
using verifications to help their memory, repetition verification requests
would be the type they would make. Since structuring queries were often
used to solve current problems, there was less need for clients to remember
everything that happened during the interaction because once the problem
was solved, they wouldn't (presumably) need to solve it again. Therefore,
they made fewer repetition verification requests.

Switching now to the query-based method of analysis, clients made
fewer verification requests of all kinds in structuring queries. As shown in
Figure 7, only 8.7% ($SD = 5.8\%$) of client utterances in structuring queries were
verifications, while verification requests made up 13.3% ($SD = 9.1\%$) of items
in indexing-role queries and 14.9% ($SD = 12.8\%$) in defining-role queries. A
one-way ANOVA indicated that this difference was significant, $F(2,172) = 6.85$, $p < .01$, and a Tukey test indicated that the structuring-role queries were
significantly different from the other two groups, which did not differ from
each other.
Figure 7: Portion of description, verification, and question items made by clients as a function of advisor role in the query. "Verifications" combines the three categories of verifications, and "questions" combines the two question categories.

The portion of questions asked by clients in structuring queries was also less than in other types of queries. As shown in Figure 7, clients asked questions in 5.8% (SD = 5.6%) of their utterances in structuring queries, in 10.2% (SD = 9.4%) of their indexing queries, and in 15.9% (SD = 12.8%) of their utterances in defining queries. This difference was significant based on a one-way ANOVA, $F(2,172) = 14.55, p < .01$. A Tukey test indicated that again, the structuring-role queries differed significantly from the other two groups, which did not differ from each other.

Thirty point four percent (SD = 9.4%) of client utterances in structuring queries were description, while the means for indexing (M = 26.1%, SD = 9.4%) and defining (M =25.9%, SD = 14.1%) queries were slightly lower. A one-way ANOVA indicated that this difference was significant, $F(2,172) = 3.40, p =$
.036, but a Tukey test (a conservative test) found no significant difference between any two groups.

Not surprisingly, as shown in Figure 8, use of advisor categories also varied by advisor role. In defining queries, 7.8% (SD = 9.0%) of advisor’s utterances were instructions, while instructions made up 20.5% (SD = 12.9%) of advisor utterances in indexing queries and 19.0% (SD = 11.0%) of utterances in structuring queries. This difference was significant, $F(2,172) = 16.04, p < .01$, and a Tukey test indicated that the defining queries differed significantly from each of the other two types, which did not differ from each other. These results were parallel to the verification request results shown in Table 7. Fewer of client's verification requests were procedural in defining queries. Advisors gave clients fewer instructions to carry out in these interactions, so there were fewer procedural details that clients needed to verify.

![Graph showing the portion of advisor items](image)

**Figure 8:** Portion of instruction, question, and statement items made by advisors as a function of their roles in the query.

Advisor statements and questions showed an opposite trend to client description and questions/verifications. Clients asked fewer questions in
structuring queries, but advisors asked more. Questions account for 27.8\% (SD = 12.8\%) of advisor utterances in structuring queries, but only 17.0\% (SD = 12.7\%) of utterances in indexing queries and 12.1\% (SD = 13.9\%) of utterances in defining queries. This difference was significant, \( F(2,172) = 20.02, p < .01, \) and a Tukey test indicated that the structuring queries differed significantly from the other two groups, which did not differ from each other.

While clients did less "telling" in defining queries than structuring ones, advisors did more. Advisor statements accounted for 36.0\% (SD = 10.8\%) of their utterances in defining queries, 29.1\% (SD = 11.3\%) in indexing queries, and 25.7\% (SD = 9.7\%) in structuring queries. This difference was significant, \( F(2,172) = 11.18, p < .01. \) A Tukey test indicated that the defining query group differed significantly from each of the other two groups, which did not differ from each other.

The portion of controlling utterances (everything but responses) made by each speaker also varied as a function of advisor role. As described earlier, across all queries advisors made a greater portion of controlling utterances than clients did. However, this difference did not exist for all the types of queries. As shown in Figure 9, clients decreased the portion of their utterances that were controlling when moving from defining to structuring queries, while advisors increased theirs. 59.7\% (SD = 14.7\%) of client's utterances were controlling in defining queries, 53.1\% (SD = 13.1\%) were controlling in indexing queries, and 48.4\% (SD = 13.3\%) were controlling in structuring queries. For advisors, 56.0\% (SD = 12.7\%) of utterances in defining queries were controlling, 66.5\% (SD = 14.0\%) of utterances in indexing queries were controlling, and 72.4\% (SD = 9.8\%) of utterances in structuring queries were controlling. A 2 (speaker) \times 3 \) (role) mixed-design ANOVA indicated that
there was a significant interaction between speaker and role, $F(2,172) = 16.28, p < .01$.

![Chart showing the portion of items for each role across utterance types](image)

**Figure 9:** Portion of each speaker's utterances that are controlling, as a function of agent role in the query.

**Advisor Role: Conclusions**

Role of the agent appears to be an important determinate of the format of the technical support interaction. I expected some differences. I predicted that there would be more instructions and more procedural verification requests in indexing queries than in structuring and especially defining queries, and that there would be some differences (although I didn't hypothesize the direction) in degree of client participation between structuring queries and the other two types. However, I did not predict that advisor role would be as predictive as it turned out to be, influencing the use of almost every utterance category. These results are even more interesting when one considers the fact that the advisor-role categories are rather gross distinctions. Nearly every query involved different episodes in the course of
the conversation that more than one advisor-role type. Queries were
categorized by what the advisor did most of the time. If each episode where
the advisor took a different role were analyzed separately, the differences
would likely be much greater.

I learned from this analysis that both advisors and clients interacted
differently depending on the role that the advisor played. Of course, the role
the advisor played was heavily influenced by the query the client posed, and
perhaps also by subtle factors introduced by the client: his displayed levels of
competence and confidence, which questions he asked and what he chose to
describe to the advisor. But regardless of how the advisor came to play the
role she played in a particular query, that role seemed to partly determine the
types of utterances both parties made.

For structuring queries, that meant that clients made more procedural
verifications than new ideas or repetitions, and that clients described more
but asked questions and verified information less than in indexing or
informing queries. On the other hand, advisors asked more questions in
structuring queries, and described and explained less. They gave instructions
at about the same rate that they did in indexing queries, but much more than
in defining queries. Finally, advisors seemed to simply talk more than clients
did in structuring queries, as shown by the portion of non-response
utterances made by each party. Therefore, a structuring query could generally
be characterized as one where the advisor took charge by asking lots of
questions to get information from the client, who spent a lot of time
describing what was happening and following instructions. The client didn't
ask many questions or even make very many verification requests. When he
did make verification requests, they were usually about procedural details,
not providing new ideas or repeating concepts.
Therefore, in the qualitative analysis I examine the problem solving that occurs in these interactions as being directed by the advisor. The advisor gets her information from the client, who carries out steps and reports what happens on the computer. What the client describes and how he describes it is crucial to the process and could influence the strategies the advisor used to solve the problem. Therefore, I examine the process qualitatively from the perspective of the advisor as problem-solver, but always consider the client as an important variable who can affect the process through the form, accuracy, and completeness of the information he provides, and who also can introduce new directions to the problem-solving process by asking questions or proposing solutions.

**Qualitative Analysis**

I analyzed the 63 structuring queries in detail in order to learn about the problem solving strategies used in technical support. The analysis was mainly descriptive and not critical, because in most cases it was not possible to evaluate the success or failure of the strategies used. As shown in Table 3, the success of the interaction was certain (the conclusion was "resolved") in only a third of the queries. The actual success rate in a single structuring interaction may be much higher – up to a maximum of 81% if all the indeterminate queries were successful. Unfortunately, I was not able to follow up with the clients after their calls. So, since success or failure of the problem solving done is difficult to evaluate, I focused on examining heuristics the advisors used regardless of whether they found the correct solution.

Time was not a good measure of efficiency, because some problems were very difficult, and should have taken more time. Also, taking extra time to explain the steps to the client or to wait while the client carries out longer tests can be beneficial to the client and to the perception of the advisor as
customer-service oriented. Therefore, queries that take "a long time" were queries where the search for the solution could have been more direct.

**Fitting Problem-Solving Theories to Technical Support**

At its most basic level, problem solving involves two cooperating subprocesses: understanding and search (VanLehn, 1989). It was clear that the understanding process continued throughout the technical support interaction. In nearly every query recorded, the first coded utterance was "Client Description" where the client described what the problem was. Thus, every interaction began with part of the advisor's understanding process. Then, the advisors continued to add to their understanding throughout the rest of the query. As presented in the previous section, 27.8% of advisor utterances in structuring calls were questions, and 30.4% of client utterances in the same queries were descriptions. Thus, information gathering by advisors and information giving by clients were both very big parts of these interactions, taking nearly 1/3 of each person's speaking turns. Besides the initial exchanges where the client gave an overall description of the problem, it was virtually impossible to separate information gathering and search in these interactions. Table 8 shows an example. Almost all advisor questions shown here were examples of information gathering. Some were also examples of search – of testing hypotheses.

**Table 8: Edited excerpt from a query. Information gathering and search.**

| [1] | C: I've been working in Painter and sometimes when I lay everything out here on the screen the actual printout, some of the type, some of the boxes and things I create don't show up the same and they're all off-line. |
| [2] | A: What does it look like when you go to the print preview? Click on File and Preview. Does it look fine there? |
| [3] | C: Hang on just a second. Let me get to the right size. Actually, it is OK in the print preview. |
| [4] | A: What kind of printer are you printing out to? Is it a DeskJet or HP LaserJet Printer? |
[6] A: And you're going through Windows 95?
[8] A: What version of Painter are you using, 4.1 or 4.0?

In the typical query excerpted in Table 8, the client began with a description (line [1]), which was purely information giving to the advisor. Then the advisor asked four questions, each of which could be considered evidence of both hypothesis testing and of information gathering. The advisor's first question (line [2]) was whether the printing problem showed up in the on-screen preview of the printout. This can be taken as evidence of a hypothesis — "maybe the cause of this trouble effects the imaging of the printout and not just what the printer does" — and thus the advisor was searching for a solution. However, she also was gathering information. By asking this question, she learned that the trouble did not affect the preview, and so she could eliminate hypotheses of causes that would affect both the preview and the printout. However, she knew that there were some problems with older versions of Painter that affect the printing on certain printers and with more recent operating systems, so she asked her next three questions (lines [4], [6], and [8]) in order to confirm that the problem could be due to using older versions of Painter. Her questions were hypothesis-driven, but if her hypothesis proved false, the information she collected here could be used to confirm or disprove any number of other hypotheses she developed later.

So technical support structuring queries fit problem-solving models at their most basic level in that there is an understanding process that cooperates with a search process. The search process consists of searching for the best hypothesis by developing and testing a series of hypotheses. The understanding process consists of gathering information. Each process drives the other, and the processes overlap. When an advisor has a hypothesis in
mind, she asks questions to confirm or disprove that hypothesis, and each of these questions provides information that can be used in the pursuit of any hypothesis. Conversely, information gathering helps drive the search process, since any information gathered can help the advisor develop new hypotheses.

How do specific problem-solving models fit technical support? A combination of different models seems to fit the data well. As described in the introduction, there are two kinds of search: search in the problem space, as in information processing theories of problem solving, and search for a representation. Search for a representation is usually not referred to as a search process explicitly, but is called "selecting a schema" or "attempting to restructure." Technical support advisors do both types of searching. The process of searching the problem space in information processing theory is to go from problem state to problem state. This process can also be done internally, so that a problem solver can "look ahead" at possible states. In this case, the term "knowledge state" is used in place of "problem state" to indicate that there is no actual physical change in state as the problem solver looks ahead (Ohlsson, 1984b). This internal search process occurs when a problem solver imagines making moves in the Tower of Hanoi problem but does not actually make any physical changes.

Much of the information-gathering process in technical support can be viewed as a series of knowledge states or as a search through the problem space for an understanding of current state of the machine. For example, in the dialog shown in Table 9, the client explained the basic problem in lines [1] through [9]. Then, the advisor asked a long series of questions to clarify the exact setup of the file.
Table 9: Edited excerpt from a query. An involved information-gathering process.

[1]C: First of all, in the old version, I was using a text field to indicate whether a person attended a session or didn't, a training session or didn't.

[2]A: What were the values in that field?

[3]C: There weren't any values. It was just a straight enter. What I always used to enter was either a "1" if they showed up that night or a "-" if they didn't show up or a "Y" if they made it up because I knew in the old version a "1" and a "Y" acted like a one.

[4]A: That's right, a 1 and a Y both, that's correct.

[5]C: But in the new version it doesn't seem to do that.


[7]C: It's counting it as a zero. In other words, it's as if I didn't type anything in there at all. So when it sees "Y" in a field where there used to be a "Y" before, it would count it as "1..." The reason it's so important to me is because it totaled into a calculated summary field and I used that score for other purposes in my reporting, all right?


[9]C: Now, what's happening is when a person, when I type in a "Y" because they've made up that session, it doesn't change their bottom score.

[10]A: What are the names of these fields?

[11]C: S1, S2, S3...

[12]A: They're all text fields or are they number fields?

[13]C: They're all text fields. They have to be text fields if I'm typing a "Y" and a "1" and a "-".

[14]A: You can type a "Y" into a number field...

[15]C: Wait, let me check. They're all text.


[18]A: And then you have total fields for S1, S2, and S3, right?


[20]A: So let's call this one S1 total.

[21]C: Well, no. So I would have session total which would be S1, +S2, +S3, +S4, +S5, equals 6, or whatever.


[23]C: Follow me?

[24]A: S1, S2, S3, S4, S5, okay and then there's a total field called...


[26]A: Total. Okay, let's see, S... and that's going to be a calculation field so it'll be S1+S2+S3+ and so on.

[27]C: Correct.
Most of the advisor's questions shown in Table 9 provided her with further information about exactly what the client had done. Therefore, it was information gathering and not an attempt to prove or disprove a hypothesis. However, it is possible that any one of the advisor's questions could have elicited an unexpected answer that would have shed light on the cause of the problem. This happened in another query the client was having trouble when he exported a file from Painter on a PC and wanted to read the file on a Macintosh computer. This is shown in Table 10. This excerpt was preceded by a long discussion about exporting from a PC.

Table 10: Edited excerpt from a query. Unexpected answer during information gathering.

[A]: What program are you taking it to in the Mac?
[B]: Painter, it should be a piece of cake.
[A]: Because if you're taking it to Painter on your Mac, you really shouldn't have a problem taking the Painter file, without exporting it to anything else.

In Table 10 the advisor got an unexpected answer that changed the problem space dramatically – the long process the client was going through was unnecessary, and he should have just saved the file normally rather than exporting it. So, the problem could be solved using information gathering only. The advisor did not need to develop any hypotheses to drive the questioning process; by simply collecting information the advisor was able to reach the solution serendipitously. The search through the problem space can sometimes lead directly to the solution.

So, technical support queries can be forced into the information processing model. The advisor asks simple questions, gradually adding to her arsenal of knowledge about the situation. Sometimes she stumbles into crucial pieces of information that can easily lead to a solution, but the questions are only meant to make small steps through the problem space. The
advisor does not necessarily have a strong hypothesis in mind but, by gradually adding details, she can reach a solution by becoming familiar with the situation. However, although I can map these processes onto the basic search process described by Newell and Simon, the heuristics described by Newell and Simon do not apply to technical support. There are no examples of difference reduction or means-ends analysis in the 63 structuring queries, mostly because the heuristic does not make sense in this context. There is no way to make the current state "look like" the goal state because the states are not physical, and because the goal state, while known (make the file print, make the software interpret "Y" as "1", make the Mac read my PC file...) is not completely specified. Everything about the goal state is not known any more than is everything about the current state. In the Towers of Hanoi problem, the difference between the current and goal state is obvious – the rings are not on the correct peg, and they need to be on that peg when the goal is reached. The problem solver searches for a way to move from the current to the goal state. In technical support, the problem is to find out what the difference between the current and goal state is. Eliminating that difference may be trivial. For example, if a computer won't print because the printer is not in "ready" mode, the current state is that the advisor knows that the client has issued the print command but that nothing has printed. The goal state is to make the file print. The advisor might not have considered what mode the printer is in. However, as soon as the advisor discovers that the printer is not in "ready" mode, the difference between the current and goal states is found. Eliminating that difference (pressing the "ready" button) solves the problem.

Therefore, although the information-processing approach can be used to describe some of the advisors' actions, such as searching through a problem space by moving from knowledge state to knowledge state, most of the
important problem-solving heuristics described in that theory simply do not apply to the kinds of problems computer advisors face.

When advisors solve problems using information collecting only (as in Table 10) they don't seem to be using heuristics; they just move from knowledge state to knowledge state filling in details as gaps in their knowledge become apparent. Sometimes they discover a solution serendipitously. If they don't, the information they have collected eventually leads to a hypothesis. Many of the advisors' questions were clearly hypothesis-driven. For example, in one query, the client was trying to print a report from a database but the "subsummary" that he wanted was not appearing on the printout. One of the requirements of the software was that the database had to be sorted in order for the subsummary to appear. Therefore, one of the first questions the advisor asked was "Did you sort the database?" When that hypothesis proved incorrect, she formed the hypothesis that the "objects" that displayed the subsummary might be set not to print. She directed the client to the screen where this setting was turned on and off and asked "There's a box at the bottom that says, 'Do not print the selected objects. Is that checked?'" These questions affected the advisor's knowledge state by adding information – that the database was sorted, that "Do not print the selected objects" was not checked, and so on. However, the advisor did not ask these questions simply to gain more information about the problem or to move slowly through the problem space. She had hypotheses about what caused the subsummary not to print. In other words, she had an explicit idea of what the difference between the current and goal states could be.

In order to formulate these questions, the advisor had to create a representation of the problem space. She initially gathered some basic
information about the problem, and then represented the problem as "may not meet the basic requirements for subsummaries to work." So she checked whether the database was sorted, one of the basic requirements. In checking, she learned several pieces of information that brought her to a new knowledge state. She learned that the client believed that database was sorted, she learned what key was used to sort the database, and she confirmed absolutely that the database was sorted by instructing the client to sort it again. Then, she switched to a new representation of the problem. She represented the problem as "file may be set so that the objects we want to print are set not to print." Then she asked the client to check whether or not those objects were set to print.

The two hypotheses the advisor tested represent schemas – knowledge structures that the advisor had developed through previous experience and was attempting to fit to the current situation. The advisor stopped simply collecting general information to aid understanding and started working from a certain representation of the problem. Conversation transcripts don't usually reveal exactly when the advisor develops the representation, and it is not always clear whether a particular question that an advisor asks is schema-driven. However, many questions clearly reveal that the advisor is working from a hypothesis, because the advisor describes the hypothesis to the client. An example is shown in Table 11.

Table 11: Edited excerpt from a query. Advisor explains hypotheses, narrows hypothesis frame.

[1]A: Okay, let's see here...basically, in the illegal operation error that you're getting, usually it's going to give you an option to view details on that error.
[3]A: Can you get to that? Or do you know what those details are?
[4]C: Painter caused a general protection fault in module GDI.EXE. You want more?
[5] A: No, that tells us a lot right there. Now this is occurring in Windows 95?
[7] A: Okay, GDI is basically, your basic interface of Windows. Usually relates directly to your printer and your video. Have either of those been changed recently? Maybe change perhaps the type of driver that you're using...
[9] A: For the printer. And what had you just changed to?
[10] C: Well, I changed to...The intent was to go to a plain vanilla printer so I just chose the Epson FX80. And that's now the default printer.
[11] A: Okay, let's see here. What were you using before?
[12] C: We were using an HP DesignJet 755. The other end of the spectrum.
[13] A: Right. Okay, let's just check this out. This is probably going to be where our problem lies. Let's go ahead and from your start menu go to settings, choose printers...

In the query excerpted in Table 11, the client told the advisor that he got a "general protection fault" error (not shown in the excerpt). The advisor was not able to construct a hypothesis based on that information, so her first question (lines [1] and [3]) asked for more information. She told the client to "view details." This instruction was a knowledge-gathering one that moved the advisor from one knowledge state to another. Once she found out the details on the error message, the advisor developed a hypothesis – the client changed a driver, and that was causing problems (line [7]). Her next several questions and instructions helped gather information on that hypothesis. "What was changed? ... What did it change to? ... What did it change from? ... Let's look at the driver more closely." When this hypothesis developed, the advisor's representation of the problem changed from "some kind of system error" to "a conflict with a driver." When the client said that he had changed the printer (line [8]), the representation was refined to "a conflict with the new printer driver." The "conflict" representation had a procedural schema associated with it – change the driver back to what it was before to see
whether the problem disappears. If so, get an updated version of the printer

driver you want to use. This was the schema the advisor followed in the next

portion of the query. This process also follows Gick's (1986) model of problem

solving (see Figure 2) in which a representation is constructed, and, if it

invokes a schema, that schema is followed.

So, the computer advisors seemed to collect information in order to go

from knowledge state to knowledge state, and from this they developed

problem representations that gave rise to hypotheses. These "representations"

however, are still rather vague. The various faults technical support advisors

encounter can be described as frames. A frame for computer problems has the

slots "symptoms/expectations," "propositions," and "inference/causes."

"Symptoms/expectations" is related to the advisor's symptom knowledge,

where symptom knowledge consists of rules that associate symptoms to non-

obvious symptoms or causes. "Propositions" is related to the advisor's system

knowledge, and the slots are filled with information about how the system

works. "Inference/causes" uses the system information to describe how the

symptoms arise. Table 12 gives an example of one frame.
Table 12: A hypothesis frame for a computer problem.

Symptom/Expectations slot (If this frame is the correct one, these symptoms will exist.)
- The sub-summary is not printing.
- The header and footer are printing.
- The subsummary does appear in the preview of the printout.

Propositions slot (This system knowledge is relevant to this hypothesis frame.)
- Subsummary is composed of multiple records.
- The preview does not include options that are selected in the print dialog box.
- The print dialog box allows the user to choose "print only current record."

Inference/Causes slot
- The "print only current record" option is selected in the print dialog box.
- The preview is different from the printout because of the selection in the dialog box.
- Because only one record is selected to print, no subsummary can be calculated.

Govindaraj and Su's (1988) model (Figure 10) of the use of symptom and system knowledge to aid in frame selection applies to many technical support queries. The process of collecting information from the client – of moving from knowledge state to knowledge state – can be seen as the process of developing a set of known symptoms. The advisors asked questions until the set of symptoms could be matched to a rule in the symptom knowledge base. When the problem was not familiar, they relied more heavily on system knowledge to try to determine possible causes for the set of symptoms. For this they used novel frames, or adjusted known frames to fit idiosyncratic details of the particular situation.
Figure 10: Govindaraj and Su's model of fault diagnosis.

**Recognition of a Familiar Problem Based on Symptoms**

The use of symptom knowledge is most obvious in cases of routine problem solving. A frame is immediately activated because of the familiar symptoms. In the present study there were several routine queries where the advisor immediately recognized a set of symptoms that matched a frame. In these the understanding phase was very short, and a frame was activated right away.

At least ten of the structuring queries appeared to be easily solved by matching familiar (but non-obvious) patterns of symptoms to the advisor's memory. For example, crashing on startup indicated a corrupt font in two cases, and in each case, the advisor tested the hypothesis by locking a font file and starting the program again. In another "familiar" problem, a document printed so that it was much smaller than it appeared on the screen. The advisor knew about the problem and provided a work-around. In two other cases, an error message indicated that in order to print, the user would have to change a control option. The advisor recognized this error message from
previous experience, and told the clients that really their documents were corrupted and needed to be repaired. In each of these cases a brief description of symptoms brought a frame to mind. Table 13 shows an example of the "corrupt font" problem.

Table 13: Edited excerpt from a query. Immediate invocation of a familiar frame.

[1]C: I have Database II version 4.1 and um.. you want the serial number?
[2]A: Not just yet. What uh...
[3]C: Ok well, I installed it on... I have two machines here, one of which is a 7200 PowerMac and the other is an old Mac II running System 7.1.
[5]C: And um what's going on is that I installed this thing for the PowerMac and it works nicely and all that, but I also need it running on the Mac II so, but - the old 800K disk drives - so I installed a version onto the PowerMac that is to run on any Macintosh and then using Appletalk I transferred it over to the Mac II...
[7]C: And then I tried to launch it and while it's trying to build a font menu – I have a lot of fonts – it gets part way through the list of fonts and it tells me it has a bus error and crashes. So what the hell am I doing wrong?
[8]A: I don't know that you're doing anything wrong. Usually when it crashes while it's trying to build the font menu that indicates that there's a corrupt font.

Working Directly from Surface Features: Use of System Knowledge

Many problems were not "routine" but the symptoms easily gave rise to a hypothesis. One example of these is excerpted in Table 11, where the advisor used the error message as the basis to pursue the issue of changed drivers in the system. In another case, a client indicated that he couldn't open a file from a floppy disk and that he got the error message "cannot read from this disk." The advisor recognized that the floppy disk was damaged and explained how to try to recover the data. The distinction between "surface feature" hypotheses as described here and "familiar" hypotheses as described
in the last section is the knowledge base the advisor draws from. In familiar or routine problems the advisor recognizes symptoms that are somewhat odd and must be learned from experience with these particular software packages; they are almost all "bugs" that may give rise to "false" error messages. In "surface feature" recognitions, the advisor uses her general knowledge about computers and works directly from symptoms rather than prior experience with bugs. This distinction is a blurry one, and in most cases (and as is described in the Govindaraj and Su model) advisors use both sources of knowledge to identify a frame.

**Extending the Models**

The frame model does not completely explain how troubleshooters change frames or how they move between them. Govindaraj and Su found that troubleshooting was hierarchical. The troubleshooter would start off with a high-level frame and become more specific. For example, the original hypothesis may be "something with the boiler" and as the troubleshooter reasons through the symptoms the hypotheses become more specific. When the hypothesis is specific enough to check, the specific hypothesis will be proven correct or incorrect. If it is proven incorrect, the troubleshooter simply moves back up the hierarchy to the more abstract hypothesis again. Therefore, the troubleshooters of the physical system Govindaraj and Su studied stayed within one high-level representation of the problem. They moved up and down the hierarchy of their mental model of the system, but their view of that model stayed the same.

Gestalt theory says that problem solvers sometimes have to restructure their mental representations of the problem space in order to generate new hypotheses. The frame model does account for this in that the restructured representation would still take input from the problem solver's symptom
and system knowledge bases, but the model does not treat restructuring situations as special. The frame model implies that the processing of frames continues in an orderly manner, not with large switches to new types of hypotheses.

Levine's transfer hypothesis, as described earlier, predicted that problem solvers would pose hypotheses from the same domain until that domain is exhausted. Once the domain is exhausted, the problem solver is forced to take a different view of the problem by moving to a different domain. This is in the spirit of "restructuring" from the Gestalt tradition. The frame model as depicted in Figure 10 generally assumes that all frames are within the same hierarchy – within the same domain. In this model the system knowledge remains constant. One way to depict a change in frame domain is to allow multiple representation of the system knowledge. This way, when a technical support advisor restructures the problem space, she retains the same symptom set, but she views the set from a different perspective, with a different set of relevant system knowledge.

Gick's model of problem solving (Figure 2) that combines schema and information processing theories allows for two ways of changing the hypothesis. If an attempt to implement a schema fails, the problem solver either searches again using the same representation of the problem (same-domain hypothesis) or returns to the top of the model and selects a new representation (new domain/restructured hypothesis). Ohlssen's model as depicted in Figure 3 also allows for this. A search in the description space is an attempt to restructure or look for hypotheses in a new domain, while search in the problem space looks for solutions in the current domain or representation of the problem. A model that explained the process of problem solving by computer advisors would combine the concept of searching
through a problem space by collecting information, the concept of changing
representations of the space from schema and Gestalt theories, and the
concept of drawing information from multiple sources to fill slots in
hypotheses frames from Govindaraj and Su's refinement of frame theory.
Table 14 shows a detailed example of moving through several frames.

Table 14: Edited excerpt from a query. Advisor moves through several frames.

[1]C: Why can't I get Painter and Writer running at the same time?
[2]A: That shouldn't be a problem. You should be able to. What Mac
are you working on?
[4]A: And how much RAM does that 580 have?
[6]A: In the Finder, go to the Apple Menu, and go to About
     Macintosh. And what is the...
[7]C: Total memory is 20480K.
[8]A: And what's the total largest unused block?
[9]C: 17,007 and it's fluctuating. 17,744.
     What's the system software on that?
[12]A: OK. Let's close that up. Could you go to your Memory Control
     panel...
[13]C: uh...memory
[14]A: I'd like to see what the memory cache settings are?
[17]C: My virtual memory is off, my RAM disk is off.
[18]A: There also should be something called "32 bit addressing" in
     your control panel.
[19]C: In this particular control panel?
[20]A: I believe the 580 has a 32 bit, let me just double check.
     ...
[21]A: Here's what I'd like to try. We've got your memory control
     panel set up OK. Let's close that out.
[22]C: Yep.
[23]A: I need to you to go to the application folders for each, Writer
     and Painter, and we'll do one at a time.
[24]C: Applications now open, Painter?
[25]A: I would like you to open the Painter folder, highlight the
     Painter application itself.
[26]C: Okay, Painter yup.
[28]C: <beeping> OK, what do you want to know?
[29]A: OK. Down at the bottom right hand corner, it says Memory
    Suggested Size...
[31]A: What is that asking?
[32]C: 1000K.
[33]A: What is the minimum?
[34]C: 800K.
[35]A: What is the...
[36]C: Preferred? 1200K.
[37]A: Now, close that window.
[38]C: OK.
[39]A: I want you to do the exact same thing with Writer.
[40]C: Okay, Writer 4, okay now it comes up by, okay I'm going to
    view by icon...
[41]A: While you're looking at Writer 4.0, there's a line in that
    window for version. Would you also tell me that, when you get
    that window open.
[41]C: OK, 4.0 v1.
[42]A: v1, okay
[43]C: Suggested size is 1400K. Minimum size is 1000.
[44]A: OK. Would you increase Writer memory to at least 1400K.
[45]C: The minimum size?
[47]C: OK. The preferred size is at 1400.
[48]A: Go ahead and increase it to 2000K.
[49]C: OK.
[50]A: You have 20 megs, so having both applications open at the
    same time should be absolutely no problem.

In the interaction in Table 14 (which has been edited for brevity) the
advisor used her symptom knowledge base to immediately identify the less
specific "out of memory" frame from the client's question in line [1]. She then
moved to the use of her system knowledge base to identify the various ways a
system could be out of memory. The advisor asked one general information-
gathering question in line [3], and then began to examine the "computer does
not have very much RAM" frame by asking the question in line [4]. The
client's response disproved that hypothesis, but the advisor pursued it,
again disproved that frame. Govindaraj and Su then predict that the advisor
should move to a higher level of abstraction, which she did. She dropped the "not much RAM" hypothesis but remained in the less specific "out of memory" hypothesis frame. Lines [12] through [15] explored the specific "memory" hypothesis frame, which was again disproved. The advisor then moved up to general memory problems again, and explored a third specific memory frame in lines [18] through [20] and a fourth specific frame in lines [21] through [49]. All of this exploration is predicted in frame theory. However, in line [50], the advisor gave up on the memory frame altogether. She had exhausted the domain of "memory problems" hypotheses and needed to identify a new hierarchy of hypothesis frames.

In the query that continued after the portion quoted above, the advisor asked the client to restart the computer and then go through the process of opening files in each of the two programs. This was a form of information gathering. At a certain point during this information-gathering session, the advisor seized an entirely new representation of the problem. She no longer saw it as a problem of not being able to open both programs at once, but reconstructed it as a problem wherein the computer could not open files that were created by Painter. The next few hypotheses she tested involved looking at the files and trying to re-associate them with the Painter application.

In general, technical support interactions follow the following pattern. The query begins with information gathering. The advisor sometimes discovers information that immediately leads to a solution. Most of the time, the advisor uses the information she has gathered to select a frame. Often this frame is very general and actually represents a domain of hypotheses. I'll call this a frame hierarchy. Once the advisor has selected a frame hierarchy, she further specifies the frame to formulate a concrete, testable hypothesis. Some frame hierarchies contain only one hypothesis, so the specification is not
required. The advisor tests the concrete hypothesis. If the test is successful, the problem is solved. If the test fails, the advisor either moves back up the frame hierarchy and re-specifies another concrete hypothesis in the same domain or, if the hierarchy is exhausted, she either immediately selects a new frame hierarchy (she changes domains or constructs a new representation of the problem) or she gathers more information so that she can find a new frame hierarchy. This prototypical process generally follows the existing problem-solving models.

There are some exceptions that occur in technical support that do not follow the models. Occasionally an advisor would simultaneously consider two unrelated hypotheses — two hypotheses that came from different representations of the problem space. Or, sometimes multiple solutions were suggested without testing the first hypothesis. In other words, the advisor would suggest multiple sets of procedures a client could try without having the client actually implement those solutions sequentially during the conversation. In these cases the advisor would work from several specific frames from the same hierarchy at the same time. An example is shown in Table 15.

Table 15: Edited example from a query. Advisor keeps two familiar frames in mind simultaneously.

<table>
<thead>
<tr>
<th>1</th>
<th>A: Have you reinstalled Painter at all?</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>C: No.</td>
</tr>
<tr>
<td>3</td>
<td>A: That might be a good way to proceed with this. The other thing that occurs to me is that if there were more than one copy of Painter or more than one copy of the SHSC Folder in the system...</td>
</tr>
<tr>
<td>4</td>
<td>C: Well, there might be on another drive buried, but it's not in what you would call a System Folder.</td>
</tr>
<tr>
<td>5</td>
<td>A: OK...</td>
</tr>
</tbody>
</table>
[6]C: I had a PowerBook. Also, this is my desktop at home and I just threw all my System Folder stuff into another folder, because there's always things I ended up not having but it's not labeled a System Folder anymore. Could it still find it somehow?

[7]A: It's possible. Let's go back to the Finder and let's do a Find on just the word "SHSC." See how many SHSC Folders you have.

In Table 15 the advisor suggested one way to proceed on line [1]. This suggests that she was in the frame "application files corrupted or incomplete" that can be solved by reinstalling the software. However, without testing this hypothesis (which would be done by reinstalling Painter), she suggested two more hypotheses (line [3]): there were two copies of Painter or there were two copies of the SHSC folder. This advisor not only iterated through related frames, but also kept several hypotheses in mind at once. Problem-solving models don't account for this behavior.

Another missing detail from most problem-solving models is that sometimes accumulated evidence will cause the advisor to restructure the problem without exhausting the current domain. Table 16 is from the same query as Table 15. In this excerpt, the advisor was still working on hypotheses related to file conflicts.

Table 16: Edited example from a query. New information causes advisor to switch domains.

[2]C: SHSC TRANS System?
[3]A: Yeah. SHSC is one word, TRANS is another word, and then System. It should find only one.
[4]C: OK. Well, it's finding some on...
[5]A: Can we have it search on just your startup drive?
[6]C: Now in that same SHSC Folder, it has a date of 7/20/92. Last modified. That's the one I just trashed. So I'll get rid of both of them.

[7]A: That's bizarre! Have you run Norton on this recently? Do you use anything like that?
[8]C: I haven't run it recently. Yeah, I've got it.
[9]A: It's usually a bad sign if dates keep shifting around like that. I don't know that that indicates a problem, but in the past when I've see dates shift around, it usually suggests there's a system level problem.

In Table 16 the advisor was in the middle of testing her hypothesis that there might be two versions of the file "SHSC TRANS System" when she discovered some information that surprised her; the file that the client had already thrown away had reappeared (line [6]). She expressed her surprise at this information ("That's bizarre!" line [7]) and moved to an entirely different frame: system-level problems. Thus she restructured the problem not when she exhausted a domain, but when she discovered surprising information.

Behavior such as changing hypothesis domains without exhausting the current domain or posing several related or unrelated hypothesis simultaneously are not completely omitted from problem-solving theories. Ohlsson (1984b) suggested that problem-solvers may restructure the problem space "on novelty" in addition to "when stuck." Studies of hypothesis generation implicitly assume that problem solvers simultaneously entertain multiple hypotheses when they ask subjects to create comprehensive lists of possible hypotheses (e.g. Mehle, 1982). However, most comprehensive models of problem solving assume only one frame or frame hierarchy is considered at a time, and that people only change their perspective of a problem only when they are stuck.

**Technical Support Strategies**

If the advisor could not think of a frame to apply using system or symptom knowledge, what did she do? If no frame was most salient, where did she start? What strategies did she use for information gathering? Advisors used a number of different strategies, outlined below.
Create More Surface Features

If the advisor did not recognize a schema from a client's initial description of the problem, she gathered more information on the problem. One strategy was to have the client go through the procedure he used when he ran into the problem, and look for something to trigger a hypothesis. This is a search for simple solutions as well as a way to gain more information that can lead to non-simple hypotheses. Often a client error became obvious during this process and invoked a schema that could be used to correct the problem.

Eliminate High-Level Frames

Another heuristic advisors used was to eliminate domains of hypotheses and then focus on those not eliminated. This heuristic is similar to the split-half algorithm in which whole sections of possible faults are eliminated at once. There were a number of methods advisors used to eliminate hypothesis domains. These included: a) on a Macintosh computer, restarting without extensions limited the hypothesis frame to extension problems or eliminated all extensions as problematic, b) creating a new file or new layout limited the problem to that file or showed it was something larger, c) starting in Windows 95 "safe mode" selected or eliminated the domain of problems with drivers, d) replicating the problem on the advisor's own machine could indicate if the problem was caused by a bug or if an upgrade was needed. This was not usually trusted to completely eliminate any hypotheses because the advisor may have made the same simple error. But if the error was not replicated, the advisor could usually infer that there was a user error or that an upgrade was needed, e) sometimes advisors looked for boundaries of the problem space. The client might only have noticed only one of many symptoms. If the problem was more
widespread, the domain was more general, and the advisor moved to a frame of higher abstraction or to a new frame altogether.

**Other Heuristics**

Advisors used several other heuristics to help them identify frames. One method was to attempt to identify unique differences. For example, if a client had three computers that were set up similarly to one another and the problem only existed on one of the computers, the advisor tried to determine unique attributes to the nonworking machine: different video boards, extra system software, and so on.

Another heuristic was to have the client read entire screens worth of information, looking for something wrong. For example, clients were asked to read aloud all the control panels, read all the options on print setup, or read everything in the "get info" dialog box. In these cases, the advisor usually had a high-level frame in mind (something with extensions, something in the print setup) but couldn't narrow the frame down easily. Asking the client to read the information seemed to help the advisor recognize potential trouble spots.

A final heuristic was simply to start over from scratch and hope problem didn't reappear. Some solutions advisors proposed were to copy all the text from a troublesome document and paste it in a new document or to reinstall the system software or application software to see if the problems would disappear. In these cases the advisor gave up on finding the cause of the problem because it was more important to find a solution. This shows that in technical support, the frame slots were not always completely filled in. The cause slot was frequently filled with very vague information that pointed directly to a solution. For example, problems on a Macintosh that involved files losing contact with their applications, and many other unexpected
system behaviors fell into the "desktop file messed up" frame. Many parts of
the computer were treated as "black boxes" in which details were
unexamimable. "Desktop file messed up" is a very general cause, but it has a
straight-forward solution: rebuild the desktop. The existence of these black
boxes made the advisors' frame structures very flat. The frame hierarchies fit
well into Levine's domains of hypotheses because each hierarchy usually had
at most only two levels. The top level can be viewed as the domain, and the
lower level as the set of hypotheses in a domain.

Social Influence

Social influences such as client expertise had little influence on
problem-solving strategies that advisors used. On the whole, I did not find
patterns of strategies where advisors began with simple solutions for novice
clients and complex solutions for experienced clients. Social influences had
greater influence on indexing- and defining-role queries, in that advisors
sometimes gave procedures or explanations to satisfy novice clients. For
example, when there were several ways to accomplish the same goal novice
clients were usually given the least-expensive but easier to apply procedures.
Also, in all kinds of queries, advisors adjusted the way they gave instruction.
For example, an experienced client was told "throw the file away," while a
novice client was told, "open the folder, click on the file, and drag it to the
trash."

In structuring queries, social factors had more influence on the
information process than the search process. The length of the information-
gathering process depended more on the ability of the client to describe the
problem in a way that gave the advisor the information she needed to start
looking for a solution than it did on how complicated the problem was.
However, the client did occasionally affect the order of hypothesis testing.
Table 17 shows four excerpts from a single query. In this query, the advisor generally followed the technical support procedures described in this paper, but the client also provided a number of his own hypotheses. These hypotheses were not related to the other hypotheses the advisor was testing, and they did not tend to occur while the advisor was gathering information to develop new hypotheses. The client tended to suggest his hypotheses right after tests of previous hypotheses failed, the same time that advisors tend to move to new hypotheses. Each of these suggestions was a kind of interruption to the advisor's reasoning process because the advisor did not believe that any of these client-suggested hypotheses would be correct. They were not severe disruptions, though, because the client waited for appropriate moments to suggest them, right after failed tests. Table 17 shows four of these "interruptions" and the advisor's reaction to them. It does not show the advisor's line of reasoning, for space considerations.

**Table 17: Four edited excerpts from a query. Client provides hypotheses.**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]A: It should have opened up a dialog showing you the choices that you have.</td>
<td></td>
</tr>
<tr>
<td>[2]C: Hm, it doesn't. Oh, do I have to launch on my sharing, the one that's sharing it on now that I've indicated it's public? Do I have to relaunch?</td>
<td></td>
</tr>
<tr>
<td>[3]A: No, the one that's public is already launched as public. It's already open as public.</td>
<td></td>
</tr>
<tr>
<td>[4]A: Okay, which part of the... does the upper part of this window or the lower part have the bold border around it? I'm sorry; do this: click in the big white area that's underneath the words &quot;network access.&quot;</td>
<td></td>
</tr>
<tr>
<td>[7]C: Since I reinstalled something in the system folder, does that mean I have to relaunch the computer maybe?</td>
<td></td>
</tr>
<tr>
<td>[8]A: It shouldn't matter, but why don't we go ahead and relaunch the computer and we'll rebuild the desktop file as well.</td>
<td></td>
</tr>
<tr>
<td>[9]C: It's up and running. Now I got to call back on the network to get into this other computer first though, under chooser, right?</td>
<td></td>
</tr>
</tbody>
</table>
[13] C: I'm not linked up to the other computer though.
[14] A: But you are connected to the other computer with cable...
[16] A: And the other database is set to be shared. Database II uses it's own networking protocol.
[17] C: Oh really.

[19] C: For some reason I can bring my computer up on number 2 but I can't bring number 2 up on number 1. I don't know if that has anything to do with it.
[20] A: Well, okay, we can...
[21] C: Maybe we should fix that problem first, maybe that is interfering.
[22] A: Okay, let's look at that.
[23] C: All right.

The query that Table 17 is taken from involved a problem in which the client had the Database II software on two computers and wanted to open the same file on both computers but could not. In line [2] the client confirmed that the advisor's last test failed, and then he suggested a very different but simple hypothesis: the program needed to be restarted. The advisor explained why that hypotheses was incorrect, and continued with her previous line of reasoning. In the next excerpt, in line [7] the client again suggested a relatively simple solution (restart the computer) after the advisor's last hypothesis failed. The advisor again explained that this hypothesis was incorrect, but told the client to go ahead and test the hypothesis anyway. The advisor added another step (rebuild the desktop) that would test a hypothesis of hers at the same time. The rebuilding the desktop frame was a change in perspective from what the advisor was testing immediately before the interruption, so the client may have influenced the advisor's line of reasoning by interrupting.

In line [9] the client suggested another hypothesis that interrupted the advisor's line of reasoning. The advisor seemed certain this hypothesis was
incorrect and rejected it out of hand. The client was doubtful and pushed his again hypothesis in line [13]. The advisor followed the client's frame by testing a related very simple hypothesis in line [14] and then explained why the client's hypothesis was wrong in line [16]. She then moved back to her line of reasoning. The client was still not convinced that his hypothesis was wrong, for he interrupted again in line [19] with the same hypothesis as in the previous excerpt. This time, the advisor followed through by fully testing the client's hypothesis, and it was proven to be incorrect. The advisor then carried on with her line of reasoning.

In the above example, the client made many suggestions of hypotheses – more than most clients did. On the whole, his suggestions seemed to make the problem-solving process less efficient because the suggestions he made were based on misconceptions. His suggestions were easy for the advisor to know immediately that they were incorrect. However, since the advisor (eventually) indulged the client's ideas and usually explained why his hypotheses were incorrect, the digressions were probably beneficial to the client in the long run. Also interesting is that the client timed his suggestions to the same model that advisors used. He almost always made suggestions after a test had proven another hypothesis false. So, even though he formed different hypotheses than the advisor would have, he suggested them at the same points in the interaction that the advisor would have. And, he stayed in one frame hierarchy when he suggested the same hypothesis twice (but in slightly different ways) in lines [9] and [19] of Table 17. This shows that the client, like advisors, tended to keep one representation of the problem until it was disproved, and did not tend to suggest unrelated hypotheses while remaining with that representation.
Clients often suggested credible hypotheses as well as incorrect ones. Table 18 shows an example where the client made suggestions that fit completely into the advisor’s hypothesis generation and test procedure. The advisor suggested a hypothesis, in line [1], that the fields the client was using had no data in them. The client immediately instantiated that hypothesis to a related frame. One reason a field could seem to have no data in it is that he was on the wrong record and the fields in the current record were blank (line [2]). The advisor suggested a more general solution (from the higher-level frame): the fields could be made global so that the current field would not be important (line [3]). The client then showed that the advisor’s solution was incomplete: the fields were already global, but they weren’t on the layout. This hypothesis was a second instance of the high-level frame of "no data coming from fields." The client reasoned his way out of his own instantiation of the frame in line [10], so the discussion again returned to the higher-level frame. Then, the advisor suggested a third instantiation: the correct fields were being read, but there was something wrong with the data (line [13]).

Table 18: Edited excerpt from a query. Client’s hypotheses fit easily into the advisor’s problem-solving process.

[1] A: Okay, so at least it’s not giving you a question mark. Have you set your start date and your end date, because if there’s nothing in there, maybe your start date and your end date don’t have any value in them.

[2] C: Oh, yes you’re right, I’m on the wrong record. You’re right. I forgot to be on the test record. Um…find…

[3] A: And also I would recommend, is it possible that start date and end date could be global fields?

[4] C: Uh yes, they are.


[7] A: If they’re global fields we should see them on all records.

[8] C: Yeah, that’s true. The thing is they are not on the same layout. I need to change...

[9] A: They do need to be on the layout.
[10] C: Well, actually not. Do they need to be on the layout? Because what needs to be on the layout are the dates I'm looking for but start date and end date are just input. Input in the field I'm looking for.

[11] A: So, those are global date fields? Start date and end date are global date fields?

[12] C: Yes, yeah.

[13] A: No, they should be fine, but let's go check to see what values we have in there. Let's go to the layout where they are.

So we see that both sophisticated, expert clients and less sophisticated, novice clients both suggested hypotheses, and these hypotheses fit into the advisors' prototypical troubleshooting process as to the types of hypotheses that were suggested (related ones suggested consecutively) and the times at which they were suggested (after failed tests).

**Did Advisors Test Simple Solutions First?**

The advisors seemed to try to look for simple solutions first, and they seemed to try to rule out easy-to-test hypotheses before harder to test hypotheses. However, it is impossible to quantify these results because there is no way to know the whole universe of reasonable hypotheses for a given problem. Advisors often ruled out simple and common hypotheses even when there was no clear evidence that those hypotheses could be true. For example, "extension conflict" is a very easy to test for Macintosh users. It also tends to cause unexpected symptoms, so it has the possibility of applying to all kinds of problems. This was a very commonly tested hypothesis in the queries recorded.

Advisors clearly put off hard-to-test hypotheses, often explicitly saying "I think we need to upgrade the software, but first let's try this..." In these cases the advisors deferred attempting solutions that took a lot of time, even if they thought they were correct.
Social pressure did not seem to dissuade advisors from testing simple solutions. Some clients occasionally took mild offense at the "obvious" questions advisors asked. However, they generally complied with the advisors' instructions. The risk of giving offense did not appear to make advisors avoid asking about simple solutions. Table 19 shows two examples from different queries where the advisors checked for very simple solutions immediately. Both excerpts are shown from the point where the client first explained what the problem was. In the first excerpt, the advisor made the suggestion that the error message was accurate, which was the most obvious simple solution (line [6]). She easily dropped this hypothesis based on the client's answer. In the second excerpt, the advisor suggested a simple hypothesis in line [14] at which the client took mild offense. However, in this case the advisor continued after this excerpt by asking the client to sort the database again, just to be sure the simple solution was incorrect. The willingness to quickly drop simple solutions without robust proof may be influenced by perceived client expertise. The advisor dropped the simple solution very easily in the first query but not the second. The client appeared slightly more novice in the second excerpt because of his comment in line [11] that indicated that he had needed a great deal of help from technical support to set up his database.
Table 19: Two edited excerpts from two different queries. Advisors check for simple hypotheses immediately.

[1]C: Well, all of a sudden my Database II database is causing some really funny little situations. I have a database where I have several reports generated and when I try to go today ... when I tried to go to the print preview I get an error message that says the combined size of the header plus the footer is longer than can fit on the page. To print, first make the header or the footer shorter or use longer paper.

[3]C: And I've never seen that before. It's never come up before and it just refuses to even get into print preview.

[5]C: ... and <another advisor> suggested try reloading the printer driver and also rebuild the database.

[6]A: Ok. Well first of all, OK, that's fine. Those are both good suggestions. Before we go that far let's go into layout mode. Take a look at the layout. How big is the header and the footer for that layout?

[7]C: Some of those reports don't even have headers and footers.
[8]A: Ok. Then it does sound like a corrupted database. It might be the printer drivers. That was a good idea. You reloaded that?

[9]C: Hi! I need to figure out why when I'm printing out my report, that it's not printing (laughs).

[10]A: OK, this is Database II?

[11]C: Database II 4.1, and I go into my print preview, it's all there, it looks absolutely "beautimous" and it's only taken me like two days to figure this out after I called you like twenty times.


[13]C: Now, it's printing my header and my footer and it won't print my summary that has all my information in there.

[14]A: OK, did you sort the database?

[15]C: It's sorted. I did that, I read the book, I did the whole thing. I did the troubleshooting just to make sure I had done everything right.

Einstellung

Despite the fact that advisors tried to look at simple solutions first, at least four incidents of Einstellung occurred. These four queries accounted for 19% of the 21 structuring queries that were resolved. The 42 structuring queries that were either unresolved or indeterminate may also have included some cases of Einstellung that remained undiscovered because no solution was found or the solution given was not completely proven. The minimum
rate of Einstellung was 6%, if all the indeterminate and unresolved queries did not have simple solutions. It is likely that at least some of those cases represented missed simple solutions. A conservative estimate would be that half of the indeterminate queries were in fact solved. Among the queries that I am certain of the solution (resolved queries), 19% were cases where a simple solution was missed. Therefore, if this same rate existed among the other half of the indeterminate queries and the unresolved queries, the total estimated maximum incidence of Einstellung was about 14% over all the structuring queries.

The four cases of Einstellung had very simple solutions that were missed in the initial troubleshooting process. These queries were classified as examples of Einstellung because the advisor and client explored several much more complicated tests and hypotheses before hitting on the simple source of the problem. Also, in each case the advisor expressed some disgust (after the completion of the phone call) at having missed such an "easy" solution.

Only one of the four cases took a very long time, just over 33 minutes. This made it the third-longest structuring query and the longest resolved structuring query. But in all, the four cases of Einstellung took 21% of the total time spent on recorded resolved structuring queries, only 2% more than their share. The other three cases were fairly short, but could have been much shorter if the simple solution had been explored sooner.

Social pressure (clients taking offense at elementary suggestions) did not keep advisors from testing the simple solutions first. But in two cases the apparent or real expertise of the client influenced the advisor to believe that the probability of these simple solutions being correct was extremely low. Also in these two cases there was an immediate hypothesis that seemed very likely to be correct (a red herring). This made the relative probability of simple
solutions even lower. When the first hypotheses were disproved, the advisors in these cases tended to stay with hypotheses of similar complexity, making them prototypical cases of Einstellung.

The third case resulted from a deficiency in the advisor's system knowledge. Although she tested several simple solutions early in the conversation, she did not consider a whole set of other potential causes because she believed that one of the symptoms made that set of causes impossible. The client himself finally suggested the solution, serendipitously noticing an unexpected setting of a control.

The fourth case is harder to diagnose. The advisor did not propose many hypotheses, but she spent a great deal of time trying to understand the intricacies of how the client had set up his file and to replicate the problem on her own computer. Eventually she asked a very basic question about the field type the client was using, and was able to tell him that the field type was incompatible with his goal. The cause of this problem may have been that the client was not able to fully articulate what his problem was and the advisor had a high cognitive load in trying to understand what the client was asking. Since she devoted so much effort to understanding how the client had set up his file, she wasn't able to step back and consider simple things that could go wrong.

Because two of the cases of Einstellung were caused by a combination of the influence of an expert client and a red herring, I describe one of these here in detail. The client was using the database program and wanted to open a set of files across a network. He had done this before, and all the files that he had set up to "share" across the network were still working. However, he had a set of four new files that he could not share. He asserted that the only thing
different about this new set is that they were in a folder. All the files that worked were mixed together at a higher level in the directory system.

The advisor seized on the idea that "the only thing different about these files is that they are in a folder" as stated by the client. She developed a hypothesis that there may have been some security settings on the folder, or the folder could be corrupted or damaged in some way. This is a fairly complex hypothesis. Since being in the folder was the only thing different, she wanted to take the files out of the folder and make them the same as the working files by putting them at the same directory level. Testing the hypothesis was fairly involved, however, because the client did not want to make the change the advisor suggested. During the process the advisor suggested several other hypotheses that were similar in complexity to the folder hypothesis. Each of these hypothesis were disproved.

Out of ideas, the advisor turned to simple solutions. She first confirmed that the client was using the correct procedure to open the files. Then, she asked whether the files were set to be shared. They were not. This solved the problem.

This query took just over eight and a half minutes. Setting the "sharing" option is the basic step to making a file sharable, and checking or fixing it is a matter of changing one menu option. The advisor did not begin with this hypothesis partly because the client was obviously knowledgeable and had shared files successfully before, and because he provided the red herring that there was only one difference between the working files and the non-working ones: the existence of a folder. Of course, there were in fact two differences, but the second one was not discovered until later. After the call was over, the advisor commented to me, "I didn't think he would have made
that mistake." This was a clear case of the client's expertise causing an advisor
to not to consider simple solutions.

**Summary of the Qualitative Analysis**

I have discussed three basic theories of problem solving and shown
how well they describe problem solving in technical support. Newell and
Simon's General Problem Solver dictates a step-by-step process wherein the
problem solver transforms the current state into the goal state. Each
intermediate step is required in order to reach the goal. Although some of the
information-gathering process used in technical support could be described in
these terms, the basic ingredients of this theory did not exist in the real-world
problems encountered by advisors. The real-world problem solving of
technical support involved a series of false starts: of ideas tested and then
proven false. Although information gained from the process can help in
finding the ultimate solution, the steps taken in pursuit of the false starts did
not move the problem state closer to the goal. Thus, most of Newell and
Simon's heuristics for problem solving do not apply to technical support.

The other two problem-solving perspectives better match the field data
described in this study. Schema theory, and especially the more specific theory
of hierarchical hypothesis frames as described by Govindaraj and Su,
accurately predicts a portion of the hypothesis generation and test process that
the technical support advisors demonstrated. Advisors used a combination of
symptom and system knowledge to select a frame hierarchy and then tested
hypotheses from that frame hierarchy. Schema theory indicates that when a
selected schema cannot be fully specified, the problem solver drops the
schema and reconstructs the problem space from another perspective.
Operationally, in technical support, reconstructing the problem space meant
reevaluating the set of relevant system knowledge.
The Gestalt perspective and Levine's theories that were derived from it focus on this reconstruction of the problem space. The Gestalt perspective holds that the problem solver restructures the problem space when reaching an impasse. Levine theorized that problem solvers sampled from a single hypothesis domain, and that they only changed to a new domain when they exhausted the first.

A new model of technical support problem solving is shown in Figure 11. The model is a combination of the problem-solving theories discussed above. Most of the 63 queries studied here can be explained with this model, but not all. This model describes the prototypical query, where the advisor suggests one hypothesis at a time, the client does not suggest hypotheses, and the advisor only switches to new frame hierarchies when the current one is exhausted.
Figure 11: A model of prototypical technical support problem solving. All queries begin in box A, with the client's description of the problem.

In Figure 11, all queries began in box A, with information gathering. At a minimum this consisted of the client's description of the problem, and sometimes this step included a very long series of exchanges of descriptions and advisor questions. Once the advisor had an initial understanding of the symptoms, she could either recognize these symptoms from previous experience or use her general system knowledge to construct a representation of the problem and select a domain to examine. Sometimes no frame was needed. The information from the client and the advisor's system knowledge were enough to identify a possible solution, as in Table 9. That example followed the path through boxes: A -> B -> C -> E -> I. When a frame was
activated, the advisor selected a frame hierarchy in box E. This was a top-level frame such as "a memory problem," "something with the printer" or "extension conflict." Sometimes this frame was specific enough to test immediately. If it was not, the advisor used her system knowledge to select a subframe that was testable. For example, a subframe of "memory problem" could be "computer doesn't have much RAM." The specific frame was then tested. Testing could be done by asking the client questions, having the client look at settings on the computer, or by attempting to carry out a solution. If the test failed, the specific frame selected was incorrect. Sometimes carrying out the test provided further information and could be considered an information-gathering process. Sometimes further information gathering was necessary. Or, sometimes the advisor would simply move back up the frame hierarchy and use system knowledge to select another hypothesis in the same domain.

The model is simplified to some extent in that information gathering was done at all phases, not just in forming new representations and in testing. However, much of this other information gathering was incidental details or "by-the-way" questions, such as the software version, computer model information, name of a particular file. This information was typically used by the advisor to tailor her advice. For example, knowing the software version helped her give the correct command names because they sometimes change between versions. Knowing the client's file names allows the advisor to say "open Thesis" instead of "open that other file."

Fitting the Model to Non-Prototypical Cases

The risk of developing a well-specified model of problem solving is that the model's developers tend to select the kinds of problems that exemplify the model. Newell and Simon's General Problem Solver was very
well-specified but fits only a very limited range of artificial problems. It does not do a good job of explaining problem-solving in the real world. The model shown in Figure 11 was mostly based on frame theory and Levine's hypothesis domain theory and, with some modification, it does a good job of explaining most of the real-world problem solving in technical support.

It does not account for the contribution of the clients, and it ignores cases where advisors keep two hypotheses in mind at once. It also ignores cases where advisors suggest a hypothesis but delay in testing it. It does not account for the strategies advisors use to generate or limit the list of possible frames. It does not account for changing to a new frame hierarchy because of novel information. These are the non-prototypical cases of real-world problem solving. The existence of these "exceptions" to a simplified model exemplify why it is so difficult to build an expert computer system to replace human support.

**Potential Technical Support Failures**

Precisely because technical support is so flexible, it has many points of potential failure. There are risks in both taking a client's comments too seriously and in not listening carefully enough to the client's description of the problem. Two of the cases of Einstellung described earlier occurred because of these risks. In the first case, the advisor trusted the client when he said that there was only one difference between the set of working files and the set that did not work. In fact, there were other differences, and one of these differences was the solution. In a second case, the client displayed a very basic misunderstanding of how the computer worked, and this was the root of his problem. However, instead of carefully listening to the problem description, the advisor thought he recognized a common problem. This
frame that the advisor invoked was a red herring. The overall lesson for technical support is difficult to apply: listen carefully but with skepticism.

Those first two Einstellung cases would have been avoided with better listening or more skepticism. The third case was caused by a small deficiency in the advisor's knowledge. This deficiency caused the advisor to incorrectly assume a domain of hypotheses was impossible. In other queries, the advisors consciously tried to avoid this problem by testing hypotheses that they believed were false. Sometimes they tested impossible hypotheses only to humor their clients (see Table 17 for an example), but they also tested other impossible hypotheses if quick tests were available. This was clear when they prefaced an instruction with, "I really don't think this could be it, but let's check real quick..."

Advisors also made obvious efforts to listen carefully to the problem and apply skepticism as needed. The fourth example of Einstellung was caused by the advisor trying to gain an overly deep understanding of the client's whole setup, when a surface-level question about the setup would have found the problem much sooner. Advisors often had clients redo tasks that the clients said they had already done, just to be sure that the client really understood the basic steps needed. Advisors frequently asked clients to go through the steps they took when the problem occurred, both to gain a full understanding of the problem and to ensure that the client was using the correct steps.

In summary, advisors occasionally solved problems inefficiently but there were no systematic errors they made that caused the inefficiencies. Most causes of inefficiency were simply misapplications of heuristics that worked most of the time, or were aberrations where the advisor simply didn't think of problem from the right perspective that one time.
General Conclusion

This paper has described telephone technical support using ethnographic data. The results were found to partially support previous technical support research findings involving conversational control and client participation, and also to demonstrate how contributions of advisors and clients change across three types of queries. The qualitative analysis of problem solving strategies applied theoretical models to this real-world data and explained that a combination of schema/frame theory and the ideas of the Gestalt psychologists are enough to describe most of technical-support behavior, although not all of it.

Some unanswered questions remain. Although the description of who controls the interaction is interesting, there is no empirical data that indicates what the optimal balance of control is in a technical support interaction. Previous research has implied that more control to the client is better, and has found that expert clients generally have more control that novice clients, but there has been no research into the optimal amount of control for a client, novice or expert, to have. Another unanswered question is how much of a difference there is between telephone and face-to-face support. Part of the reason I was interested in studying telephone support is that it demands a level of collaboration between the client and advisor that face-to-face support does not. In face-to-face support the advisor could potentially sit down at the client's computer and fix the problem with no input from the client. However, it would be interesting to see whether the process of hypothesis testing is different in face-to-face interactions, and also to directly compare the client's participation and use of verification requests between the two modalities.
One goal of ethnographic research is to provide a basis for more controlled basic research. One direction for this research could be to test the model given here by limiting the domain to a certain set of computer problems. It would be helpful to try to define the shape of advisors' system knowledge hierarchy and to determine, as hypothesized here, whether changes in hypothesis domains or in frame hierarchies map onto changes in the relevant set of system knowledge. Another way to test the model would be to use client input. Evidence presented here suggested that clients only suggest hypotheses at expected points in the advisors' reasoning processes. It would be interesting to manipulate the timing and type of client input to see how it affects the advisors' hypothesis generation and testing.

Future research should also make an effort to identify expert and novice clients. I attempted to measure the advisors' impressions of client expertise by having them give an expertise rating after each call. However, this effort was characterized by a large amount of missing data and most of the ratings were at the midpoint of the scale. An anchored rating scale may help with the latter problem. Also, it would be helpful to know the advisors' first impressions of their clients' expertise, since that is likely to influence what the advisor does early in the interaction. However, asking the advisors to give ratings during the interaction adds to their cognitive load, which is already high, and could potentially influence the advisors' actions. In some settings it may be possible to identify two very different client bases, one novice and one expert, and look for differences in interactions with clients from each of the groups. I attempted to do this by comparing questions involving more and less sophisticated products, but the distinction was probably not strong enough.
Technical support is an increasingly important asset to customers as computer technology becomes more pervasive. It is also a valuable realistic source of information on everyday problem solving and basic goal-oriented human-human interaction. Basic research in this area can teach us about troubleshooting and problem solving, while insight into technical support can help improve the technical support process, benefiting both companies and customers.
References


Appendix A: Individual Differences Among Advisors

The number of advisors made it impractical to add the factor of "advisor" to the factor of role in the ANOVAs presented in this paper because the cells sizes became very small. However, as long as the factor of advisor did not interact with role, the main effects of role that were found could still be interpreted. Figures 12 - 15 indicate qualitatively that advisor might have had a significant main effect in the analyses but that there probably would not have been an interaction between advisor and role for these dependent variables. Each of these figure shows a line for each of the seven advisors who had at least ten queries, as well as a line representing the mean for all eleven advisors.

In each case there were one or two advisors who used a pattern different from the mean, but the majority of advisors followed the same pattern of use, albeit with different base rates. The fact that base rates of use varied across advisors indicates that there probably is a main effect for advisor.

Figure 12: The use of the category "instruction" across roles by the advisors.
Figure 13: The use of the category "question" across roles by the advisors.

Figure 14: The use of the category "statement" across roles by the advisors.
Figure 15: The use of controlling utterances across roles by the advisors.