Journal of Business and Technical Communication

http://jbt.sagepub.com

Using Customer Data to Drive Documentation Design Decisions

Karl L. Smart and Matthew E. Whiting

Journal of Business and Technical Communication 2002; 16; 115
DOI: 10.1177/1050651902016002001

This article shows how user-centered design can be applied to documentation and reports the results of a two-year contextual design study. The article (1) demonstrates how contextual design can be applied to information and (2) reports some of the study's results, outlining key insights gleaned about users. The study found that users vary widely in their information needs and preferences. Users employ a variety of learning strategies in learning new software and in overcoming problems encountered within applications. Documentation can better meet variances in learning styles and user preferences when tightly integrated into applications, accessible in the user's own language. Additionally, documentation is most beneficial when several assistance options exist for users to choose among, varying according to context, task, and user need. Finally, the article discusses the constraints that affect the implementation of design ideas and explores implications for practice and additional research.

Using Customer Data to Drive Documentation Design Decisions

KARL L. SMART
Central Michigan University
MATTHEW E. WHITING
Microsoft Corporation

Successful organizations understand the necessity of designing products and services that meet customers' needs and expectations. In a competitive global market, seeking customer satisfaction has become an important business strategy—a means for companies to gain competitive advantage and maintain economic viability (Parasuraman; Rust and Zahorik). Frequently, businesses invoke common managerial mantras: "Get close to the customer," and "Listen to the voice of the customer" (Leonard and Rayport 102). Donald Norman observes that "modern industry must distinguish itself through its consideration of the needs of its customers. . . . As companies design more for usability and understanding, they will discover a competitive edge, for these principles save customers time and money while increasing morale" (*Design* vi-vii). To create products and services that meet customers' needs, product developers must gather customer data and use the data to drive design decisions. Unless product developers have direct contact with users and let customer data guide their design, the systems or products they develop will reflect their own biases, rationalizations, and views rather than

Journal of Business and Technical Communication, Vol. 16 No. 2 April 2002 115-169

genuine customer needs. Studies show that the more customer feedback a project has, the more likely it is to be successful (Keil and Carmel).

As technology and products become increasingly complex, developers, or designers, must increase their awareness of customer needs and the ways humans interact with systems. As JoAnn Hackos and Janice Redish explain:

> Good design happens only when designers understand people as well as technology. . . . Designs that don't meet users' needs will often fail in the workplace or in the market, resulting in high costs in productivity, frustration, and errors that impact users and their organizations. (1, 25)

Although an increasing amount of research demonstrates a user-centered approach in product development, especially in the computer industry (Carroll, *Scenario*; Den Buurman; Leonard and Rayport; Schuler and Namioka), little research specifically discusses how users deal with information or explains how to create user-centered documentation (Raven and Flanders; Beabes and Flanders).

The importance of documentation has grown throughout the past decade as organizations have reassessed their need for it and its role. An increasing amount of research has focused on demonstrating the value of technical documentation (Mead; Redish, "Adding"), with some research even demonstrating a causal link between documentation and customer satisfaction (Smart, Madrigal, and Seawright). The value of documentation has increased in an information age because the document—taking such forms as an audit report, a promotional brochure, a pharmaceutical drug-package insert, a computer manual, or an online help system—is one of the few material products of information work, a reflection of intellectual capital (Bernhardt and McCully; Norton; Stewart).

Because of the growing importance of information relating to products and services, additional research is needed to show how customer data can be gathered and used to drive decisions about information design. This article reports on the results of a two-year study of a development team within ProQuest (pseudonym), a major computer software company, as the team worked with customers to explore how users learned new software and how they behaved when encountering problems within software applications. Both ProQuest and the participants of the study granted permission for the publication of the study's results. In addition, the research met the university's standards for human subjects research and was approved by the university's human-subject review board.

ProQuest is a high-tech company that develops and markets software for home and business use through retail and corporate distribution channels. The average users of ProQuest products (major office automation/productivity applications) are office workers in all types of industries. As the software market became more competitive during the mid-1990s, ProQuest (previously differentiated from its competition by its outstanding customer support) sought further ways to strengthen customers' satisfaction with its products, particularly as its market share began to be eroded by competitive products marketed by larger software companies with more money and clout. ProQuest realized that to design innovative products that would better meet customers' needs, it required more data from actual users and a better understanding of the work practices and work cultures of users. To that end, the development team explored methods of user assistance and documentation that helped users learn new software and recover from problems encountered when using software—with the overarching goal of finding ways to improve users' experience with software applications.

To gather and analyze information on computer users' wants, needs, and work habits, the development team used a contextual design method developed by Hugh Beyer and Karen Holtzblatt. In this article, we demonstrate the way this contextual design method was applied to documentation design and report the results of the contextual design study, outlining insights gleaned about users and showing how contextual data can be used to inform design decisions. Although the data from the complex contextual design process used by ProQuest also serve as the data for our study, we resisted imposing additional qualitative methods on an already complex research and design process. In part, this article shows how individuals not trained in traditional qualitative methods learned and applied a complex ethnographic method to gather user data and to inform documentation design decisions. The primary purpose of the contextual design study was to gather customer data for product design; however, the process used to collect the data as well as the data have significant applications and implications for others and serve as the basis of our research data.

First, we demonstrate the contextual design methodology, outlining the development team organization and research focus (how users learn and how they solve problems, or get unstuck), showing how the team gathered and interpreted user data, and describing the process of creating an affinity diagram and consolidated work models. Next, we report several of the key insights resulting from the study relating to users' preferences and variances in learning, experience with application information, and experience with documentation. Then, building on the results of the study, we show how the development team used contextual data to inform design decisions. Finally, we discuss the constraints and resistances (technological, user, and corporate) that affected the team's implementation of design ideas from the study and the implications of the study for both practice and research.

CONTEXTUAL DESIGN METHODOLOGY

Contextual design is a user-centered design approach based on ethnographic methods adapted to the discipline of computer science and systems design. Ethnography, with related approaches of action inquiry and participatory action research, has become part of "a quiet methodological revolution" in the social sciences (Denzin and Lincoln vii). The goal of these approaches involves gaining experiential knowledge through cooperative participation with another group, with the intent of producing "knowledge and action directly useful to [the] group" (Reason 269). Contextual design uses a data-gathering interviewing method called contextual inquiry (CI). Beyer and Holtzblatt define *contextual inquiry* as "a field data-gathering technique that studies a few carefully selected individuals in depth to arrive at a fuller understanding of the work practice across all customers" (37; see also Raven and Flanders).

Although significant research has focused on user-centered design methods for computer applications, less time and effort have been devoted to creating documentation genuinely based on customer need because most companies traditionally view documenting software as a secondary support issue—even though some research suggests that 30% of human errors with computers result from information that is poorly developed (Bailey). Knowing that users needed more than online help and printed instruction manuals to fulfill their information requirements (Hackos and Redish 408), ProQuest organized a development team to explore users' experience with information and documentation components of software. The following subsections detail how ProQuest organized the development team (which we call the contextual inquiry, or CI, team), how the team determined its research focus, and how the team gathered and interpreted contextual data to create an affinity diagram (a large model showing hierarchical relationships in the customer data gathered by the CI team) and to construct consolidated work models (models that capture the culture, environment, work processes, and information flow of users).

Organizing the Team and Determining Its Research Focus

Because the development team would study information and documentation components of software, the leadership of the team was organized from within the technical documentation department of ProQuest. A documentation group manager was selected to lead the team, and three others from the documentation department were assigned to work on the team full-time: a technical editor who was familiar with print, a technical writer who was known as the online help guru of the department, and a technical writer who had also worked as an editor. From the project's conception, the company understood that a documentation department could not independently implement the changes that would likely result from the study, so the team was designed to have members from outside the documentation department to create the cross-functional group needed for a valid study. These additional members included a lead software developer, a usability specialist, a user interface designer, a marketing representative, and several other software developers who rotated assignments with the team throughout the project. This development team, the CI team, became part of ProQuest's larger R&D effort to design the next generation of software applications.

Realizing that many complaints had been expressed about the uselessness of documentation, particularly in the popular press (Irvin; Grech; Rettig), the CI team wanted to determine not only how documentation is used but how users learn and interact with applications and how best to support that interaction. As a result, the team initially set out not to design a new style or type of documentation but to understand users better—their work practices and needs—and to determine how to develop applications, not just documentation, that would best assist users. This focus led the CI team to look specifically at how users learn new software, how users get unstuck within computer applications (the problem-solving strategies they employ), and how documentation (defined broadly as any information in the program, from menu structures and long prompts to printed manuals and online help) interfaces with users' learning and problem-solving strategies.

The CI team realized that technical communicators and information designers frequently operate from largely unverified assumptions about how and why individuals use documentation. The team also realized that documentation usage was in part dependent on how people work—how they learn and solve problems—as well as on cultural and environmental factors in the workplace. Therefore, to explore the specific ways that information and documentation help individuals complete their work using computers, the CI team focused their efforts on obtaining answers to two questions: (1) How do people learn a software program? and (2) How do users get unstuck? The two research questions of the CI team also serve as the primary questions we address in this article.

Gathering User Data

Work-site interviews with individual software users are the foundation of the entire CI process. The purpose of the interviews is to explore the work environments and practices of users rather than test predetermined hypotheses. Interviewers, as participant-observers, seek to understand the work context by going to users' workplaces and watching them do their own work. The interviewer and the user become partners as they discuss the user's work and uncover unarticulated aspects of it. As Karen Holtzblatt and Hugh Beyer explain:

> Contextual Inquiry provides techniques to get data from users *in context*: while they work at real tasks in their workplace. In a contextual interview the interviewer observes the user at work and can interrupt at any time and ask questions as an outsider: "What are you doing now?" "Isn't there a policy for this?" "Is that what you expect to happen?"
>
> Confronted in the moment of doing the work, users can enter into a conversation about what is happening, why, and the implications for any supporting system. The user and interviewer discover together what was previously implicit in the user's mind. Talking about work as it happens, artifacts created previously, and specific past projects reveals the user's job beyond the work done on that day. (93)

Unlike other methods that use a systematic sampling to determine subjects to study, the CI method seeks participants with widely different roles who perform an assortment of tasks and work in different ways. Wanting to find a spectrum of participants with different skills in a variety of workplaces, the CI team identified potential organizations and individuals for the study by looking for ProQuest and non-ProQuest product users who performed different types of work in diverse organizations (government, business, and education). To make certain the interviews captured a heterogeneous group of customers and users, the team selected a representative number of male and female users from these diverse organizations in different regions of the United States. Following the standard CI protocol of interviewing 15 to 20 customers from at least four or more work sites, the team conducted a total of 18 interviews (8 men and 10 women) at 10 different sites. Five of the interviews were conducted with individuals working at government agencies, 8 in businesses, and 5 at educational institutions; the sites were located in five regions of the United States. Table 1 details some of the demographic information about the users interviewed, including gender, job description, organization type, and location.

The participants worked in a variety of roles, from managerial to secretarial and from legal research to product support. Although the team's participant-selection method appears to be less systematic than that of some other methodologies, research and experience have shown that individuals exhibit surprisingly few different approaches when completing tasks—that underlying commonalties exist among seemingly dissimilar users. As Beyer and Holtzblatt observe, "In every case we have studied, we discover that the underlying structure of work practice is consistent enough by the time 10 to 20 interviews have been conducted, we are discovering little that is new" (38).

The site interviews were conducted by one or two members of the CI team, generally with one asking questions and working with the user and the other taking detailed notes. In addition, each interview was tape-recorded. The interviewers would later listen to the tape recording and validate the notes taken. These interviews consisted of more than just talking to users. A fundamental principle of contextual design centers on the need to partner with users to understand their work habits and environment. Although the interviewers would ask questions of the user, the questions led to users actually performing tasks and completing work, with the interviewers and user examin-

TABLE 1
Demographic Data on CI Users Interviewed

User #	*Gender*	*Job Description*	*Organization Type*	*Location*
1	Male	City manager	Government	Orem, UT
2	Female	Software technician	Government	Orem, UT
3	Male	Vice president of document services	Business	Provo, UT
4	Female	Development office manager	Education	Salt Lake City, UT
5	Female	Department executive secretary	Education	Salt Lake City, UT
6	Female	Administrative assistant	Government	Orem, UT
7	Female	Administrative assistant	Business	Provo, UT
8	Male	University librarian	Education	Berkeley, CA
9	Male	Help desk manager	Education	Berkeley, CA
10	Female	Insurance underwriter	Business	San Francisco, CA
11	Male	Government economist	Government	Chicago, IL
12	Female	Library cataloging manager	Education	Berkeley, CA
13	Male	Accountant	Business	Chicago, IL
14	Female	Marketing assistant to vice president	Business	Seattle, WA
15	Male	City program director	Government	Chicago, IL
16	Female	Office manager	Business	High Point, NC
17	Female	Computer support specialist	Business	High Point, NC
18	Male	Sales manager	Business	High Point, NC

NOTE: CI = contextual inquiry.

ing and discussing the work as it was performed—for what people say is often different than what they do (Hodder 119). The participants engaged in their regular work when meeting with the interviewers, using a variety of software applications. The interviewers looked for instances when participants had to learn something about the application they were using or when they needed to recover from some error or problem. The strategies they used in learning and problem solving were more important than the particular applications they used.

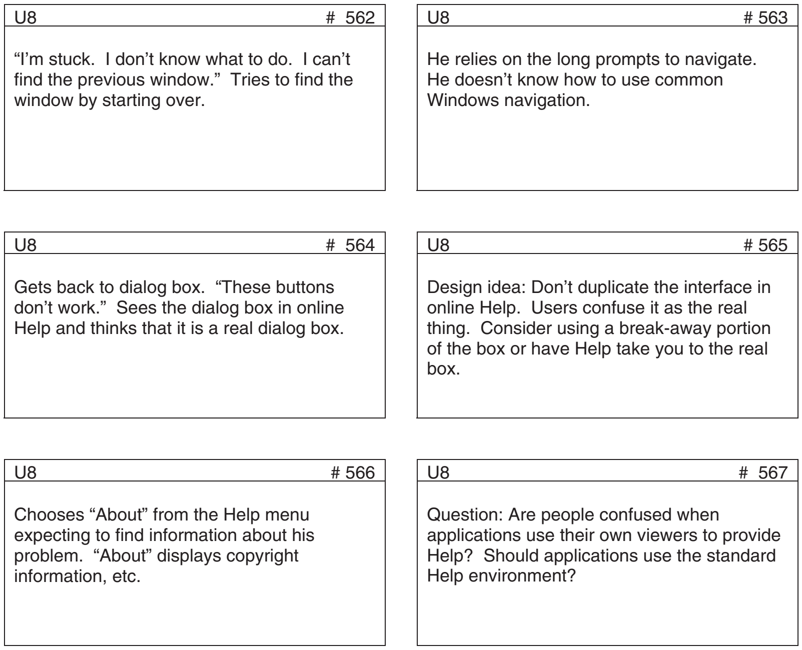
The data gathering also included collecting artifacts from the site that participants used in learning a task or getting unstuck—such things as copies of pages referred to in print documentation, a printed copy of online help screens, and copies of any internal documentation or user-created tip sheets. The use of artifacts has become a significant element in many ethnographic approaches (Hodder). The CI team used artifacts as physical evidence to support the intents and actions of users and as a reminder that systems design needed to accommodate such intents and actions. In addition, the team gathered detailed notes about the work culture and environment of each user.

The interviewers watched the participants using a variety of software (not just the software developed by ProQuest): word processors, presentation graphics, spreadsheets, e-mail, and custom-built in-house applications. The interviews centered on observing users actually working. To get users to perform tasks in the context of their work environment, interviewers posed questions that forced users to do, not just recollect, their work: When was the last time you were stuck in an application? What did you do? How did you get unstuck? Can you show us the process by recreating what you did? These questions helped users reconstruct their work processes and, as a result, revealed work practices frequently hidden or forgotten by the user. The average interview lasted two to three hours.

Interpreting the Data

After collecting data from the interviews, the interviewers returned from the sites to meet with the entire CI team to interpret the data. The CI team held these interpretative sessions as soon as possible after the interviewers returned from each site visit. Just prior to an interpretative session, the interviewers would listen to the tape recording of the interview and review their notes. As the CI team gathered, one person sat at a computer and captured information in note-card format as the interviewers discussed the interview. Each note card identified the participant, or user, from the study (U1, U2, etc.) and a sequence for the information captured in the interview (#1, #2, #3, etc.). Each card contained a single piece of information from the interview or a possible design idea or insight. Through the interpretative sessions, the CI team created 1,407 note cards, with an average of about 75 individual cards per participant. Figure 1 shows a selection of six note cards generated at the interpretative session for User 8 (U8) and the sequence of the data for later reference (#562–#567). The events described in Figure 1 occurred near the beginning of the interview with U8.

The note cards described the actions of the users, often with direct quotes from the users that captured noteworthy actions, concerns, attitudes, or problems. Sometimes, a single event produced multiple cards that identified different aspects of a problem. The cards shown in Figure 1, although not all-inclusive, are representative of the types of cards produced during the interpretative sessions. Most cards fell



U8 # 562

"I'm stuck. I don't know what to do. I can't find the previous window." Tries to find the window by starting over.

U8 # 563

He relies on the long prompts to navigate. He doesn't know how to use common Windows navigation.

U8 # 564

Gets back to dialog box. "These buttons don't work." Sees the dialog box in online Help and thinks that it is a real dialog box.

U8 # 565

Design idea: Don't duplicate the interface in online Help. Users confuse it as the real thing. Consider using a break-away portion of the box or have Help take you to the real box.

U8 # 566

Chooses "About" from the Help menu expecting to find information about his problem. "About" displays copyright information, etc.

U8 # 567

Question: Are people confused when applications use their own viewers to provide Help? Should applications use the standard Help environment?

Figure 1. Example of User Information Captured on Note Cards in Interpretative Sessions

into one of four categories: direct quotes, descriptions of actions, interviewer or team questions (often independent of the actual interview but generated by the team's discussion of the data), or design ideas (possible innovations to be discussed later). The note cards were projected electronically on a screen during the meeting so the team could verify and correct the information. Once the CI team agreed on the information on the cards, they were printed and became a permanent record of the interview. These cards were later organized to create an affinity diagram.

In addition to the note cards, the CI team used information from the interviews to create models of users' work—representing specifically such things as intents, purposes, or motives behind accomplishing tasks; actual structures of physical environments and of task processes; and strategies used in completing work, including communication patterns and cultural influences. Whereas the note cards are text-based descriptions of users' work, work models are visual representations of users' work and work environments, capturing various aspects of the users' work graphically. Work involves detailed, complex activities that have significant implications for system and information design and support. Although the design of new systems may facilitate existing processes and work activities, it could also potentially change them. For example, an existing company may have a process for handling rush orders, in which an operator who receives an incoming rush order phones the person filling the order to alert that person that a rush order is on the way. A new order-processing system could inadvertently eliminate this advance notice, resulting in rush orders being shipped late. However, a work model that captures the sequence of handling a rush order would make explicit such informal practices that are critical to the success of new systems.

To better understand and account for users' actual work activities, Beyer and Holtzblatt have developed five general work models specifically relevant to design (89-123): a sequence model, a flow model, a cultural model, an artifact model, and a physical model. These models are similar to existing diagramming techniques, such as process flows, transition diagrams, object models, data-flow diagrams, and blueprints (Suchman, "Representations"; Yourdon and Constantine). The models provide a formal language of visual diagramming that can be used to explain the work situation and to generate design ideas. Table 2 briefly describes each model and its purpose.

To create work models for each of the users, the CI team used recorded conversations and detailed notes about various aspects of the work and work culture of users as well as sketched diagrams of the actual work desk and physical surroundings. The work models represented different aspects of software-application work that the CI team needed to account for later in the design phase. For instance, the process of an individual using a word processor to type letters is one work activity that could be identified in a sequence model. But additional work activities and factors that affect how a system would support this activity must be considered. What are the specific parts of a user's letter (return address, date, inside address, salutation, etc.)? Does a user frequently write certain types of letters (sales, request, rejection, etc.)? Does a user follow a particular sequence when writing certain types of letters? What are the related activities involved in writing a letter (such as referring to an original letter or getting a name and address from an e-mail message or Rolodex)? Does the user address an envelope or label in addition to writing the letter? How does a label differ from an envelope? Is the printer in the same location as the user? Is it near the keyboard? Is the person creating the let-

TABLE 2
Description of Different Work Models Created in Interpretative Sessions

Type of Work Model	*Description and Purpose*
Sequence model	Shows the process of a work task or action, including a trigger that causes the sequence, steps involved in the process (with corresponding order), and any breakdowns in the steps.
Flow model	Shows the user in relationship with others and the flow of information between individuals, including artifacts used, communication topics and actions, the places where work is done, and breakdowns.
Cultural model	Shows the culture of the user's work environment, including expectations, desires, policies, values, influences, and attitudes about work.
Artifact model	Shows artifacts that are involved in the user's work, with explanatory notes keyed to how the artifact is used.
Physical model	Shows the physical environment of the user's office and work space, including its physical structure, the arrangement of furniture and objects, the layout of work, and the placement of artifacts.

ter skilled with the word processor or just learning to use it? Knowing this type of work information is critical to designing effective support systems. The five work models provide a graphic way to capture this type of information in a format that can be replicated across a diverse user base and later compared and consolidated for design purposes. Therefore, the CI team created each of the five work models for every user, with multiple models created for some users (e.g., more than one sequence model was created for several of the users to show the sequences they followed in various tasks). Later, the individual models were consolidated to create summative models that identified work patterns and issues common to several of the users.

The work models captured details about the users' work environment and work practices. Each model reflected a different perspective of a user's job—such as the person's role or roles in the organization (formal and informal), the responsibilities associated with those roles, and the ways the exchange of artifacts helped in carrying out responsibilities. For instance, one sequence model details how one participant accomplished a specific task and how she used an artifact to accomplish that task. Or, another sequence model shows the strategies a user (U11) followed when stuck in an application (see Figure 2).

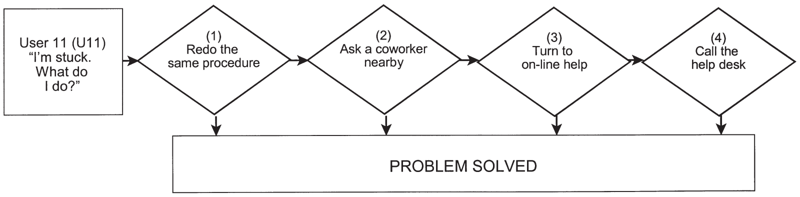


Figure 2. Sequence Model for User 11 Showing His Sequence for Solving a Problem

Creating the work models, along with formalizing the note cards, ensured that later discussions were based on actual data and that design ideas and discussions could be grounded from data. The CI team followed this process of formalizing note cards and creating work models in interpretative sessions for each of the 18 users interviewed.

To some, the process of having the entire team review information from each interview to create note cards and work models may seem cumbersome and even archaic, given the technologies available for organizing and managing complex observational data. Although sophisticated programs could locate patterns in data, such programs would neither fully communicate what interviewers learned in their site visits nor allow the entire CI team to reconcile varying messages from diverse users and come to a consensus as to the meaning of the contextual data and how they should be used. In addition, in database management systems, data may be "miscoded, mislabeled, mislinked, and mislaid" (Huberman and Miles 183). In the interpretative process, the data were clarified and verified by the interviewers, and the rest of the CI team gained a shared understanding of the users they did not interview. The clarification and verification process resulted in better, more accurate data, and the creation of the note cards put the data in a form appropriate to drive the later design process of the CI team.

The collaborative nature of interpreting the data in the manner described provided multiple perspectives on problems and issues and served to promote effective cross-functional cooperation among team members (who handled various job duties and product specialties within the company). The note cards and work models became the basis for creating the affinity diagram and consolidated work models, which became the essential core of design. Creating the affinity diagram was an interactive, real-time process for the entire CI team, one that required inductive reasoning to find meaningful connections between seemingly unrelated data. This process is not easily duplicated with observational analysis programs. Furthermore, the processes and models used by the CI team show that individuals not trained in complex qualitative research can learn and use sophisticated tools that provide practical data that can be readily applied to design.

Creating an Affinity Diagram

After capturing information from the interviews in note cards and in work models during interpretative sessions (which occurred during a series of months as interviewers returned from site visits), the CI team followed the inductive process of bringing all the data together to create external representations of work practices. The first step involved creating an *affinity diagram*, a hierarchical structured diagram in which note cards with an affinity, or similarity, to one another are grouped together according to related issues, worries, or problems relevant to the team's research focus. Affinity diagrams are based on total quality principles and processes developed in Japan (Brassard; Kawakita). By grouping together similar issues in the structure of the affinity diagram, relevant themes emerge. The intent of the diagram is to organize data across all the customers, thus revealing the scope of the data and identifying any holes or weaknesses in the data. Organizing the data within an affinity diagram makes key issues stand out, emphasizing critical knowledge about customers in an easy-to-share format (Beyer and Holtzblatt 154). The process of creating an affinity diagram helped the CI team arrive at a consensus as to what the data meant.

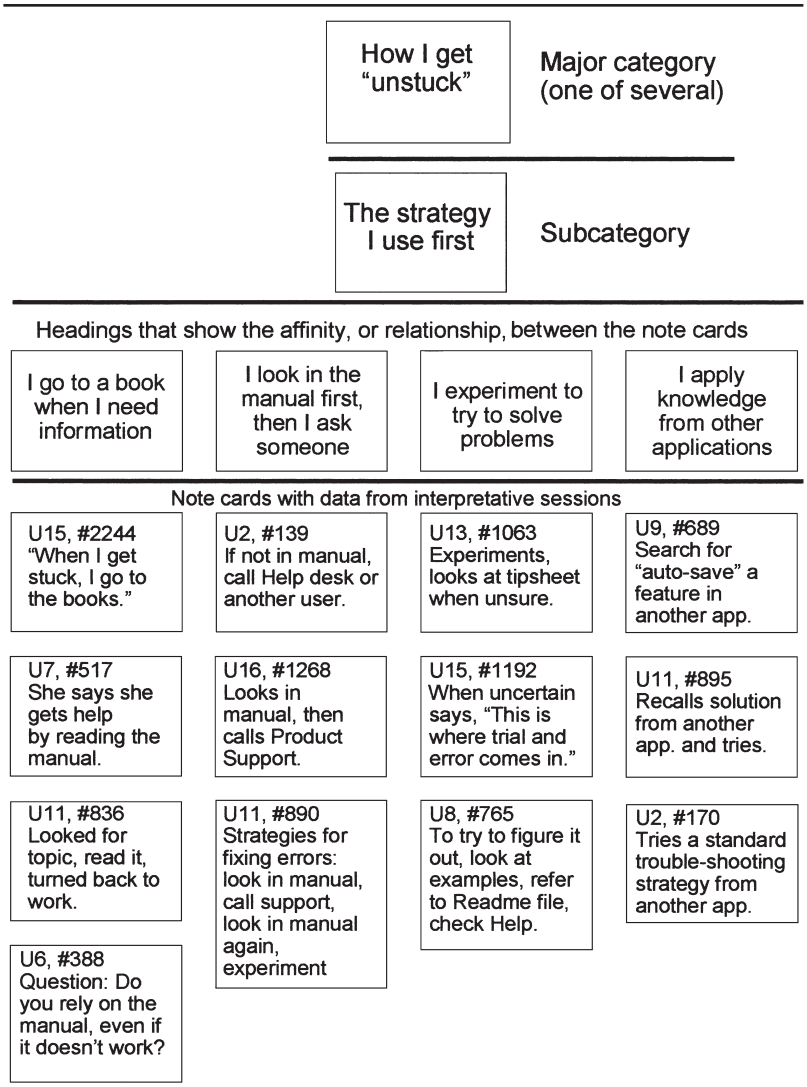
The first step to consolidating the data—building the affinity diagram—helped the CI team find common themes and structures from the individual note cards. The team set aside an entire day to build the affinity diagram. Rather than starting with a predefined structure or set of categories, all the note cards were displayed on tables. One member of the team would select a card, then the other members of the team looked for notes that seemed to be related to, or have an affinity with, the selected card. The related cards were then organized and taped to a wall. Generally, cards had an affinity if they represented similar ideas or problems in the users' work. The grouped note cards were then labeled with a phrase or a sentence that delineated the work issue they represented. These first-level groups of cards were then grouped into more general categories, resulting in a hierarchical structure that organized the data into manageable chunks, with usually two or three hierarchal levels under the main areas of research focus. The resulting diagram became a tangible representation of the users' story, specifically depicting problems and issues the team needed to address in redesign. Figure 3 shows an example of an abbreviated portion of the affinity diagram, identifying several strategies users employed when stuck in an application.

The completed affinity diagram covered the entire wall of a large conference room and served as a constant reference for supporting decisions. Themes or general categories resulting from the affinity diagram covered a broad range, including such things as "How I get unstuck," "How I handle overwhelm," "How do I learn," "Why I don't learn," "What I do after I learn," and "Ways words confuse me." The conference room with the affinity diagram became the design center for the CI team. When members of the team would make a claim about a user or design idea, they frequently referred to the diagram to validate their claim. In addition, the affinity diagram provided a good mechanism for showcasing the CI team's work to new team members and internal visitors. CI team members would "walk the wall" for guests, using the highest level of affinity cards as talking points for discussion.

Consolidating Work Models

After creating the affinity diagram, the team consolidated the work models of all of the users. The purpose of consolidating the work models was to create concise visual representations or diagrams of the customer population, showing common structures without losing variations across customers (Beyer and Holtzblatt 154). Through the models, the CI team created schemata of the customer population, illustrating their actual work practices and environment.

The function of the consolidated models was to reflect all the information captured in individual models. Figure 4 shows an example of a consolidated sequence model created from several individual user sequence models. (We discuss this example in detail later in the article.) Taken together, the consolidated models provided detailed information about users and their environments: capturing specific pro-



How I get "unstuck" — Major category (one of several)

The strategy I use first — Subcategory

Headings that show the affinity, or relationship, between the note cards

Note cards with data from interpretative sessions

I go to a book when I need information	I look in the manual first, then I ask someone	I experiment to try to solve problems	I apply knowledge from other applications
U15, #2244 "When I get stuck, I go to the books."	U2, #139 If not in manual, call Help desk or another user.	U13, #1063 Experiments, looks at tipsheet when unsure.	U9, #689 Search for "auto-save" a feature in another app.
U7, #517 She says she gets help by reading the manual.	U16, #1268 Looks in manual, then calls Product Support.	U15, #1192 When uncertain says, "This is where trial and error comes in."	U11, #895 Recalls solution from another app. and tries.
U11, #836 Looked for topic, read it, turned back to work.	U11, #890 Strategies for fixing errors: look in manual, call support, look in manual again, experiment	U8, #765 To try to figure it out, look at examples, refer to Readme file, check Help.	U2, #170 Tries a standard trouble-shooting strategy from another app.
U6, #388 Question: Do you rely on the manual, even if it doesn't work?			

Figure 3. An Abbreviated Portion of the Affinity Diagram That Shows Strategies for Solving Problems and Getting Unstuck

cesses and actions, reflecting individual users' relationships with others within the organization, showing the environment of users in relationship to performing tasks, identifying commonalties from artifacts collected, and showing the physical environment of users.

RESULTS OF THE CONTEXTUAL INQUIRY

Collectively, the consolidated work models and the affinity diagram showed key elements of the users' work and behavior and became a crucial foundation for identifying users' needs and preferences and the subsequent design requirements. As a result of their contextual inquiry into how users learned to use software and how users got unstuck, the CI team arrived at several key insights about users' preferences and variances in learning and problem-solving strategies, experience with application information, and experience with documentation. These insights—substantiated from actual data in the affinity diagram and consolidated models (see, e.g., Figures 3 and 4)—are discussed in more detail in the following subsections.

Preferences and Variances in Learning and Problem-Solving Strategies

The following key insights were gleaned from the interpreted data about users' preferences and variances in learning and problem-solving strategies:

- Users have different learning styles and problem-solving strategies.
- Users have little time for formal learning or training.
- Users get angry when a new version destroys their knowledge base by improving the software.
- Users do not care if their methods are inefficient.

The primary purpose of our study (as was the main objective of the CI team's study) is to examine how individuals learn software and how they get unstuck when encountering difficulties. Although one of the key insights from the study—users have different learning styles and problem-solving strategies—may seem obvious in light of existing research on adult learning (Bostrom, Olfman, and Sein; Pask, "Styles"; Säljö), the specific learning profiles identified from the study add to our understanding of how we view users, with significant implications for the type and manner of information provided within applications.

In recent years, many companies have reduced the amount and type of documentation they ship with products in an effort to cut costs. Although that may result in short-term savings, differences in users' learning styles and problem-solving strategies suggest that

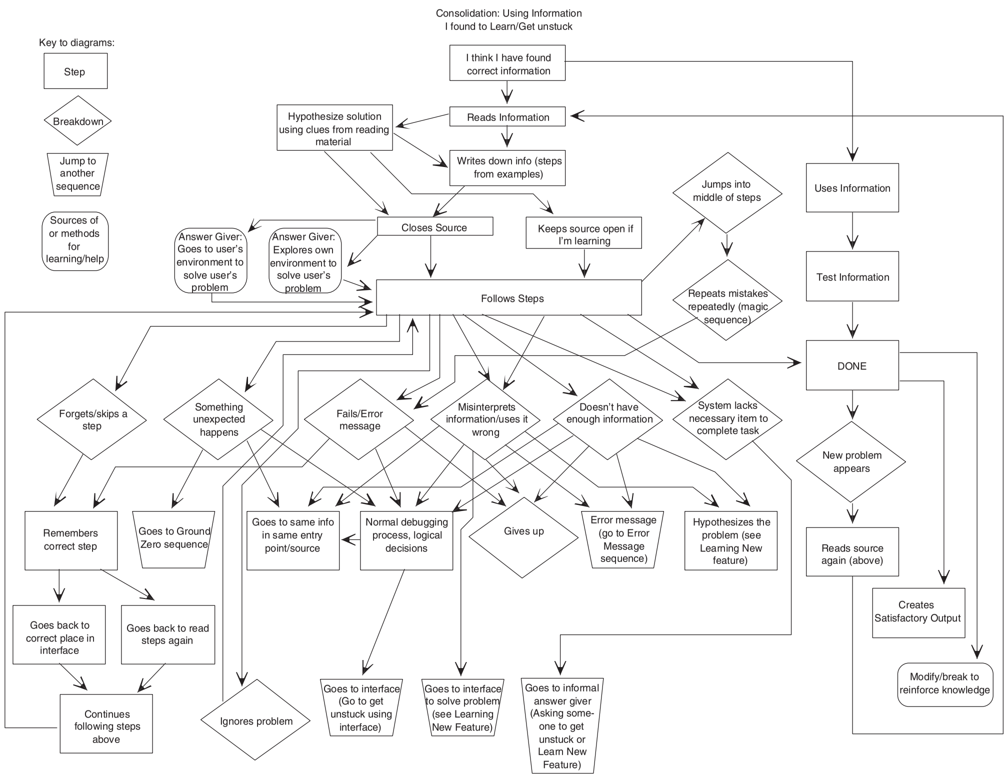

Consolidation: Using Information
I found to Learn/Get unstuck
Key to diagrams:
Step
Breakdown
Jump to another sequence
Sources of or methods for learning/help
I think I have found correct information
Reads Information
Hypothesize solution using clues from reading material
Writes down info (steps from examples)
Closes Source
Keeps source open if I'm learning
Answer Giver: Goes to user's environment to solve user's problem
Answer Giver: Explores own environment to solve user's problem
Follows Steps
Jumps into middle of steps
Repeats mistakes repeatedly (magic sequence)
Uses Information
Test Information
DONE
New problem appears
Reads source again (above)
Creates Satisfactory Output
Modify/break to reinforce knowledge
Forgets/skips a step
Something unexpected happens
Fails/Error message
Misinterprets information/uses it wrong
Doesn't have enough information
System lacks necessary item to complete task
Remembers correct step
Goes to Ground Zero sequence
Goes to same info in same entry point/source
Normal debugging process, logical decisions
Gives up
Error message (go to Error Message sequence)
Hypothesizes the problem (see Learning New feature)
Goes back to correct place in interface
Goes back to read steps again
Continues following steps above
Ignores problem
Goes to interface (Go to get unstuck using interface)
Goes to interface to solve problem (see Learning New Feature)
Goes to informal answer giver (Asking some-one to get unstuck or Learn New Feature)

Figure 4. Consolidated Sequence Model Showing How Individuals Use Information to Learn or Get Unstuck

companies should consider more, not fewer, support options. Although such a strategy may incur greater up-front costs, the resulting benefit of increased usability and customer satisfaction (and future sales and repeat purchases) could well outweigh the initial expenditure. The differences in the learning styles identified suggest the importance of providing information that better matches users' needs and preferences. In reviewing the consolidated models and the affinity diagram, the CI team identified three major types of user learning styles: experiential, reflective, and unintentional.

The profiles of users who have experiential learning styles fell into three categories—experimenters (users who like exploring on their own, often without a specific task to complete), do-to-learn users (those who learn by doing, usually in the context of a work task), and learn-to-do users (those who want to learn a concept first, then try it out). These profiles corroborate research in active learning: Some individuals learn best through active involvement in the learning process (Cross; Kolb; Lawler). Specifically, the findings support related research on elements of active learning applied to such areas as minimalism (Carroll, *Minimalism*) and "reading to learn to do" (Redish, "Reading"). For instance, experimenters and do-to-learn profiles reflect aspects of minimalism, an approach to creating computer documentation that relies heavily on task orientation, helping users to learn by doing (Van der Meij). A fundamental tenant of minimalism suggests that users learn most effectively by experimenting through guided exploration (Van der Meij and Carroll 23). Individuals exhibiting these profiles actively immersed themselves in using a software program, even when they were uncertain how to accomplish a specific task. They seemed to learn best by experimenting or trying things out. For example, several users commented on their preference to experiment or "do something" to learn a new application or feature: "I learn through lots of trial, lots of error. . . . I like to experiment and play around" (U4). "I just go—jump in and try things out" (U6). "I learned by trial and error" (U11). "I just get into it and try it out" (U16).

The learn-to-do profile resembles Janice Redish's principle of "reading to learn to do." Redish points out that most users do not read documentation for its own sake; rather, they read documents to learn the necessary information to help them accomplish a desired task ("Reading" 289), an approach validated in these findings. Although individuals exhibiting this profile liked to engage themselves actively with the software, their experimentation generally followed reading something about the application in the documentation first. Learning about the application through reading the documentation helped them to understand concepts and features that they then wanted to try. For example, one user (U10) indicated that she learned as she tried out a task she had read about: “Now that I’ve brought it up [found the item she was looking for], it makes sense.” Another user (U8) completed his work as part of the learning process, learning as a result of the need to get work done: “Just give me what I need to know to do it.”

In contrast to experiential learners who wanted to engage more immediately in the work task, reflective learners exhibited a more intrinsic, reflective approach to learning. The profiles of users who have reflective learning styles also fell into three categories—model builders, read-and-follow users, and watch-and-ask users. For instance, the profile of the model builder corresponds with previous research on mental models in users (Carroll and Olson; Gerlach and Kuo; Jonassen; Norman, “Some”). Although the research on mental models suggests that all users develop cognitive models of systems, the CI data from this study suggest that some users consciously construct models of a program or application in relation to what they already know as a learning strategy. Such users look to create an overall vision of what the program is and does and then work to fit specific experiences and components of that system into their vision or model. For example, a user (U2) wanted to “learn the whole picture, not just the necessary task info.” Another user (U1) indicated he had “a model of word processing” that he used in helping him learn a similar program. He claimed that this model helped him learn the new program in much less time than he took to learn an unfamiliar, complex drawing program: “If I understand the structure, I don’t need to be as methodical in my learning.”

The profiles of read-and-follow and watch-and-ask users reflect an intrinsic approach to learning that corresponds with additional learning research. For example, conversation theory characterizes a serialist learner who prefers learning in a very sequential, orderly fashion informed by rules and guidelines (Pask, *Conversation*), similar to read-and-follow and watch-and-ask approaches. Others have identified learners who prefer parroting or following exactly specific instructions—the read-and-follow users—as well as those who favor reflecting on, and integrating, what they see and experience with existing knowledge before acting—the watch-and-ask user (Coe; Schneider). Several users typified these learning strategies: “When I need to do something, I read the manual to see how to do it, then I do it" (U16). "I like the stroke-by-stroke technique in the manual so you can just walk through it" (U18). "I learned cut/copy/paste by watching [a coworker]. . . . It's more effective to learn as I watch someone else than doing it on my own" (U2). "I'm more willing to try a shortcut when someone shows me than learning it on my own" (U5). The CI interviews disclosed that several users preferred this reflective approach to learning.

The CI team also identified profiles of users who have unintentional learning styles. These learners, rather than exhibit a specific learning strategy, seemed to discover an answer to a question or a solution to a problem serendipitously. For example, one user (U2) found out by accident that she did not need to delete selected text before replacing it, a second user (U5) accidentally found out how to resize a window when trying to perform another function, and a third user (U8) determined where to put his path script unknowingly. Thus, some learning occurs tangentially even when a user's intent is not to learn but to complete some other task or action. Although the unintentional learner is not specifically identified in existing research, several studies describe barriers faced by adult learners. Some of these barriers to learning identified in research suggest obstacles to learning that the CI team observed in profiles of unintentional learners. One such obstacle—indifference—was demonstrated forcefully by a user: "I don't know how I learned, and I don't care!" (U3). Some learning theory suggests that significant learning occurs only when the subject matter is relevant to the personal interests of the learner and when external threats are low (Combs; Rogers and Freiberg). Other research suggests that anxiety can be an obstacle to learning as well as closely related factors such as arousal, attention, motivation, and emotions in general (Clark and Fiske; Mandler; Weiner). Certainly, these factors that affect learning play important roles in the way users approach learning new software applications and the way they behave when problems arise. A number of these factors were evident in profiles of users who exhibited an unintentional learning profile.

Although the existing research mentioned provides a good foundation for understanding users (and an approach like minimalism provides useful heuristics for designing documentation that suits an active learning style), the CI data found that some users exhibit characteristics from several learning-style profiles in their work. As Redish observes, "Users are not all the same, all the time, in all situations" ("Minimalism" 221). For instance, from the data, we saw a new user who combined a read-and-follow approach with a more active experimenting method in several instances. We also saw learning styles vary to meet the demands of a specific task. For example, one user installed software with a watch-and-ask approach but demonstrated a model-building profile when trying to understand how to use the software. The critical implication from the data is that because users have diverse learning styles and needs, systems and information that supports systems must be flexible enough to accommodate various differences.

In addition to identifying distinct learning-style profiles, the CI team also classified several problem-solving strategies users tried when encountering problems. The team found that when users were stuck, they usually employed one (or several) of the following problem-solving strategies (partially shown in Figure 3):

- consulting a book or manual
- looking in a manual, then asking someone for help
- applying knowledge from other applications
- trying to figure it out for themselves
- trying standard troubleshooting methods
- checking for simple things first
- applying information from training
- repeating faulty procedures
- thinking through the problem aloud
- consulting other people or places for information
- trying suggestions from others
- rebooting the computer

Having a clear sense of the problem-solving tactics users applied, the CI team was prepared to discuss how a system and documentation redesign could best assist users when stuck.

As with the learning profiles, individual users frequently used more than one problem-solving strategy when encountering difficulties, especially when the problems were complex. For example, many users became stuck when the option that they wished to access was grayed out (in many software applications, options are grayed out or dimmed when they cannot be used in a specific task). First, the user tried a procedure repeatedly, even when the action failed to bring the desired result after the first couple of attempts. After giving up on that procedure, the user turned to either a printed manual or online help.

If that proved futile, the user resorted to calling the help desk or a coworker in close proximity. These results suggest that the CI team look to find documentation and support options that would accommodate the various learning preferences and problem-solving strategies of users.

Despite the different learning profiles and problem-solving strategies identified, one constant surfaced among most users: Although they wanted to be competent with the software required to perform their jobs, most users had little time for formal learning or training. For example, users made such remarks as "When I'm busy, I don't want to have to learn something, I want to continue working" (U7); "A full day of training was too long and intense—I wish it was a half-day" (U10); "With all I juggle every day, I don't have time to learn" (U12); and "We don't use [the application's] full potential because of lack of training. When you're working, you need to work and can't take an hour to figure it out" (U16). The exigencies of the workplace forced users to learn only enough about an application to complete essential tasks, even if they would have eventually been more productive and better off if they had taken time to learn about a program more carefully and systematically (Carroll and Rosson).

Consequently, users became angry when a software application improved a function but in the improvement destroyed their existing knowledge base: "It didn't do what I wanted [the function keys were different from those in the previous version]" (U11); "I still use the old system to print checks. I have too many problems [printing checks] with the new system" (U13); "I know the program has an eraser. I just don't know where they put it [menu options changed from an earlier version]" (U15). Such observations showed that users preferred the methods and learning strategies they were familiar with and did not care if their methods were inefficient: "I use what works [uses tab and space bar to get date to the right margin rather than right justify], even though I know there's a better way" (U7); "I know this way works [instead of learning how to change uppercase to lowercase, the user erased entire words and started over]" (U3); "It's faster this way [uses a typewriter for labels and envelopes even though she knows the program could do it]" (U5). These findings have implications for dealing with the legacy of previous versions when redesigning systems and documentation. Applications must not blithely discard previous procedures and processes, and documentation must help bridge changes from old methods to new ones.

Experience with Application Information

The following key insights from the CI study regarding users' experience with application information suggest that we reassess how we view documentation:

- Users see the interface as part of the documentation.
- Quicktips are important in helping users complete tasks and get unstuck.
- Error messages are important for recovering from errors.
- Users need explanations referenced in their own terms.
- Information that is closer to the user is less costly and more reliable than is information that is farther from the user.

In analyzing the models and the information in the affinity diagram, the CI team realized that users view the interface (menus, dialog boxes, long prompts, etc.) as part of the documentation. Users frequently seek assistance and clues from information displayed in the interface. For example, long prompts and quicktips proved to be important aids in helping users complete tasks and get unstuck: "I use descriptions [long prompts] to help me figure things out" (U2); "I do what the interface tells me" (U6); "When I'm busy and am having a problem, I'll just go through all the keys and notice the prompts" (U7); "I look through the menus to find what I want" (U10); "With 'pop-up menus' [context-sensitive prompts], you know what you're doing" (U15).

In addition to interface information, error messages served as critical junctures determining whether users could recover from errors: "I basically know what the problem is [from the message], but I don't know how to work around it" (U9); "I don't understand what this means [referring to an error message]. What am I supposed to do?" (U11); "When I receive 'Error 12: Not enough memory,' I close the unnecessary program that's running" (U15). In one instance, a meaningless shared code error message caused U17 to quit what she was doing. Unsuccessful in understanding the message and recovering from the error, she stopped working and exited the program. These findings suggest that documentation specialists should carefully evaluate the input they provide in the interface and in long prompts and error messages.

Not understanding terminology in application information was one of the leading frustrations and problems for users. The results of the study confirmed that users need explanations to be referenced in their own terms or need help in finding what an unfamiliar application calls a function or task. From the affinity diagram, the CI team identified several ways words confused users:

- *The application required learning a special jargon*. “This isn’t giving me help [clicking on QuickFinder]” (U3). User tries to install a new font using “Insert” from a pull-down menu (U8). “That’s not what I’d call it. The name makes me think it will do something different than it does [thinks that “Character” will change the font]” (U11).
- *Words had specialized meanings within an application that were different from what users thought or expected*. “It must be a code word [referring to an unfamiliar term]” (U10). User goes to “Symbols” section in the manual but is looking for “Special Characters” (U15). User searches for “Keyboard” in help, whereas “Function key” or “Key strokes” would have been a better search term (U17).
- *Terminology was not consistent from application to application, even within the same operating system*. “Previous and Next are both search and hyperlink terms. That is confusing” (U8). “I can’t find how to do it [comment after an unsuccessful search]. Other programs use the term ‘autosave’” (U14). “I have to deal with software from nearly two dozen companies, and they all have a different name for things, and it all gets jumbled together” (U15).
- *Users knew the task they wanted to complete but did not know the name of the functions used in an application to complete it*. “I go to the index in the manual, but I couldn’t find the word, so I end up paging through to see if I can find what I want” (U9). “How would I type a word to describe that [wants to know what to search for to find out about a specific icon]?” (U10). “It’s like the Help says, ‘Guess. No, that’s not it. Guess again’” (U15). “You can read it [documentation], but some of it still doesn’t make sense. You go to the index to try to find something using what you think it’s called” (U16).

Because of the preponderance of language problems that existed in a variety of ways among most users, the CI team realized that any successful design or redesign must account for language variations and differences in users’ experiences with application information.

One of the most striking findings from the CI study involved the cost and reliability of information. In looking at where users go and strategies they adopt to get information, the CI team found that information that is closer to the user is less costly and more reliable in solving problems than is information that is farther from the user. Through cultural and flow models, the team identified three categories, or levels of proximity, where users obtained information, as shown in Table 3.

TABLE 3
Types of Available Information Sources, by Category/Proximity to the User

Category/Proximity of Information	*Types of Information Sources*
Informal internal (close to the user's immediate work area)	Coworker Informal training coordinator Informal troubleshooter
Formal internal (within the user's organization)	Formal trainer Master/mentor Information systems person Formal help-desk consultant
External (outside the user's organization)	Source of sale representative Hardware vendor Software vendor Formal external trainer External consultant

Closest in proximity to the user were the informal internal information sources. For instance, several coworkers served as potential sources of assistance. Frequently, a podmate or other coworker in the immediate vicinity of a user's desk helped when a user encountered a problem or needed to learn a new task or feature. Such a coworker provided support in many ways, either acting as a resident troubleshooter, functioning as an expert who came to the user's computer and solved the problem, or solving the problem collaboratively with the user by having the user come and demonstrate the problem on the coworker's machine. Although these informal sources, or coworkers, were interrupting their own work in giving assistance (thereby incurring costs by not functioning in their hired roles), the measurable, balance-sheet cost of this informal internal support was negligible. Users generally found the quality of assistance coming from these sources extremely high because those assisting them knew them and had a good understanding of the problem and the context.

Farther from the user were the formal internal information sources. These sources included formal trainers and help-desk consultants within the user's organization. In some companies, information systems workers also served a formal role in assisting users. A few organizations even had a position of a master/mentor, who was assigned to provide support to specific users. The assistance received from these formal internal sources was reasonably good because, in most cases, those helping had a good grasp of the users' work and work context. However, their understanding of the users and the context of their problems was not as great as that of informal internal sources; consequently, users did not perceive their assistance to be quite as reliable. These information sources are also more costly because they are often staffed as secondary support services that are not central to an organization's primary mission or work. Companies frequently spend significant resources to provide this type of internal support.

Farthest from the user were external information sources. These external sources included source of sale representatives, hardware vendors, software vendors, formal external trainers, and external consultants. Information from these sources proved to be the most costly and the least reliable (they were only marginally aware of the context of specific problems and users). For example, support calls to software and hardware vendors typically cost between 20 and 75 dollars per call (Nielsen 84). Those external sources that were able to take the time to understand the context of individual users were very costly to the company. These findings led the CI team to consider ways to provide information that would be close to the users' workplace. If, for example, documentation close to the user (such as a printed manual or online help) could provide accurate and useful information, the user's organization would likely save money and increase its productivity.

Experience with Documentation

Through the CI study, the team also discovered these key insights about users' experience with documentation and documentation usage:

- Users create their own documentation.
- Some users—such as help-desk consultants or macroprogrammers—want or need printed manuals.
- Tutorials are seldom used in the workplace.
- Indexes are crucial entry points to documentation.

In numerous instances, users created their own documentation—from formalized paper documents containing instructions gleaned from coworkers or formal training sessions to Post-it® notes containing handwritten reminders stuck to the monitor: "Once I know how to do this, I will write it down [shows a copy of a cheat sheet]" (U3); "I write down the important steps so I remember the next time I need to do this" (U8); "These Post-its [on the side of the monitor] help me remember the keystrokes" (U17). These tip sheets suggest that users needed to create documentation that matched their internal processes and needs and that users could not find solutions to particular work problems in product documentation (or that they did not have ready access to documentation); therefore, users created their own documentation for future reference and for sharing with coworkers solutions to problems. Users could benefit from software applications that helped them with the process of creating, managing, and distributing user-generated tip sheets.

Related to the earlier finding that users have little time for formal training, the CI team found that tutorials were seldom used in the workplace. Several factors seem related to users infrequently turning to tutorials: "I learn more by just jumping in and trying things" (U1); "It's so crowded and busy [sharing the work space with three others—with lots of noise and interruptions]; I don't have time" (U5); "I'm too busy doing my 'real' job to train" (U7); "I don't want to do this; it would take too much time" (U10); "This has nothing to do with my real job" (U12); "They get you started, but there aren't any [instructions] for the more advanced features" (U17). These comments likely reflect the reality that most tutorials focus on contrived scenarios that have little relevance to users' work and that users want to be actively involved in learning with actual tasks (Charney, Reder, and Wells; Suchman, *Plans*). Thus, if tutorials are to be successful, they should focus on tasks and features related to users' work. Also, a company's attitude toward training and learning affects whether users feel that improving their skills is part of their work and justifies taking time from other, often more pressing tasks.

The CI team also found that some users wanted and needed printed manuals. Some individuals, because of the nature of their jobs or roles within an organization (such as help-desk consultants and macroprogrammers) rely on printed manuals despite the increased delivery of information in online mediums. Printed manuals best supported these users' preferred learning styles and problem-solving strategies: "If I have a problem, I look in the manual" (U5); "I'm a manual user. All problems can be resolved in the manual if you just read it" (U6); "When I get stuck, I go to the book" (U15). For some users, their experience and success with printed manuals had created a high level of trust in the information printed in hard copy (U1, U6). One user (U7) felt that reading the manual could solve most problems, whereas another (U16) kept a copy of the manual next to her computer for whatever program she was currently using. Even though users sought assistance from sources in addition to the printed manuals (such as a help desk or software/hardware support), several users indicated that they generally looked in the manual before turning to these other sources (U2, U11, U16). Manuals seemed particularly important in particular situations, such as when the information was conceptual and users needed an overview, when the task being performed was a new task that would be performed repeatedly and the user wanted to commit it to memory, when the user was unfamiliar with an application and needed help getting started, or when online information was incomplete or poorly displayed.

Among all the users of the study, not one used or referred to the table of contents in trying to locate information, but indexes were crucial entry points to documentation in almost every instance: “When I need to know something, I go to the index” (U11); “The information in this manual is not helpful. I wouldn’t know how to look it up [frustrated in inability to find an entry in the index]” (U13); “Ah, help is nice to me [commenting after finding needed information through an index search]” (U15). Related to the use of indexes, users wanted and needed things explained in their own terms (as discussed earlier with users’ experience with application information). With indexes serving as an entry point to not only the documentation but also the application, users wanted things described in their own terms. Users expected to be directed to a term an application used if the term that they were familiar with was not used. The findings we have discussed in this section have several implications for the design of systems and the information that supports such systems as we look at how the contextual user data are transformed into actual design ideas.

USING CONTEXTUAL DATA TO INFORM DESIGN DECISIONS

A challenge with any user-centered design approach is finding ways to transform collected data into actual design ideas. Recent research has focused on ways to bridge the gap between the user information and the actual design process (Wood). This section demonstrates how the CI team applied the contextual design methodology with the collected data in a design process of visioning, building a user environment, storyboarding, and prototyping specific application features. Once they validated design ideas through prototyping, the members of the CI team developed design specifications and worked within technological and organizational constraints to implement their ideas. Although the proprietary nature of the data and the scope of this article limit us in giving a complete description of the design process, a selective discussion of design ideas and issues encountered provides useful insight into document and information design and reveals additional insights into the ways humans interact with computers.

Visioning

Although the contextual design process is detailed in other sources (Beyer and Holtzblatt; Hackos and Redish; Raven and Flanders), a brief review of the process the CI team used to transform data into design ideas demonstrates useful tools and strategies sometimes overlooked in designing documentation. Realizing that they were not just designing a new type of manual or help system but rather designing the tools necessary to support the work of the user, the CI team members tried to look for creative ways to provide user assistance. The gathered data reflected a diversity of users with different learning styles and needs who used various strategies for trying to solve problems. With such diversity in learning styles and problem-solving strategies, the team wanted robust support that assisted a variety of users. Consequently, the team used a visioning process to invent possible responses to different types of users and problem-solving strategies. As defined by Beyer and Holtzblatt, *visioning* is a process that involves focused, or grounded, brainstorming (227). In brainstorming, potential design ideas and solutions are discussed without critical evaluation (i.e., “That’s a dumb idea!” or “That idea is simply not feasible”). The brainstorming is focused, or grounded, in that ideas are based on, or driven by, the work of the customers as represented in the data. As the CI team brainstormed, the vision that evolved focused on what the user needed or needed to have done to learn new applications or features or solve problems encountered in applications.

Focusing on the key insights learned about users’ preferences and variances in learning, users’ experience with application information, and users’ experience with documentation, the CI team wanted a system that could customize assistance according to varying user needs.

Building on initial research on intelligent agents (see, e.g., Fischer 138-63), the team developed a concept they called the Blue Fairy (reminiscent of Pinocchio's Blue Fairy, who made Pinocchio's wish come true), an interactive agent whose function was to track what users were doing and provide assistance when needed. In discussing and refining the idea of a Blue Fairy, the team realized that the concept involved two separate functions. Subsequently, the team refined the idea into what became known as the Watcher and the Communicator.

The function of the Watcher was to analyze the setup of a user's computer and monitor the user's actions. By monitoring the user, the Watcher could identify possible problems. For instance, the CI team found that as a problem-solving strategy, some users would continue to repeat an action even if it failed to solve the problem. Generally, if users tried the same sequence unsuccessfully three times, they did not understand the problem they had encountered and needed help. If a system had a Watcher, then any time a user tried to perform an action and repeated the action three times without success, the system would take that as a cue that the user needed help. Information gathered by the Watcher would then be transferred to the Communicator. As part of an application that directly interacted with users, the Communicator would assist users in understanding and operating the application, warn the user of problems, step the user through tasks as needed, and help the user recover from errors. The Communicator could interact with users through voice or online text or pointing to pictures or parts of the interface.

From the data, the team realized that users often turned to fellow workers to get information about applications. The intent of the Communicator was to create an electronic neighbor that would provide needed information. For instance, if the Watcher noted that the user had unsuccessfully tried to complete an action three times (an action the team dubbed the *magic sequence*), the Communicator would intervene and ask, "Are you lost?" or "Do you need help?" If the user responded positively, the Communicator would then suggest possible interventions. The reliability of the information from the Communicator would be high because of its awareness of the user's needs based on the information gathered by the Watcher. Because users naturally gravitate toward talking to an actual person, for users to turn to the Communicator, it would need to have something that the regular podmate did not necessarily possess: superior application knowledge.

Building on the concept of the Watcher and the Communicator, the CI team explored methods of assistance that best suited variances in individual needs and preferences. Although the Communicator was the part of the program that interfaced directly with users as the electronic neighbor, it was not the actual support. The team used the visioning process to create types of user assistance to which the Communicator could direct users according to their differing learning profiles and problem-solving strategies. Table 4 highlights some of the major ideas the team developed for providing assistance to users. In generating this list of ideas, the CI team members did not focus on the immediate technical or financial feasibility of the ideas; rather, they developed a vision of what an application would do from which they would implement ideas over a series of product releases. Later in the process, the team looked for appropriate technology to build a system that would meet the identified needs within constraints.

The different types of assistance provided various support options based on the learning profiles and problem-solving strategies identified. For example, a learn-to-do user (who wants to learn a concept first and then try it out) may need help in figuring out how to use a particular feature to complete a task. To help such users, the proposed system had the Communication Center, which would provide orientation by finding and accessing an online discussion group from a list of resources. Or, the system could generate a list of people who could provide individual assistance to the user. In addition, learn-to-do as well as do-to-learn users who may fear that their exploratory learning may damage some files or a part of the system could turn on a safety switch that would allow them to explore the application in a practice area without saving any work or harming any existing data. Many applications have features, like an Undo command, that allow for some recovery from altering data in unintended ways, but this safety switch would be for users who want to learn but feel insecure about experimenting with the software.

The Watcher and the Communicator, aware of individual users' preferences, would help to suggest the appropriate type of assistance based on users' profiles and their actual actions while working at the computer. But, if users desired, they could select other assistance options independent of the Watcher and Communicator recommendations. For instance, learn-to-do users could select a get-the-big-picture approach through accessing the Demo Room, which would give users a freestanding quick-tour video. Although the data suggested they seldom used tutorials, many users (such as watch-and-

TABLE 4

Types of User Assistance Developed from CI Data

Type of Assistance	*Purpose*
Communication Center	Provides an area that allows two or more workers to work and talk together synchronously or asynchronously to solve problems and to access information and support produced by third-party companies.
Demo Room	Demonstrates tasks as users watch. Lets users see task concepts.
Map Maker	Provides a big picture of the system, showing relationships between features and tasks. Provides another way for users to access features and navigate the interface.
More Info	Gives users a way to expand interface items (text and icons) to get more information (long prompts expanding to paragraphs of information, with steps or examples as needed). Also, gives users a search option using a Natural Language Interface, which allows users to ask questions in their own language.
Cue Card	Provides users with exact steps to follow online, helping users to know exactly what to do. Displays steps according to what the user wants and where the user is in the program.
Tip-Sheet Manager	Lets the user record steps of an action or task and creates a tip sheet the user can run (as a cue card), save, print, or send to someone else.
Printed Cookbook	Gives the user information without having to use the computer. Lets users scan in printed form to follow along online or customize to fit individual needs.

NOTE: CI = contextual inquiry.

ask learners) liked watching someone perform an action before trying it themselves, and the Demo Room would allow them to do that.

Or, users could better understand the application's structure and build an appropriate model of the system through the Map Maker feature. Maps are graphic representations that depict the way features and tasks fit together, such as system, task, feature, comparison, menu, or button maps. For example, new users wanting to learn about the task of creating a new document in a word processor could select a task map titled "Make a Document." The document map would include a visual index that displayed a finished document, with specific aspects of the document labeled to assist users in trying those features (e.g., borders, bulleted lists, headers and footers, graphics). The CI team thought that this type of assistance would be appropriate for the model-builder learner because maps are models, or representations, of a system that could assist users in building their own models of tasks and features. Another type of map would be a visual index, a type of assistance that responds to language problems identified by the data. A visual of a document with call-outs could identify potentially unfamiliar terms (e.g., headers, footers, watermarks, bullets, etc.). Users select from the visual index the type of finished document they desire and then see in the visual that they selected the task or feature needed to get that result.

The CI team also realized that users needed different amounts of information depending on their learning styles and their proficiency with an application. Part of the team's vision for user assistance included an option to obtain more information (More Info). This option would give users a way to expand interface items (i.e., text and icons) to get additional information. For example, users who did not understand a long prompt could expand the information to a full paragraph, including steps or examples if needed. In addition to the More Info option, the CI team felt that a search mechanism that allowed users to ask questions or search for information using their own terms was critical to overcoming several of the language issues identified from the data. Part of the team's vision included a Natural Language Interface and online help that could match user terminology with terms used by the application (a feature that has subsequently appeared in several software and Internet applications but was seldom seen at the time of the study).

In looking at the collected and consolidated data—particularly the data on learning and problem-solving strategies—the CI team, through visioning, tried to invent ways to support the issues and strategies identified in the data. For example, read-and-follow learners could use the Cue Card—a program prompt that would lead them step-by-step through a task. Or, considering the importance of tip sheets to users, the CI team looked for ways applications could support the generation and distribution of user-created tip sheets. With the Tip-Sheet Manager, users would have a formalized method of creating custom-made documentation. For instance, if a user completed a task, the system would record the steps taken to complete that task. The user-created tip sheet could then be saved, printed, displayed in an online help screen, or sent to someone else. Or, knowing from the data that some users preferred using printed documentation, the team felt that the application should ship with a printed manual. Part of the team's visioning involved brainstorming about a manual that would lend itself to the task orientation of users in a widely recognized format: the cookbook. The metaphor suggests that, like recipes for cooking (which are task oriented and easily shared), users could use or create recipes for tasks they wished to complete. These task recipes could be added to, discarded, or shared depending on the individual users' needs and preferences.

Building a User Environment

The concepts of user assistance that resulted from the visioning process (only discussed in part because of the proprietary nature of the information) then became a basis for building an actual user environment. With design ideas in mind, the team relied on developers to help translate the vision into the reality of a functioning application. Many of the vision ideas required the development of fundamental program features. The CI team developed a prototype of a user environment that contained information and support options based on the vision. The prototype, shown in Figure 5, provided an overview of how the various types of user assistance related to one another.

The foundation of the vision for user assistance centered on the Watcher and the Communicator. The Watcher gathered information about the user through an awareness of the users' actions, but that information could also be altered through specifically selected preferences that a user could change as desired. The dotted line surrounding the Watcher indicates its transparency to the user: The user only interacted with the application through the Communicator, which in turn interfaced with the Watcher. Although the Communicator served as the main point of assistance through which help options were communicated to the user, users could select other options directly without using the Communicator. Aspects of the help options that are independent of the system—"List of Help People/ Resources" and "Third-Party Information"—are also surrounded by a dotted line, in that these options only provided a link to the actual assistance from outside of the system. Besides the actual help options, the application assisted users in creating additional assistance through the Tip-Sheet Manager and Map Maker tools.

After developing an initial prototype of the user environment, the team wanted to relate it to users' actual work. The consolidated sequence models from the interpretive sessions provided a look at specific instances in which users' work processes broke down. The CI team members looked at ways their vision ideas reflected in the new user environment could help users with breakdowns. For example,

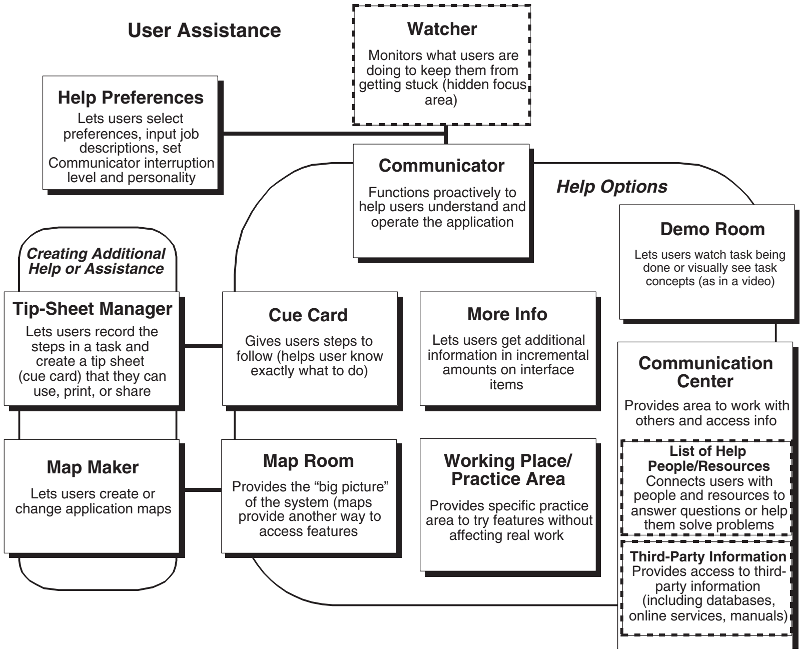

User Assistance
Watcher
Monitors what users are doing to keep them from getting stuck (hidden focus area)
Help Preferences
Lets users select preferences, input job descriptions, set Communicator interruption level and personality
Communicator
Functions proactively to help users understand and operate the application
Help Options
Demo Room
Lets users watch task being done or visually see task concepts (as in a video)
Creating Additional Help or Assistance
Tip-Sheet Manager
Lets users record the steps in a task and create a tip sheet (cue card) that they can use, print, or share
Cue Card
Gives users steps to follow (helps user know exactly what to do)
More Info
Lets users get additional information in incremental amounts on interface items
Communication Center
Provides area to work with others and access info
List of Help People/Resources
Connects users with people and resources to answer questions or help them solve problems
Map Maker
Lets users create or change application maps
Map Room
Provides the "big picture" of the system (maps provide another way to access features
Working Place/ Practice Area
Provides specific practice area to try features without affecting real work
Third-Party Information
Provides access to third-party information (including databases, online services, manuals)

Figure 5. Prototype Based on the CI Team's Concept of User Assistance in a User Environment

Figure 4 shows a consolidated sequence model of ways individuals get unstuck by using information they have found. The rectangular boxes in the model show various steps the users may take, with breakdowns in the sequence indicated in the diamond-shaped boxes. For instance, users may start by finding information they think is correct for learning a new feature or getting unstuck. They then read that information. Frequently, users will hypothesize a solution from the information. Users then generally write down the steps and either close the information source or leave it open for further reference. As they follows the steps, several possible scenarios arise (the diamond-shaped boxes) that keep users from successfully completing the task. Users may do the following:

- jump into the middle of the steps
- forget or skip a step
- continue a sequence of mistaken steps repeatedly
- encounter an unexpected happening

- get an error message
- misinterpret information and use it wrongly
- be unable to get enough information
- fail because the system lacks a necessary item to complete the task
- decide to ignore the problem
- give up

The consolidated sequence model also shows where users jump to other sequences (keystone-shaped boxes) or turn to other sources of help (the rounded rectangles) when they cannot get unstuck by following the original steps.

The CI team looked at the breakdowns (diamond-shaped boxes) in this consolidated sequence model as points where the system needed redesigning to provide better support by offering relevant information to help users to complete the steps and the related task successfully. For example, users often turned to an outside source for additional information—from a colleague or podmate to a help desk or product-support service. As noted previously, related CI data found that information that is closer to the user is more reliable and less costly than is information that is farther away. These data supported the idea of creating the Watcher/Communicator, an electronic assistant close to the user and aware of the users' context. In analyzing the sequence models, the CI team members looked at how their new user environment could respond to the breakdowns.

Storyboarding

As the team members found ways to respond to breakdowns, they integrated the task sequences with the new user environment through a process Beyer and Holtzblatt call storyboarding (287-89). A storyboard takes a specific task from the consolidated sequence models and demonstrates how it will be accomplished in the new environment. In essence, the storyboarding process lets the team redesign the consolidated sequence models to respond to breakdowns by integrating the new user environment, an approach that has become more widely used in software design (Carlshamre and Karlsson; Jacobson). During storyboarding, the team analyzed each of the breakdowns and looked at ways the system or documentation could assist users who encountered problems. The storyboards visually looked like the consolidated sequence models, with the addition of features from the new user environment integrated into the sequences. The storyboarding provided the first opportunity for the CI team members to test their vision of a system with actual task sequences. The storyboarding provided concrete structure to the team's vision and the starting point for prototyping specific application features.

Prototyping

With the new user environment connected to user tasks through storyboards, the team then wanted to test specific features of the proposed assistance with actual users, a procedure called *prototyping*. Because contextual design, as an iterative method, seeks user feedback at several stages of the design process, prototyping has become an effective and economical way to gather customer feedback on design ideas (Beyer and Holtzblatt; Hackos and Redish). Through low-fidelity prototyping, the CI team created paper prototypes of several assistance features of the new user environment, such as the map room and cue cards. Users were then asked to complete tasks using the features represented in the prototype. These prototypes allowed the team to validate design ideas and to test the new ideas for workability with actual users.

As the CI team members tested and refined the design of the user environment through prototyping, they realized that the entire new user environment (with its innovative methods of user assistance) could not practically be implemented fully in the next product release. Consequently, the team (in consultation with development) selected those features from the new environment that could be integrated into the next release. With the assistance of several developers, the CI team finalized these features that would be part of the next release and then developed corresponding detailed design specifications. Although financial, technological, and time constraints required that some features be implemented in future releases, the design specs (along with the prototype of the new user environment) became the blueprint from which development efforts ensued. The design specifications provided a concrete plan for the software, interface, and documentation requirements for the next release, and the new user environment gave a direction for development in future releases.

Although many aspects of the envisioned user assistance could only be minimally implemented (such as a limited Natural Language Interface for help searches and visual indexes) due to several constraints, the vision gave developers an idea of what the team proposed in subsequent releases. The design ideas that resulted from the contextual design process demonstrate some of the innovative ways documentation can assist users, innovations that have begun to appear in software applications from various companies both in recent software releases and in Internet applications. By analyzing some of the constraints imposed on the CI team, we discover the challenges any team encounters when implementing new ideas.

DEALING WITH CONSTRAINTS AND RESISTANCE

As Norman observes, design ultimately must "take into account real issues in cost, manufacturability, and aesthetics" (*Design* 3). As in any organization, certain constraints prevented the CI team from implementing immediately the entire vision of a new user environment. The factors that most affected the implementation of the vision were (1) technological constraints, (2) potential user resistance, and (3) corporate constraints and resistance.

Technological Constraints

An important part of the new user environment included an intelligent system that the team described as both pulling and pushing information. The team found that to truly respond to users, a system needed to pull, or gather, information from users. But, in addition, when the system found a user stuck (such as repeating an action three or more times while unsuccessfully completing a task—the magic sequence), the system needed to push, or bring, information or assistance to the user.

Several technological challenges prohibited a complete pulling and pushing of user information. At the time of development, for instance, the team still had hardware constraints surrounding processor speed, connectivity, download speeds, and memory (gigabyte storage devices were not common at the time of the study). In addition, the standard operating system was not enough aware of user events at the time for developers to implement completely the Watcher and Communicator ideas, and the use of intelligent applications to collect and analyze information was limited. Moreover, the Internet was just becoming a significant medium in businesses and the workplace. For some components of user assistance to work—

such as the Communication Center—the application required a majority of users to be connected to the Internet so that online resources and data could be seamlessly retrieved from the Web. These types of constraints kept many aspects of the vision from being implemented immediately.

Although some of these technical constraints prevented the CI team's vision from being fully implemented in the next product release, the constraints did not invalidate the usefulness or feasibility of the ideas. For example, the Watcher and the Communicator concepts represented a more self-aware system than had been evident in any home and business office products on the market at the time of the CI work. We have since seen the implementation of limited intelligent agents in applications such as *Microsoft Office*. Although its ubiquitous (and frequently criticized) Paper-clip does not embody the full vision of the CI team's Watcher and Communicator, it shows the feasibility of implementing such features in applications. Also, intelligent agents have become a growing part of consumer interaction on the Internet, performing many of the functions envisioned with the Watcher and the Communicator (Iacobucci, Arabie, and Bodapati; Seol; Setton). Just as the Paper-clip and Internet agents continue to evolve, the CI team envisioned the Watcher and the Communicator as design ideas that future iterations would work toward as technology advanced. As new technologies develop and systems become incrementally more aware of users, additional aspects of the vision could be added.

Potential User Resistance

Even if all aspects of the CI team's vision had been technologically possible, the team was aware that several parts of the new user experience might not be palatable to some users. Certainly, challenges remain in successfully implementing the use of intelligent agents in software even as technology makes it possible. One of the controversies concerning Microsoft's Paper-clip surrounds the unsolicited pushing of information. For instance, when *Office 97* users perform a seldom-used task, the Paper-clip often appears without being asked. Many users have complained about this unsolicited appearance of the agent: "It pops up without warning—even when you don't want it or don't need it. First it startles you, then it takes over your computer." "Paper-clip wouldn't be so obnoxious if you could control it. As is, it moves all over the place and gets in the way" (Shroyer 238).

Although *Microsoft Office 2000* has a reformed, more mild-mannered Paper-clip that sits in the corner of the screen, it does not meet the challenge of providing personalized information for specific users in unique contexts. The difficulty of the challenge, however, does not diminish the desire of users to have assistance that is "tailored to them, delivered in a way that is polite, accessible and easy to understand" (McKeon 33) nor does it attenuate a development need or effort to implement agent technologies. Recent strides in self-customizing software have, however, demonstrated a greater understanding of user behavior and sense making as systems interact with users, observe patterns of user behavior, and personalize information for specific users (Hirsh, Basu, and Davison). In addition, advances in mixed-initiative computing—computing that interleaves contributions by users with automated, computer-generated activities—have also shown how computers can successfully gather information and make inferences about "intentions, attention, and competencies of users" to provide automated service and customized information (Horvitz 17).

Besides being aware of the need to provide the appropriate information to users through agents, the CI team also realized that some users might view a system that constantly watched or recorded their actions as a potential threat. For instance, if management used the Watcher to monitor employees' work habits and use of company computers, users would resist it—especially if they feared the information gathered might be used against them. Also, users may question the need for a system to collect data on their work habits and behaviors at all. To function effectively, the Communicator needed to use information gathered about user actions and preferences as well as about the context of a user's work. The CI team was concerned that users would neither like nor permit such data mining—a concern validated in several instances, such as the negative reaction toward RealNetwork's gathering and using information about the music tastes and preferences of consumers (Null 44) or DoubleClick's using information it gleaned from user profiles for commercial purposes (Alpert 16). The CI team realized that this type of potential user resistance needed to be addressed in the design and development of the new system.

Corporate Constraints and Resistance

In addition to the technological constraints and potential user resistance, the CI team also encountered corporate constraints and resistance. The CI team's vision of the new user environment required significant development resources. The team's plan to design not just a new manual or help system but significant application features required a large commitment of time, money, and people. Development resources were needed for building features such as the Watcher, the Communicator, and the Tip-Sheet Manager.

Other constraints included cultural and organizational issues. The contextual design process was relatively new to the company, and acceptance by development, management, and others within the company took time: It required of individuals a new way of thinking and doing, forcing them to be involved in new processes in new ways. For instance, individuals from technical documentation had never had such a significant or early role in the development process. The CI team members needed to use artful persuasion in seeking support within the company for their new ideas and methods.

Interestingly enough, some of the greatest resistance came from others in the documentation department who had not been intimately involved in the process. Although the CI team's vision was never intended for wholesale implementation in one release, even its incremental implementation threatened the traditional way of doing things. Some technical writers and editors resisted working on interface issues and documentation integral to the application, such as long prompts and error messages. Others who had failed to keep current by reading journals and attending conferences were threatened by innovative ways of doing things. Some in the department who had resisted the implementation of online help also resisted the vision of a new user environment built on the foundation of contextual inquiry. Collectively, the documentation department had established itself as a producer of printed manuals and online help, and some felt threatened by the new skills and roles required of them by the CI vision. Others in the department, however, readily embraced and supported the contextual design process and the results of the CI work.

Perhaps the major organizational constraint affecting the CI work resulted from a business event totally independent of development and the data: The company was sold. Although the new company ultimately endorsed the work of the CI team, new managers and personnel needed to be convinced of the value of the contextual design approach. The organizational transition curtailed the implementation of the vision, at least in the short term.

Ultimately, some features the CI team proposed were implemented in the next product release. Although the vision of the total user environment was not realized, several important changes were made. For the next product release, the team was able to include a limited intelligent agent that assisted users in certain tasks. The documentation group redesigned error messages and implemented pop-ups to display critical information, both ideas grounded in the CI data. Also, elements of a Natural Language Interface were integrated, particularly in the help system (building on the data that users want information in their terms, in language they use and understand).

IMPLICATIONS FOR PRACTICE AND RESEARCH

In this article, we have detailed the way the contextual design method was applied to information and documentation design and reported some of the results of the contextual inquiry process, including insights gleaned about users' learning and problem-solving strategies. Although the proprietary nature of much of the data as well as space limitations have prevented us from detailing every aspect of the process or its results, the study nevertheless generates important implications both for practice and for further research.

Implications for Practice

The CI data have important implications for documentation and application design. One of the most important insights gained by the CI team was the need for the team (and, subsequently, the company) to expand its definition of documentation. The CI team realized that documentation and application information is not just user assistance but part of a broader concept of *user experience*. Designing for users' experience suggests that information and assistance be seamlessly and unobtrusively integrated into the application. Recent research suggests that this notion of the user experience is becoming a significant factor in developing, differentiating, and achieving successful products and in creating value for customers (Laurel; Pine and Gilmore). Too frequently, documentation is viewed as a printed manual and an online help system, a view often held and encouraged by technical communicators themselves. As JoAnn Hackos and Janice Redish observe, information and instruction occur at many levels: the interface (menus, icons, labels), interface messages (tool tips, error messages, button help), aids to doing (wizards, agents), aids to learning and doing (cue cards, coaches), and embedded context-sensitive help and tutorials (408). Documentation is not just about user education or assistance; rather, it involves the entire user experience. To create truly user-centered applications requires the collaboration of developers, interface designers, usability specialists, and technical communicators.

Although the CI team members found data that would help them in creating better manuals and help systems, as important, they found that their understanding and appreciation of users' experience grew as they worked closely with their customers. As Marlana Coe notes, technical communicators' biggest challenge is not in using tools and technology but in creating active partnerships with users (177). By working intimately with users, the CI team quickly expanded its vision of user assistance to a systemwide proportion that involved the whole user experience, of which a manual or help system was but a part. The team members realized that the interface and documentation are integrally linked, that the way things are named and ordered is integral to users' experience, that menus are a first level of documentation, and that error messages and system prompts are significant points of system interaction where users learn a system or recover from system problems. They grew to understand that printed manuals, online help, Web support, tutorials—what we traditionally view as documentation—are all part of the interface: the bridge between the application and the user. The CI team came to realize that genuinely successful applications need to reflect the work flow of users, support various users' learning styles, and remain compatible to users' working environment and language (Hackos and Redish 5-7).

In addition to working with customers, the CI team spent much time developing ideas—ideas that were in turn further developed and refined through iterative testing with users. Although the importance of customer contact cannot be overestimated, the time allowed for visioning and for exploring and testing options is critical to innovation and to finding ways to enhance a user's experience. Part of the CI team's success came from an often-overlooked practice: allowing time for reflection. Taking the time to reflect—to think through and explore options—was crucial for the CI team. Innovations require percolating time for ideas to germinate, and teams and individuals need time set aside to reflect on what they have done and how they have done their work. Reflecting on ideas, on users and their work, and on past performance enabled the CI team to find innovative ways to support users and enhance their experience with application software. Many software developers and technical communication groups participate in what the software industry calls a postmortem at the completion of a project. Taking time to evaluate the success or failure of a project—what went well and what did not—provides pertinent information that can be used to improve processes. But allowing time for reflection during a project is also helpful for coming up with innovative solutions. Successful teams need time for team members to reflect both individually and together.

The CI team also found in conducting this study that the contextual design methodology was a valuable process for gathering customer data in a systematic, usable way and that information about users and their work was best gathered through observing and working with users in their own environment in the context of their own work. The contextual design process has application for technical communicators in producing documentation, not just for developers of systems and applications. Frequently, technical communicators view themselves as users' advocates, with the mistaken notion that, as nondevelopers who also use an application designed by someone else, they know what users want and need. The CI team members found, however, that if they truly wanted to be user advocates, they needed direct contact with users, and the contextual design process provided that direct contact that allowed them to find out what users actually want and need.

The CI team observed that many technical communicators had difficulty thinking strategically, or outside of the box, and realizing they could and should do more than just write or edit. Particularly strong resistance to change came from other writers and editors within the documentation department. As many have advocated, technical communicators need to be willing to learn new skills and be more involved in the whole development process (Bradford; Bresko; Fisher; Vaughn and Walton), an effort that would require them to push out of their comfort zones and be willing to make changes in the kind of work they do and how they do it. Because of the growing importance of information design, technical communicators have a better opportunity to position themselves as strategic players within organizations, to understand and support the way people interact with applications. As organizations move toward increasing customer satisfaction, technical communicators (or information developers) can have a significant impact on the value of products through improving users' experience with them.

The resistance of some technical communicators and others within ProQuest to the CI data and process underscores the importance of effective communication in any design effort or process innovation. To achieve success, the CI team had to be concerned about more than just user data and design ideas. Any new idea, process, or product must establish its credibility, and an important part of any innovation is to communicate effectively to those who have a stake in the project. Although significant changes often require a strategic vision from the top, the implementation of new ideas frequently comes from the outside in and from the bottom up. Often, new ideas or approaches fail not because of their validity or value but because of "organizational reluctance to change" (Smart 312). Because "people are invested in their current ways of doing things" (Beyer and Holtzblatt 432), CI efforts such as the one described (or any innovations) need to involve others throughout the entire process and require a proactive approach by innovators in trying to get others to accept their methodology or results.

The CI team members did succeed in many ways in marketing their ideas to others. For instance, the team's design room (where the affinity diagram and consolidated models hung on the walls) became an important showcase for the work. Frequently, the CI team brought developers, marketing representatives, technical communicators, and others to the design room for a walk-through of the affinity diagram and the consolidated models. These walk-throughs helped acquaint others with the data and the processes the team used. In addition, the CI team members occasionally involved others not on the team in some of the site work with customers, particularly as they tested prototypes and the new user environment. Such direct involvement with users helped others understand customer needs at a more visceral level, which increased support for the objectives of the CI team. Also, the CI team members frequently made both formal and informal presentations to others within ProQuest, tailoring their presentations and language to the needs of respective audiences—showing marketing how the product met needs of users who would be purchasing the product, demonstrating to developers the structure of the system and how it worked, presenting management with milestones and deliverables while differentiating short-term improvements from long-term directions and strategies. These types of practices that helped the CI team succeed are critical for technical communicators and others involved in effecting change within their organizations.

The success of the CI team underscores the movement toward working in teams in today's workplace. The importance of teams has grown in corporate environments during the past few decades as an increasing number of organizations have turned to collaborative models of work. In like manner, technical communicators have become more involved in teams as organizations have sought ways to leverage resources and improve work products and systems. The impressive results of teams in organizations have bolstered their popularity. The effective use of teams can bring potential advantages to the workplace:

- enhanced communication and decision making through rich sharing of information and leveraging a wider base of knowledge and experience (i.e., more and richer information shared sooner and faster)
- increased productivity with higher levels of involvement, commitment, motivation, and subsequent accountability among employees
- improved processes, building on diverse backgrounds and experiences
- distributed workloads responding to situations where problems and tasks have become too large for one individual

As the complexity of work increases, the movement toward teams grows inevitable, as successful development requires a cross-functional representation of varying specialties and expertise. Products and services consumed in today's marketplace have become too complex for solitary individuals to design, create, or provide—requiring the skills and knowledge of numerous individuals from various functions or areas within companies. Increasingly, technical communicators have been included and must continue to be involved as members of cross-functional teams within companies (Henry 207; Forbes 117).

In addition to implications about the work of technical communicators and the importance of involving cross-functional groups in any change effort, this study provides important insights about how we develop assistance for users and the value of user-centered design. For instance, users want and need information in their own terms, and applications need ways to allow users to search for support using their own language. Natural Language Interfaces work toward this end, allowing users to type queries in their own words. Moreover, indexes (as an important entry strategy) need to cross-reference terms extensively, building on feature and task synonyms of users. Also, visual indexes that show elements of the program or results of tasks using the application could identify key features for users. Additional strategies and techniques, such as those that would allow users to point to and identify features, could further assist users in determining how to accomplish desired tasks. For example, a system that allows users to tell the computer what they want to do by pointing to an object on screen and then having the correct procedural information appear on the screen would go a long way in assisting users who have had little experience with software applications or who want to know a specific application's terms for procedures.

Furthermore, applications need to be designed to be aware of users' needs and varying preferences. For instance, because users create personalized documentation according to varying contexts and needs, applications need to assist users in creating, saving, and sharing the documentation they create. As a system becomes more aware of users' actions, it can track their work and the steps they took in doing that work to create tip sheets or cue cards that become a formalized part of the users' experience; a system with an increased awareness of users' preferences and learning styles can help provide more personalized and appropriate assistance for given contexts and needs.

The user-centered methodology of contextual design allowed the CI team, and subsequently the company, to gather information and to focus on issues important to users. The study contributed to a better understanding of the differences between users of any system, clarifying the need to approach and support users differently. Well-designed systems must include enough flexibility to allow situational learning and support that vary according to individual users and user needs. Successful applications are designed to match and support user needs.

The results of the study also reinforced the importance of the interface, including features such as long prompts and error messages that frequently receive less attention. Companies should carefully develop those aspects of the interface that support users at critical junctures, such as learning a new feature or recovering from an error. In relationship to documentation, the study emphasized the vital importance of the index—another feature that often does not receive enough attention. Technical communicators frequently wait until late in the production cycle to create indexes. Their value to users suggests that they should be an ongoing part of the documentation effort, helping to frame the system in users' own terms—ones they understand and value.

The information gathered from this study and from other user-centered design methods has various applications and potential outputs as we try to find better ways to support users' learning and problem-solving strategies. Technical communicators and system developers can use data on customers to modify existing tools, systems, and documents and to develop new applications, systems, and documentation, as well as to assist in developing new work practices. Such work, however requires an awareness of, and working within, organizational constraints as well as awareness of the customer. The CI team members realized that even with great ideas, they needed the understanding and approval of others in the organization. Design teams need to assess which ideas can be implemented immediately and which ones can wait until later versions. For instance, although the CI team could not implement a complete system that tracked and communicated with users, the team was able to develop a more extensive index for printed manuals and to implement a limited Natural Language Interface in the Index/Search feature of the help system, which moved the product toward providing assistance in the user's own language.

Implications for Research

The CI data also have important implications for additional research. These data captured information about users at a specific time. Any user-centered approach is iterative and requires continual work with the customer. As technological advances make possible the CI team's vision, the data must be validated through additional contextual inquiry and user testing. Technology will continue to allow new methods and ideas to alter the work and work practices of users. For instance, what does the idea of continually upgradeable software applications (rented via the Internet) do to the traditional model of software documentation that is rewritten once during every 18-month product cycle? Will the ability to provide updates to documentation instantly add value to users?

The CI team gathered information about how people learn and how they get unstuck, but additional contextual inquiry could provide insights into the ways users perform tasks. Although considerable information about how users perform tasks was gathered tangentially, a focused study on the way users complete tasks would provide additional clues for enhancing a user's experience. Additional research can also target users' learning styles and strategies with specific tasks. What type of information in what form promotes learning for what type of users? Which learning styles are the most effective in which situations?

A continued thrust of user assistance has been to provide more user interaction with computers. The use of electronic assistance that gives timely and correct information continues to grow. The concepts of the Watcher and the Communicator provide a vision for intelligent agents whose functions are suggested by others: to critique, to tutor, and to explain (Fischer). Although initial research indicates users react positively to human-like social interfaces (Sproul et al.), our findings confirm other research that suggests anthropomorphic representations in agents often cause problems and unrealistic expectations, frequently annoying or distracting users (Bates; Norman, "How").

Further work needs to explore how users might best interact with systems and how agents may facilitate that interaction (Riecken; Soloway and Pryer). Can users interact with systems directly rather than through anthropomorphized objects? Microsoft's Bob and Paper-clip are examples of intelligent assistance systems that were not well received by most users (Coursey; Li-Ron; Smith). What can be learned from that experience and the experience of others? What can a contextual inquiry tell us about agents, what users think of them, and what they want from them? Should agents have or display emotion? How will users learn to trust agents and instruct them when they make mistakes? Should an agent be human-like or merely display information as part of an application? How might agents teach people to learn? Can they improve the performance of users? Additional research can help answer such questions and help with the implementation of ideas such as the Watcher and the Communicator.

More fundamental than questions about the use of agents are basic questions about human learning. In a rapidly changing world, users continually need to learn new tools, procedures, and methods although they do not have time for formal learning. How can we facilitate learning? How can we apply what we know about learning and learning strategies to technical communication and information development? What constitutes a "safe" environment that encourages users to explore and apply learning?

In addition to these research issues, one of the biggest implications of the study involves how we view and define the role of technical communicators. The contextual design experience took a group of traditional writers and editors and thrust them into new roles that required them to learn new skills. The experience caused a heated discussion about what being a technical communicator means. How do we define ourselves? What skills are essential? Is the discipline evolving to adapt to changing informational needs? If technical communicators do not rise to the occasion to perform the type of work done by the CI team, others will.

As we move into what has been called the information age, technical communicators have the unique opportunity of helping us understand how humans interact with computers and other machines and of finding innovative ways to facilitate and assist that interaction. This article shows a method for gathering data about customers and how to translate that information into actual design. Such methods increase the likelihood of designing products and systems that reflect and meet customers' needs and enhance users' overall experience with technology. Technical communicators must learn to understand and implement such methods to remain competitive and increase their own value within organizations. The more we truly understand and advocate the needs of users, the greater our value—and the more value we add to the workday lives of our users.

REFERENCES

Alpert, Bill. "TroubleClick." *Barron's* 3 July 2000: 16.

Bailey, Robert W. *Human Errors in Computer Systems*. Englewood Cliffs, NJ: PrenticeHall, 1983.

Bates, Joseph. "The Role of Emotion in Believable Agents." *Communications of the ACM* 37.7 (1994): 122-25.

Beabes, Minette A., and Alicia Flanders. "Experiences with Using Contextual Inquiry to Design Information." *Technical Communication* 42 (1995): 409-20.

Bernhardt, Stephen A., and George M. McCulley. "Knowledge Management and Pharmaceutical Development Teams: Using Writing to Guide Science." *Technical Communication* 47 (2000): 22-34.

Beyer, Hugh, and Karen Holtzblatt. *Contextual Design Defining Customer-Centered Systems*. San Francisco: Morgan Kaufmann, 1998.

Bostrom, Robert P., Lorne Olfman, and Maung K. Sein. "The Importance of Learning Style in End-User Training." *MIS Quarterly* 14 (1990): 100-19.

Bradford, David B. "The New Role of Technical Communication." *Technical Communication* 32 (1991): 13-15.

Brassard, Michael. *Memory Jogger Plus*. Methuen, MA: GOAL/QPC, 1989.

Bresko, Laura L. "The Need for Technical Communicators on the Software Development Team." *Technical Communication* 38 (1991): 214-20.

Carlshamre, P., and J. Karlsson. “A Usability-Oriented Approach to Requirements Engineering.” *Proceedings of the Second International Conference on Requirements Engineering*. Los Alamitos, CA: IEEE Computer Society Press, 1996. 145.

Carroll, John M. *Minimalism beyond the Nurnberg Funnel*. Cambridge, MA: MIT Press, 1999.

———. *Scenario-Based Design: Envisioning Work and Technology in System Development*. New York: John Wiley, 1995.

Carroll, John M., and Judith Reitman Olson. “Mental Models in Human-Computer Interaction.” *Handbook of Human-Computer Interaction*. Ed. Martin Helander. New York: Elsevier Science, 1988. 45-65.

Carroll, John M., and Mary Beth Rosson. “The Paradox of the Active User.” *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*. Ed. John M. Carroll. Cambridge, MA: MIT Press, 1987. 80-111.

Charney, Davida H., Lynne M. Reder, and Gail W. Wells. “Studies of Elaboration in Instructional Texts.” *Effective Documentation: What We Have Learned from Research*. Ed. Stephen Doheny-Farina. Cambridge, MA: MIT Press, 1988.

Clark, Margaret Sydnor, and Susan T. Fiske. *Affect and Cognition*. Hillsdale, NJ: Lawrence Erlbaum, 1982.

Coe, Marlana. *Human Factors for Technical Communicators*. New York: John Wiley, 1996.

Combs, Arthur W. “Affective Education or None at All.” *Educational Leadership* 39 (1982): 494-97.

Coursey, David. “Like It or Not, Here Comes Bob 97.” *Computerworld* 19 Aug. 1996: 101-02.

Cross, K. Patricia. *Adults as Learners*. San Francisco: Jossey-Bass, 1981.

Den Buurman, Rudy. “User-Centered Design of Smart Products.” *Ergonomics* 40 (1997): 1159-69.

Denzin, Norman K., and Yvonna S. Lincoln, eds. *Strategies of Qualitative Inquiry*. Thousand Oaks, CA: Sage, 1998.

Fischer, Gerhard. “Enhancing Incremental Learning Processes with Knowledge-Based Systems.” *Learning Issues for Intelligent Tutoring Systems*. Ed. Heinz Mandl and Alan Lesgold. New York: Springer-Verlag, 1988.

Fisher, Julie. “The Value of Technical Communicator’s Role in the Development of Information Systems.” *IEEE Transactions on Professional Communication* 42 (1999): 145-54.

Forbes, Christopher. “The Role of the Technical Communicator within Organizational Information Development Cycles.” *Publications Management Essays for Professional Communicators*. Ed. O. Jane Allen and Lynn H. Deming. Amityville, NY: Baywood, 1994. 117-40.

Gerlach, James H., and Feng-Yang Kuo. “Understanding Human-Computer Interaction for Information Systems Design.” *MIS Quarterly* 15 (1991): 526-48.

Grech, Christine. “Computer Documentation Doesn’t Pass Muster.” *PC Computing* 5.4 (1992): 212-14.

Hackos, JoAnn T., and Janice C. Redish. *User and Task Analysis for Interface Design*. New York: John Wiley, 1998.

Henry, Jim. “The Value Added by Technical Communicators in Collaborative Writing Situations.” *Technical Communication* 45 (1998): 207-20.

Hirsh, Haym, Chumki Basu, and Brian D. Davison. “Learning to Personalize.” *Communications of the ACM* 43.8 (2000): 102-06.

Hodder, Ian. “The Interpretation of Documents and Material Culture.” *Collecting and Interpreting Qualitative Materials*. Ed. Norman K. Denzin and Yvonna S. Lincoln. Thousand Oaks, CA: Sage, 1998. 110-29.

Holtzblatt, Karen, and Hugh Beyer. “Making Customer-Centered Design Work in Teams.” *Communications of the ACM* 36.10 (1993): 92-104.

Horvitz, Eric. “Uncertainty, Action, and Interaction: In Pursuit of Mixed-Initiative Computing.” *Intelligent Systems* 14.5 (1999): 17-20.

Huberman, A. Michael, and Matthew B. Miles. “Data Management and Analysis Methods.” *Collecting and Interpreting Qualitative Materials*. Ed. Norman K. Denzin and Yvonna S. Lincoln. Thousand Oaks, CA: Sage, 1998, 179-210.

Iacobucci, Dawn, Phipps Arabie, and Anand Bodapati. “Recommendation Agents on the Internet.” *Journal of Interactive Marketing* 14.3 (2000): 2-11.

Irvin, Steve. “Documentation.” *InfoWorld* 13 Nov. 1995: 132.

Jacobson, Ivar. *Object-Oriented Software Engineering*. Reading, MA: Addison-Wesley, 1994.

Jonassen, David H. “Operationalizing Mental Models: Strategies for Assessing Mental Models to Support Meaningful Learning and Design-Supportive Learning Environments.” Computer Support for Collaborative Learning 95 Conference. Bloomington, IN. 1995. 21 Oct. 2001. http://www.cica.indiana.edu/cscI95/jonassen.html

Kawakita, Jiro. *The Original KJ Method*. Tokyo: Kawakita Research Institute, 1982.

Keil, Mark, and Erran Carmel. “Customer-Developed Links in Software Development.” *Communications of the ACM* 38.5 (1995): 33-44.

Kolb, David A. *Experiential Learning: Experience as a Source of Learning and Development*. Englewood Cliffs, NJ: Prentice Hall, 1984.

Laurel, Brenda. *Computer as Theater*. Reading, MA: Addison-Wesley, 1991.

Lawler, Patricia A. *The Keys to Adult Learning: Theory and Practical Strategies*. Philadelphia: Research for Better Schools, 1991.

Leonard, Dorothy, and Jeffrey F. Rayport. “Spark Innovation through Empathic Design.” *Harvard Business Review* Nov.-Dec. 1997: 102-13.

Li-Ron, Yael. “Office Assistant: Dog or Genius?” *PC World* 15.2 (1997): 116.

Mandler, George. *Mind and Body*. New York: Norton, 1984.

McKeon, Anne. “Intelligent Assistance and Natural Language Processing Drive New AI.” *Computing Canada* 24.2 (1998): 33.

Mead, Jay. “Measuring the Value Added by Technical Documentation: A Review of Research and Practice.” *Technical Communication* 45 (1998): 357-80.

Nielsen, Jakob. *Usability Engineering*. San Francisco: Morgan Kaufmann, 1993.

Norman, Donald A. *The Design of Everyday Things*. New York: Doubleday/Currency, 1989.

———. “How Might People Interact with Agents.” *Communications of the ACM* 37.7 (1993): 68-71.

———. “Some Observations on Mental Models.” *Mental Models*. Ed. Dedre Gentner and Albert L. Stevens. Hillsdale, NJ: Lawrence Erlbaum, 1983.

Norton, David W. “Tech Comm as Business Strategy: A Study of How Changes in Discursive Patterns Affect the Value of Technical Communication in Cross-Functional Team Settings.” *Technical Communication* 47 (2000): 77-89.

Null, Christopher. “Someone to Watch Over You.” *PC Computing* 13.2 (2000): 44.

Parasuraman, A. “Reflections on Gaining Competitive Advantage through Customer Value.” *Journal of the Academy of Marketing Science* 25.2 (1997): 154-61.

Pask, Gordon. *Conversation, Cognition, and Learning*. New York: Elsevier, 1975.

———. “Styles and Strategies of Learning.” *British Journal of Educational Psychology* 46 (1976): 128-48.

Pine, B. Joseph, and James H. Gilmore. *The Experience Economy: Work Is Theater and Every Business a Stage*. Boston: Harvard Business School Press, 1999.

Raven, Mary Elizabeth, and Alicia Flanders. “Using Contextual Inquiry to Learn about Your Audiences.” *Journal of Computer Documentation* 20 (1996): 1-13.

Reason, Peter. “Three Approaches to Participative Inquiry.” *Strategies of Qualitative Inquiry*. Ed. Norman K. Denzin and Yvonna S. Lincoln. Thousand Oaks, CA: Sage, 1998, 261-91.

Redish, Janice C. “Adding Value as a Professional Technical Communicator.” *Technical Communication* 42 (1995): 26-29.

———. “Minimalism in Technical Communication: Some Issues to Consider.” *Minimalism beyond the Nurnberg Funnel*. Ed. John M. Carroll. Cambridge, MA: MIT Press, 1998. 219-46.

———. “Reading to Learn to Do.” *IEEE Transactions on Professional Communication* 32 (1989): 289-93.

Rettig, Marc. “Nobody Reads Documentation.” *Communications of the ACM* 34.7 (1991): 19-24.

Riecken, Doug. “Intelligent Agents.” *Communications of the ACM* 37.7 (1994): 18-20.

Rogers, Carl R., and H. Jerome Freiberg. *Freedom to Learn*. 3rd ed. Columbus, OH: Merill/Macmillan, 1994.

Rust, Ronald T., and Anthony J. Zahorik. “Customer Satisfaction, Customer Retention, and Market Share.” *Journal of Retailing* 69 (1993): 193-215.

Säljö, Roger. “Learning Approach and Outcome: Some Empirical Observations.” *Instructional Science* 10 (1981): 47-63.

Schneider, M. L. “Models for the Design of Static Software User Assistance.” *Directions in Human/Computer Interaction*. Ed. Albert Badre and Ben Shneiderman. Norwood, NJ: Ablex, 1982.

Schuler, Douglas, and Aki Namioka. *Participatory Design: Principles and Practices.* Hillsdale, NJ: Lawrence Erlbaum, 1993.

Seol, Kap Su. “‘Bots’ Support Customer Service.” *National Underwriter*. 104.4 (2000): 7, 16.

Setton, Dolly. “Invasion of the Virbots.” *Forbes* 11 Sep. 2000: 22-26.

Shroyer, Roberta. “Actual Readers versus Implied Readers: Role Conflicts in Office 97.” *Technical Communication* 47 (2000): 238-40.

Smart, Karl L. “Minimalism: A Quality Strategy for Success.” *Minimalism beyond the Nurnberg Funnel*. Ed. John M. Carroll. Cambridge, MA: MIT Press, 1998. 312-26.

Smart, Karl L., J. L. Madrigal, and Kristie K. Seawright. “The Effect of Documentation on Customer Perception of Product Quality.” *IEEE Transactions on Professional Communication* 39 (1996): 157-62.

Smith, Rebecca M. “Microsoft Bob to Have Little Steam, Analysts Say—Ask if ‘Social Interfaces’ Are the Best Solution.” *Computer Retail Week* 3 Apr. 1995: 37.

Soloway, Elliot, and Amanda Pryer. “The Next Generation of Human-Computer Interaction.” *Communications of the ACM* 39.4 (1996): 16.

Sproull, Lee, Mani Subramani, Sara Kiesler, Janet H. Walker, and Keith Waters. “When the Interface Is a Face.” *Human-Computer Interaction* 11 (1996): 97-124.

Stewart, Thomas. *Intellectual Capital: The New Wealth of Organizations*. New York: Doubleday, 1997.

Suchman, Lucille A. *Plans and Situated Actions: The Problem of Human Machine Communication*. New York: Cambridge University Press, 1987.

———. "Representations of Work." *Communications of the ACM* 38.9 (1995): 56-64.

Van der Meij, Hans. "A Critical Assessment of the Minimalist Approach to Documentation." *Proceedings of SIGDOC 92 Conference*. New York: Association of Computing Machinery, 1992. 7-17.

Van der Meij, Hans, and John M. Carroll. "Principles and Heuristics for Designing Minimalist Instruction." *Minimalism beyond the Nurnberg Funnel*. Ed. John M. Carroll. Cambridge, MA: MIT Press, 1998. 19-53.

Vaughn, Joan E., and Katie Walton. "Beyond End-User Documentation: Opportunities for Technical Communicators." *STC Conference Proceedings.* Arlington, VA: Society for Technical Communication, 1999. 10 Sep. 2001. http://www.stc-va.org/proceedings/ConfProceed/1999/PDFs/STC-80.pdf

Weiner, Bernard. "History of Motivational Research in Education." *Journal of Educational Psychology* 82 (1990): 616-22.

Wood, Larry E. *User Interface Design: Bridging the Gap from User Requirements to Design*. Boca Raton, FL: CRC Press, 1998.

Yourdon, Edward, and Larry Constantine. *Structured Design*. Englewood Cliffs, NJ: Prentice Hall, 1979.

Karl L. Smart teaches business communication courses at Central Michigan University. He has published several articles on user-centered design and issues of quality in professional communication. His current research focuses on applying information design and quality to the Internet. Prior to his academic appointment, he spent several years working in the software industry. He may be reached by e-mail at ksmart@cougar.netutah.net.

Matthew E. Whiting has more than 10 years' experience in the software industry, including documentation-group management, program management, tools-team management, technical writing, testing, and product-system design. He has delivered papers and appeared on panels for several professional organizations. Currently, he is a user-experience group manager for Microsoft Corporation. He may be reached by e-mail at mwhiting@microsoft.com.