ANSWER GARDEN: A TOOL FOR GROWING ORGANIZATIONAL MEMORY

by

MARK S. ACKERMAN

B.A. Honors, History in the Social Sciences, University of Chicago
(1979)
M.S., Computer and Information Science, Ohio State University
(1981)

Submitted to the
ALFRED P. SLOAN SCHOOL OF MANAGEMENT
in partial fulfillment of the requirements for
the degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1994

Signature of Author _____
Alfred P. Sloan School of Management
November 19, 1993

Certified by _____
Thomas W. Malone
Thesis Supervisor

Accepted by _____
James B. Orlin
Chairman, Doctoral Program Committee

Answer Garden: A Tool for Growing Organizational Memory

by

Mark S. Ackerman

Submitted to the
Alfred P. Sloan School of Management
Massachusetts Institute of Technology
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Abstract

Answer Garden allows organizations to develop databases of commonly asked questions that grow "naturally" as new questions arise and are answered. It is designed to help in situations (such as customer "hot lines" or help desks) where there is a continuing stream of questions, many of which occur over and over, but some of which the organization has never seen before. Answer Garden includes a branching network of diagnostic questions, as well as additional information retrieval methods, that help users find the answers they want. If the answer is not present, the system automatically routes the question to the appropriate expert, and the answer is returned to the user as well as inserted into the information database. Experts can also modify this network in response to users' problems. Through their normal interactions, users and experts build an organizational memory.

The thesis examines organizational memory and Answer Garden from three perspectives: in terms of organizational memory at an organizational level, information seeking at an individual level, and software systems at a technical level. It is asserted that information technology can support organizational memory in two ways, either by

making recorded knowledge retrievable or by making individuals with knowledge accessible. The thesis also describes two additional organizational memory applications, the ASSIST and LiveDoc, and details the Answer Garden Substrate system underlying all three applications. Finally, the thesis reports a field study of software engineers' using Answer Garden.

Thesis Supervisor: Thomas W. Malone

Thesis Committee: Wanda J. Orlikowski
John F. Rockart
JoAnne Yates

Table of Contents

8.6. One last word... 161

Acknowledgments

Literally hundreds of people have contributed to the gestation of this thesis. I suspect I should apologize in advance for my poor memory -- I have undoubtedly forgotten important contributors.

First, I must say that I could not have had a better committee for this thesis. Tom Malone deserves the lion's share of my appreciation. Tom not only supported me for five years and gave me a home at the Center for Coordination Science, he provided sharp insights, challenging questions, and proddings at the appropriate times. Most importantly, he put up with me as I shifted back and forth during the process of this thesis, and was a source of general encouragement for the entire project. Tom trusted me to find a suitable path through the intellectual problems and possibilities that the project presented, and I hope I did so. His insights about system and organizational possibilities will continue to play out in my life for many years to come.

Wanda Orlikowski offered a model of qualitative studies and urged socially-aware design. She forced me to think more clearly about the role of system design in social processes and of social processes in system design. JoAnne Yates provided me with many seminal ideas about organizational memory and historical methods. Her historical work provided an important backdrop for this work, and her many rephrasings of my writings have found their way back into this thesis. Jack Rockart presented me with the clear voice of organizational requirements and the function of information technology in organizational processes. Everyone on the committee provided me with critical insights into the nature of organizations, system design, and memory. Most importantly, all provided friendly ears and constructive criticism. I hope that this thesis serves as a proof of their teaching ability; any fault is certainly not with them.

I also learned a great deal from my colleagues and friends at MIT. My friends at the Center for Coordination Science gave me their time and encouragement: Maya Bernstein, Geoff Bock, Kevin Crowston, Chris Fry, Kum-Yew Lai, Jintae Lee, Kazuo Okamura, Brian Pentland, Mike Plusch, Paul Resnick, and Ralph Swick provided sounding boards, critical suggestions, and day-to-day help. Brian Pentland and Elaine Yakura especially helped me in understanding social phenomena. Paul Resnick and Jintae Lee were my academic cohort and friends. To detail all the assistance from these people (not to mention the good times with them) would take paragraphs.

Others at MIT similarly helped. The crew at the Media Lab, Chris Schmandt, Debby Hindus, and Barry Arons, provided me with many programming breaks as well as generally great friends. The Speech Group was a good home away from home. Ralph Swick gave X answers and general assistance where the way was unclear. Pat Hardy and David Maw helped write the code for this project. Jolene Galegher and Mike Epstein provided late-night conversations and musings. The other students in Information Technology, Henry Kon among them, provided sounding boards. The staff of the X Consortium, principally Bob Scheifler, Donna Converse, Chris Peterson, and Stephen Gildea, provided me with access, information, and help.

Outside of MIT, Eric Mandel and John Roll at the Smithsonian Astrophysical Observatory plied me with bagels, forced me to re-think my design assumptions, and taught me more than one thing about software. Roger Brissenden and Steve Murray of the SAO and Joe Marks of Harvard and Digital Equipment Corporation provided access and encouragement.

Some of the ideas of Wendy Mackay, Ben Davis, Matt Hodges, Beth Anderson, Don Davis, Jonathan Grudin, John King, Rob Kling, andWin Treese have no doubt found their way into this thesis: the conversations with them, some carried on over many years,

have framed much of my work. Conversations with Harry Hersh, Bob Kraut, Tom Allen, Jan Walker, and Dick Marcus were critical to the development of this thesis.

Additionally, the hundreds of people at the various informal and formal field sites that gave me time, encouragement, and comments deserve my deep-felt thanks for their help.

Dennis Schmidt, Bill Hilliard, Kevin Osborn, Joyce Pollock, and Patty Goodson kept up the old TPS friendships even when I vanished for a while. Cynthia Hunt, Stann Chonofsky, Helen Bowns, Joel Finkelstein, Joni Ackerman, and Helen Ackerman provided the warmth of old friends and family.

But most of all, Rebecca and Erica kept me sane (or almost so). Becca grew up with this thesis work; it started about a year after she was born. I'm not sure whether I learned more from this thesis or from her, but in either case, I learned much from both. She's always been there with a smile, a new discovery, and a hug. Similarly, Erica was there when needed, but she also gave me the luxury of disappearing into my work when needed as well. To both of them, I remain grateful beyond words.

In writing this thesis, I have also been reminded constantly of my late grandparents and my father, who gave me these questions of history and memory, of social conformity and change, and of art and artifact. My grandparents spanned the last century and this; they saw men on the moon and refrigerators, television and telephones, airplanes and radio, automobiles and electric lights. My father sought out Bury's *History of Greece* as well as *Science News* for me when I was young. I think they would have viewed this thesis with the same sense of wonder they brought to their changing, but not necessarily progressing, world.

Introduction

....Memory is a house by the river,
A staircase spirals from attic
To water's lap. ....

-- from S. Schiff, "Gyrations of the Jade", 1978

...but the complication is the insidious result of the large number of aims I have in mind....

-- Fernand Braudel, *The Structures of Everyday Life.*

This thesis concerns itself with software systems that support organizational memory. These systems are among many important classes of software systems that can be considered socially oriented, in that they gain their effect and derive their meaning from the social world. For example, the proponents of the socio-technical systems approach (e.g., Mumford and Henshall 1979, Bostrom and Heinen 1977) argued that the design of Information Technology systems needed to consider organizational and workplace requirements. Likewise, researchers in the Computer-Supported Cooperative Work (CSCW) field have argued that CSCW systems must be based on our understanding of the social systems within an organization (e.g., Malone 1985, Suchman and Wynn 1984). To the extent that their technical design considers the social issues of the problem domain and attempts to incorporate social and organizational issues, IT and CSCW systems will reflect combinations of technical and social design considerations. Thus, one could argue that IT and CSCW systems especially need to be studied from both social science and technical perspectives.

Running throughout this entire dissertation, therefore, is the assumption that design for organizational memory systems cannot be based only on technical criteria, but must also include social criteria. I believe this thesis draws its interest from my attempts

to incorporate the social world in the design of these software systems, and accordingly, this thesis is concerned with a group of social considerations and premises.

In the first chapter, I explore organizational memory at the organizational level and survey the organizational literature pertaining to organizational memory. In the second chapter, I explore the same issues at the individual level, surveying the information seeking literature. In the third chapter, I discuss the technical literature, surveying several software paradigms of potential use to determine whether they serve the social premises outlined in the first two chapters.

In the fourth chapter, I introduce several new software applications, and in the fifth chapter, I briefly survey the underlying system for all of these applications. (Readers wanting more details on the underlying system, the Answer Garden Substrate, will find a detailed description in the appendices.)

Finally, in the sixth chapter I discuss the results of a field study of one of those applications, Answer Garden. The field study not only tested the application, it also examined the social premises behind the application. Accordingly, there are three sets of findings here: technical, social, and an intermediate group that I call social-technical findings.

The conclusions of this thesis are not only about a particular application; they are also about the social premises necessary (I believe) for building similar CSCW systems. In much the same way that the Artificial Intelligence community has used computer systems as a method of cognitive experimentation, we may find that CSCW systems, with their concomitant field studies, will lead to a greater understanding of underexplored areas of organizational life.

The study group

Throughout this thesis, I will use the software engineering world as my study group. I primarily examined R&D software engineers using the X Window System, a user interface system for workstation-class computers (Scheifler and Gettys 1987, Swick and Ackerman 1988). (There were a few exceptions, and I will note these as they occur in the thesis.) I did so for a variety of reasons, some theoretical, some opportunistic, and some personal.

First, this group is of interest for important theoretical reasons. Software engineering constantly requires a substantial amount of information seeking and knowledge acquisition because there is no body of commonly accepted knowledge (aside from low-level data abstractions and algorithms which all programmers are expected to know). Furthermore, the software world is undergoing constant revolution, preventing an individual engineer from acquiring lasting expertise. Software engineering is a fruitful place to examine the social issues of information seeking and sharing.

Second, I was already familiar with the software engineering world. It is often said in anthropological apocrypha that one sees best just before turning native. I went native in the software R&D world many years ago. It took a constant effort to see the software engineering world afresh. Fortunately, the inclusion of social science questions in my research made that world new. And, in many respects, my familiarity (even with the jargon) and my status in that world allowed me openings and conversations that others, with less software experience, might have had trouble obtaining.

Finally, I wanted to take this opportunity to reflect not only on the design of a technical artifact but also on a group of people with whom I have been associated for a long time. Software engineers are particularly interesting to me because I have been one. It is not a particularly easy occupation, nor a particularly well understood one. I estimate

the half-life of software knowledge to be about three years currently, and it is decreasing. There is a constant scramble to know the latest, profitable thing, and those who have not done so find themselves unemployed. I wanted to develop an artifact and a project that would help software engineers learn and acquire new skills, to reduce the scramble. And, I wanted to be able to say something interesting about the software engineering community when I was finished.

Chapter 1: Organizational Memory and the Organization of Expertise

1.1. Expertise, management, and information technology: an introduction

Expertise is everywhere in a company. Each person has his or her own areas of skill and knowledge. But few companies can effectively and efficiently manage this critical resource.

Firms could very well find that their critical success factors in the next years will include issues related to this resource. Stewart (1991) raises the possibility that the organizational issues of finding the right expert, growing knowledge and memory, and managing intellectual property may become increasingly important. Properly managing its intellectual resources might enable the organization to function more effectively and to prosper (Rockart and Short 1989). How an organization arranges, maintains, coordinates, and engineers its expertise appears to be little understood, and there may be room for considerable managerial improvement.

The technological environment now makes such improvement more likely. Perhaps the best way to describe this is by using Galbraith's (1973) classic model of organizational fit to a changing environment. He offers four alternatives for allowing an organization to grow by augmenting its ability to process information. His alternatives are creating slack resources, creating self-contained tasks (or units), investing in vertical information systems, and creating lateral relations. Extending Galbraith's model to combinations of augmentation methods, one might imagine an information technology that was not vertical but horizontal, and that served to increase lateral relations. Such an information technology could enable a firm to locate its latent sources of expertise. For

example, information technology could be of substantial assistance in matching experts with problems, re-organizing experts around problems, allowing experts who are partially outside the boundaries of the organization to participate in problem definition and solution, coordinating and integrating newly acquired experts, and enabling managers to grow expertise.

There is considerable evidence that such activity is already occurring through broadcast methods such as bulletin boards and electronic mail distribution lists (e.g., Feldman 1987, Finholt and Sproull 1990, Pickering and King 1992). There may be additional technological possibilities. For example, despite the possibilities of electronic mail, broadcast methods will often be less efficient than point-to-point communications. Systems that provide some of the benefits of information retrieval systems, broadcast systems, and point-to-point systems could be beneficial (Sproull and Kiesler 1991). Systems that enabled the information seeker to find standard responses might save an organization's scarce expert resources. Systems that sought the proper expert for an information seeker initially by point-to-point communications might allow the organization to coordinate information seeking and information providing activities more efficiently. This point-to-point communication could be augmented by point-to-distribution-list (narrow-band broadcasting) and bulletin boards (broadcasting) systems.

There are organizational considerations as well that argue that the time for these efforts has arrived. While increasing environmental complexity led to the adoption of Galbraith's intra-organizational accelerators, such as matrix organizations and vertical information systems, these adoptions, in turn, have led to a further increase in environmental complexity. In this game, all players in a marketplace must eventually adopt a new technology, lest any player create larger margins and finally a barrier to entry. As competition increases, and markets appear, disappear, and collide, environmental complexity increases. Additionally, large-scale historical forces, such as

urbanization, increasing internationalization, and the like, change the structure of markets, competing firms, and the general business environment. This prompts firms to respond internally as Galbraith described, leading to further pressures on markets and competing firms. Thus firms find themselves constantly looking for new ways to augment their ability to process information.

Cohen, March, and Olsen (1972) suggest another reason why these types of information systems are necessary today. Simply put, the likelihood that a perceived problem will find an organizational solution increases as the organization has additional resources to find previously created solutions or to newly create the solutions. Stewart, in his popular article, also saw this necessity:

> In summary, every company depends increasingly on knowledge -- patents, processes, management skills, technologies, information about customers and suppliers, and old-fashioned experience. Added together, this knowledge is intellectual capital. ...In other words, it's the sum of everything everybody in your company knows that gives you a competitive edge in the marketplace.
>
> Such collective knowledge is hard to identify and harder still to deploy effectively. But once you find it and exploit it, you win. (Stewart 1991, p. 44)

The remainder of this chapter examines the organizational issues involved in managing such expertise and knowledge. I will discuss these issues at the macro level, examining how expertise and knowledge could be captured within the firm. I will then, in the next chapter, discuss these issues at the micro-level, examining how organizational members might view them.

1.2. Organizational memory and organizational knowledge

From the macro perspective of the organization, one method of managing its intellectual resources is to attempt to augment its organizational memory. In the remainder of this chapter, I will review the small, and somewhat diffuse, literature on

organizational memory, followed with a discussion of how organizational memory might be better arranged and accessed.

Certainly organizations can be said to have some form of memory. Indeed, an organization must retain knowledge of its past efforts and environmental conditions. For example, if an organization learns, then the result of that learning should be available later (Levitt and March 1988, Duncan and Weiss 1979). Walsh and Ungson (1991), in their review article on organizational memory, make the following argument: "To the extent that organizations exhibit characteristics of information processing, they should incorporate some sort of memory, although not necessarily resembling human memory" (p. 57).

However, the exact meaning of "organizational memory" is hazy in the literature. I believe that the term "organizational memory" is being used in the literature in four different but closely related ways, which I will discuss in turn below.

First, it is used as a metaphor, anthropomorphizing an organization to note similarities with and differences from human memory. In several ways, the term "organizational memory" gets its effect from this anthropomorphization; the metaphor is key to an intuitive understanding. Writers, including Walsh and Ungson, have noted the problems in any memory such as corruption and decay over time. Here the metaphor points directly to an awareness of one dimension of organizational memory, the potential decay of valuable information over time. The anthropomorphization also suggests the immediacy of memory. In the best situation, people do not need to recall memory consciously. It is immediately present.

The anthropomorphic metaphor has an important role, I believe, for an understanding of a living system such as an organization. From the analogy, one can easily see the potential advantage in learning from past experience and the blindness that

may result from following that past experience slavishly. It suggests several key dimensions of organizational memory. Nonetheless, the analysis of organizational memory as limited to the anthropomorphic is clearly a substitute for considering the actual practices of an organization. The practices of an entire organization of people cannot be completely analogous to that of a single individual.[1]

The second use of the term "organizational memory" is to refer to the permanent, archival past of the organization (Bearman 1987; Chalmers 1989; Walch 1990). This use situates the memory as something in the dim past, and separates the organizational past from the current day-to-day activities. While this view provides a meaningful context for records and archival management, it is limited by its implied disconnection from the present. It may be usefully seen as a special case of the fourth meaning below.

The third, related use of the term "organizational memory" refers to the written record (whether on paper or in electronic form). In this meaning, organizational memory forms an active part of the organization, but it is written, intended to document some features of the organization. Yates (1989) uses "organizational memory" in this manner to discuss, for example, the railroad manuals of the nineteenth century, and Conklin (1992) uses it similarly when describing design rationale systems. Again, this will be seen to be a special case of the next meaning.

---

[1]Furthermore, this anthropomorphic metaphor can be dangerous. It can slide into an idealization of organizational memory, namely something similar to an individual's memory but without the imperfections. Such idealizations, often proposed by system builders (and perhaps even by this one), miss many of the salient constraints and limitations of organizational memory. Some of these constraints and limitations, such as imperfect recall of context and the placement of organizational memory in an organization with limited resources, will be discussed below.

The fourth meaning of the term "organizational memory" is to consider it as organizational knowledge with persistence.[2] What does it mean for knowledge to persist, especially knowledge within an organizational setting? If organizational knowledge is *now*, organizational memory is just before *now*. What then is the memory? Historiography has considered this question in detail (e.g., Collingwood 1946, Carr 1961, Braudel 1980, Burke 1991). Organizational memory, like any remnant of a past, faces three standard historiographical issues: temporal distance, point of reference, and context. However, the placement of these historiographical issues within an organizational setting differentiates organizational memory from professional history.

Viewed in this way, organizational memory may be arrayed along a dimension of time, with one endpoint being the second meaning of organizational memory (the archival) and the other endpoint being the knowledge of just shortly before *now*. That is, an organizational memory system may store organizational knowledge for any time period.

---

[2]This organizational knowledge can cover a range of "commonly-held cognitions" according to Sackmann (1992). She differentiates dictionary knowledge ("what"), directory knowledge ("how"), recipe knowledge ("shoulds and recommendations"), and axiomatic knowledge ("why"). In my definition, there is no clear distinction between organizational knowledge and its resulting memory except for the retentive nature of memory through time.

In computer science, "persistence" refers to maintenance of data, often objects, between sessions or uses. In this context, "persistence" implies that the knowledge does not go away, but remains somewhere, waiting to be used again. This meaning is not the same as in information science which uses a meaning similar to that in the dictionary. The meaning in information science is synonymous with perseverance, that is, existing despite active opposition. No agency or active effort is necessarily implied in the computer science meaning of "persistence."

Frame of reference is the second historiographical issue. To a historian, history is selective; the human observer selects and filters the data according to his needs and current understandings. Organizational memory, like any historical record, is fragmentary, but provides access to an understanding of the historical causes behind common assumptions, beliefs, and processes. To garner access to this understanding, professional historians view their histories from the inhabitants' points of view, attempting to be truthful to the inhabitants in the selectivity of materials.[3] To the organizational participant, however, memory is more fruitfully viewed from the current needs and assumptions of the organization. If only a small amount of time has elapsed between the event and the present, there is little difference between the two. But, if a sufficient time has elapsed, it will be difficult to reconstruct the meaning of the event in the terms of the event's participants, and organizational members will be more likely to view it from their own vantage point and interests. Organizational memory, then, belongs to the current organizational members, and the effort to interpret it must be exerted by the current organizational members.[4]

The third historiographical issue is one of context, and is closely related to frame of reference. The implicit understanding of the stored knowledge is based on a similarity

---

[3]Most historians now acknowledge the impossibility of removing their points of view from their work. However, the examination of the past from the inhabitants' point of view (that is, as they would have seen it) distinguishes professional history (Carr 1961).

[4]As Graham (1992) among many others point out, there is a reconstruction and reinterpretation of the past for the elucidation of the present by both historians and organizational members. Organizational members, then, serve as "lay" historians to borrow Callon's (1987) term. The difference would be in the additional training a historian has in a specific form of observation, just as the difference between Callon's engineer and a sociologist lies in additional training and sensitivity to sociological phenomena. Nonetheless, because of the "close-in" quality of the organizational members' interests, the distinction between organizational memory and history is important analytically.

between the current context and the context of the organizational inhabitants at the time of storage. This understanding varies from immediately shared to reconstructed or foreign. However, the organizational memory exists within an entity with limited resources, as Levitt and March note (1988). As such, an organization is likely to be little interested in achieving total recall of its experiences; a "good enough" recollection will be sufficient. Therefore, recall that satisfies an immediate problem is all that will probably be required, and the archival past will be less interesting than the immediate past. The effort that is required to construct a context will be larger, and therefore less likely, than the effort to retrieve an immediately understood context for the organizational knowledge.[5]

The differences, then, between organizational memory and an academic history are in organizational memory's location. Organizational memory is situated in an entity that is concerned with achieving its goals in a manner that minimizes the drain on its limited resources. This is a critical shift from the needs of professional history. It argues that an organizational memory that is immediately tied to the on-going processes and considerations of an organization will be more important and useful to an organization. An organization's interest is much more in the immediate than that of a professional history. Therefore, recall that satisfies an immediate problem is all that will probably be required, and the archival past will be less interesting than the immediate past. Furthermore, it suggests that systems that facilitate that type of immediacy of effect in augmenting organizational memory will be more useful.

Organizational memory, then, concerns itself with usually recent events and outcomes within an organizational context and for organizational purposes.

---

[5]The issue of context has recently begun to attract attention in the computer-human interaction literature. Examples include Berlin et al. 1993 and Dourish et al. 1993.

Organizational memory is subject to the problems of storage, recall, and interpretation by organizational agents.

Having moved away from any anthropomorphizing, we can consider where organizational memory persists and what its functions might be. Walsh and Ungson have a useful model of location:

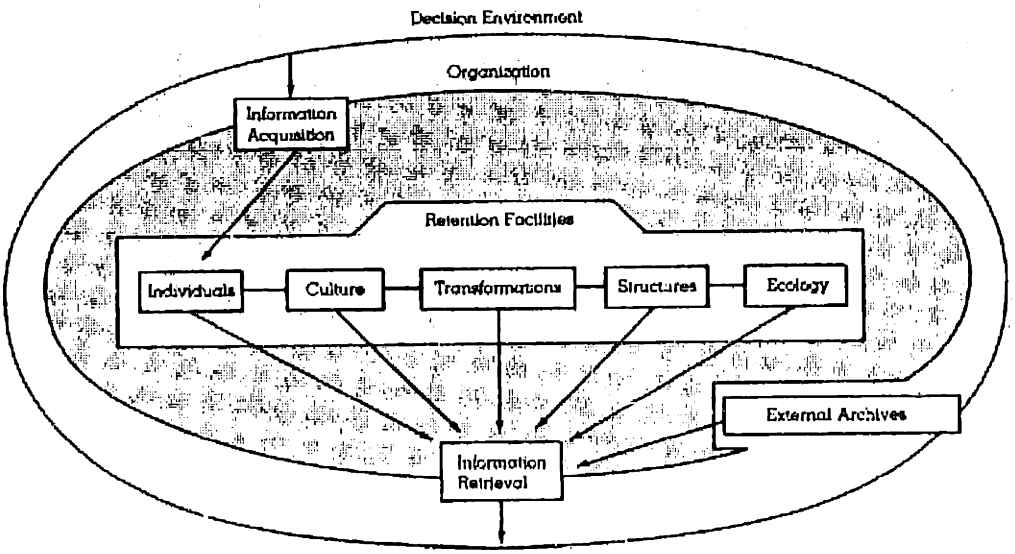

Figure 1.1: "The Structure of Organizational Memory", Figure 1 from Walsh and Ungson

In the Walsh and Ungson model (Figure 1.1), organizational memory can be retained in six places: individuals, organizational culture, organizational transformations, organizational structures, organizational ecology, and external archives. (External archives are data external to the organization, not its own archives.) This list, however, should be expanded to include internal information repositories such as corporate manuals, databases, filing systems, and even stories (see Yates 1988, 1990, 1989; Morgan and Root 1979, Orr 1986). The third meaning of organizational memory, as captured in a written record, can now be seen to be only one form of organizational memory.

One should note that the combination of information repositories and organizational members form a core of easily retrievable organizational memory (Morgan and Root 1979, Graham 1992). These are the most amenable to augmentation through information systems.

More problematically, Walsh and Ungson largely ignore where organizational memory can be used since they are primarily concerned with managerial decisions. However, organizational memory serves a role in a larger set of organizational processes. (Managerial decisions are, after all, a particular kind of organizational process.)

While organizational processes have yet to be fully defined and described (Curtis, Kellner, and Over 1992, Malone et al. 1992), some possibilities for organizational memory can be ascertained by viewing organizational processes in terms of the computer science concept of non-deterministic Turing machines. A non-deterministic Turing machine has three components: an input, an output, and some non-deterministic algorithm. If one considers an organization as a set of these processes, then some important possible uses for organizational memory leap out:

1. To document the output of a process. An example might be a design specification.

2. To serve as input to a process. In software implementation, examples might include design criteria or answers about component features.

3. To describe the algorithm of a process. Examples might include design rationales or assembly-line manuals.

Several additional roles for organizational memory result from the need to arrange these processes together. Accordingly, organizational memory is needed:

4. To maintain state between subprocesses or within a temporally separated process. A simple example is writing down someone's telephone number for later use.

5. To maintain the definitions of goal selection and decomposition in a process (Malone and Crowston 1992). This is required in complex processes when breaking the process into subprocesses.

6. To maintain inter-process coordination.

Walsh and Ungson's transformations and structures (which they take to be synonymous with organizational roles) provide function #3 above, a description of the process. One could consider their view of culture as providing some of the meta-definitions of a process (#5 and #6 above). One might even argue that individuals, along with archival records such as design rationale systems, are the chronicles of process outputs.

Missing, however, from the Walsh and Ungson model is the ability to use the past efforts of an organization to provide inputs to further activities. Examples of such inputs might include solutions to technical problems or analyses of business competition. One would like to reuse such information in order to avoid duplication of effort, but more importantly, retrieval of such inputs, especially in non-analyzable processes such as design processes, can often take far longer than might be wished.

It is worth noting this use of organizational memory as a possibility. Augmenting the ability to use organizational memory as a process input is amenable to computer support, since it involves finding the right "pieces" of organizational memory. However, this task presents interesting challenges. The "pieces" will often be unpredictable within a

given organizational process, and they may require a shift of context from their previous use.[6]

Graham has written, "We are beginning to feel the need, I think, to find a substitute form of organizational memory" (p. 8). While opinions differ radically on what that form might be, many authors have made the same call - from Morgan and Root (1979) through Huber (1991, 1990).

I next turn towards a consideration of what it might mean to find the right "pieces" of organizational memory.

1.3. The organization of expertise and expertise networks

If one wishes to retrieve the necessary "pieces" of organizational memory for further use in some organizational process, then the issues of locating those "pieces" move to the forefront. Since the most likely sources for this information are

---

[6]Such retrieval may require transformation of the information in order to be a valid input to the new process. If it does, then it remains open whether this constitutes organizational memory or organizational knowledge. The semantic problem of differentiation in this case is similar to that of whether a historical fact should be considered as part of the historical period per se or as part of the current period. Standard historiographical thought is that no historical fact can stand independently of its interpretation, and the distinction is only analytical. This retrieval, especially if transformations are required, occupies a nexus between organizational memory and organizational knowledge.

With this in mind, I believe that Graham's (1992) distinction between organizational memory as object and organizational memory as practice is both problematic and intriguing at the same time. The distinction is merely analytical in that organizational memory is always re-interpreted upon retrieval. However, she makes an important point in that the re-interpretation process can be varied and managed in itself. Her work offers an interesting consideration, that of supporting the reinterpretation and sense-making of the memory.

organizational members and organizational repositories, questions of topological connections among organizational agents must be examined.

To view organizational knowledge and organizational memory in terms of access to is to consider *organizational expertise.*[7] Examining organizational expertise assumes that organizational knowledge is embedded in the organization's agents, whether human or computer.[8] Of particular interest here are the people in the organization. Each person

---

[7]Expertise, then, I define to be knowledge as embedded in an organizational agent. I define organizational expertise to be knowledge embedded in organizational agents. Hence, it is distributed expertise. This definition is consistent with the dictionary definition, although the definition used in this thesis places more emphasis on the possibilities for a range of knowledge from novice to expert (Draper 1985, Doane, Pellegrino, and Klatzky 1990). Acknowledging a range of expertise opens the examination up to include all organizational members, not just those defined as experts. I am using "expertise" here largely because of the connotation to "knowledge" that has it abstracted from the individual; "expertise" more fully connotes the organizational members involved. I will return to this definition in the next chapter.

Expertise, organizational knowledge, and organizational memory have both an objective and a subjective side to them. For example, organizational remembering may be socially constructed; nonetheless, there can be an organizational memory of retained objects, information, routines, and the like. My next chapter will be more concerned with subjective elements; this chapter has been focused more on the objective. In practice, however, a distinction between the objective and subjective nature of organizational memory, knowledge, and expertise is only analytical. While tracing the objective and subjective sides of organizational knowledge and memory without clearly demarcating the two, I am following Pentland (1991, 1992). Pentland argues that the objective and subjective cannot be clearly separated in the use and construction of organizational knowledge, although it is useful to do so for analytical purposes.

[8]For the purposes of this argument, organizational expertise can include computational agents. It would be difficult to argue that computer agents can interact in a manner that enables them to "practice" their expertise or knowledge within the situated activities of the organization (Pentland 1992) in the same manner as human agents. For some purposes, computational agents might be better considered as structure or tools, rather than fully formed agents. For other purposes, computational agents display a limited form of expertise or agency. For this argument, the consideration of the information resources alongside that of the human expertise in a firm provides greater insights than the consideration of either alone.

has his own capabilities, both in subject matters and in skill levels. As such, the organizational access problem becomes one of finding the right agent, probably human but possibly computer, through the communications network of the organization. Organizational expertise, then, is a useful organizational metaphor, one that uses distributed artificial intelligence (e.g., Cammarata, McArthur, and Steeb 1988, Lesser and Erman 1988), or how intelligent computing resources can be distributed through time and space, as a metaphor for organizational life.

My argument is that organizational expertise is affected not only by the knowledge domains and skill levels of the organization's agents, but also by the social networks within the organization. As a useful abstraction, consider an agent's expertise as a multi-dimensional cluster of skills. Then the firm's expertise is a multi-dimensional network made of these agents. This *expertise network* is the firm's manner of organizing its expertise.[9] In "absolute" terms, the complete expertise network is the sum total of all possible skills in the firm. It is what the firm could do (presuming some connection between expertise and action), if it was able to utilize completely what all its members knew. The "actual" expertise network is dependent on the communications network within the organization. Indeed, it is dependent on communications *sub*networks, ones revolving around particular subjects of interest, expertise, and social contacts.[10] Even

---

[9]The expertise network appears to be partially equivalent to Pentland's (1992) "competence structure," although the elements of his competence structure are not fully delinated. I used the term "expertise network" originally in Ackerman 1992 because the participants, as will be noted in the next chapter, used "expertise" as a social category. (The term "expertise structure" was, I thought, too similar to computer science technical terms.) Additionally, "expertise network" conveyed the sense of uneven mapping of knowledge, memory, and expertise among the organizational agents.

[10]Thus, an expertise network can be considered as the sum of all of the agents' expertise for a given domain.

though two people might have sufficient knowledge between them to solve a problem, if they cannot talk -- because they don't know each other, they have no communications network between them, or they can't stand to be in the same room with each other -- the organization cannot be said to have the capacity to solve the problem. Usable organizational knowledge or organizational memory is, then, bound by the social networks that may or may not exist within the organization.

The expertise network is an indication of the firm's *organization of expertise*, how a firm arranges its personnel and information resources in such a way as to maximize such critical success factors as efficiency, innovation, or expertise sharing (Rockart and Short 1989). It may be argued that altering the organization of expertise within the firm, through different types of management or through technological systems, may alter the firm's abilities and ultimately its performance (Galbraith 1973).

So far, we have considered the organization and the environment at a specific time. An organization sees only what is interesting to it; some of the complete expertise network will be invisible or uninteresting to that organization. Thus, the complete expertise network structure carries its own seeds for transformation and its limitations to that transformation. It is able to move only where it has expertise, and it gains expertise only where it considers it valuable. Furthermore, the complete expertise network, and the organization as a whole, is in a dialectic with the historical processes moving around it in time; it can seldom remain static for long. These processes mold and shape the firm's environment, changing the requirements and pressures on the organization.

The expertise network, as mentioned, includes the latent ability of the organization to move in new directions. In other words, it provides the basis for allowing the organization slack in order to match newly determined problems with its latent solutions. Clearly, a large part of the latent and actual capabilities of an organization exists in its agents' knowledge and abilities.

The firm's ability to properly form and react to changes in the future environment is one of its most important resources. This management of organizational expertise over time (or knowledge as embedded in the firm) is sadly missing from the human resources literature. e standard human resources textbooks (e.g., Crane 1979; Dessler 1984; Cruth, Noe, and Mondy 1988) barely mention this organizational issue.

Considerable research is required on the issues of managing organizational expertise. It is not clear how much or how well firms' expertise networks or complete expertise network are managed. We do not know as much as we need to about how the problems of an organization are assigned to individuals for examination and action, how the experts are activated (including those who are marginal to the locus of activation or partially outside the organization), and how attention is focused on a required expertise (or expertise thought to be required).

1.4. Conclusions and summary

I have argued in this chapter that organizational memory is the persistent repository of organizational knowledge. This definition raises several important considerations in augmenting organizational memory:

- Organizational memory can be held in many places within an organization, but the most accessible locations are organizational members and information repositories such as manuals, databases, filing systems, and even stories.

  Thus, information technology can support organizational memory in two ways, either by making recorded knowledge retrievable or by making individuals with knowledge accessible. To augment organizational memory in this manner, an information system

would need to incorporate elements of information databases and communication systems.

- Augmentation might occur to facilitate the use of organizational memory as an input, output, or description of an organizational process. In addition, organizational memory could be used as a coordination mechanism. Since the organization has limited resources and interests, I have argued that augmenting organizational memory should be centered around a current organizational activity to be useful.

And finally,

- The expertise networks within an organization form the nucleus of the firm's ability to act. Since an expertise network is a combination of the social network and the expertise of the individuals in that network, an information system that wishes to augment the expertise network, and thus the organizational memory, in this manner will need to consider the social networks within the organization.

In the following chapter, I will examine this last point.

Chapter 2: The Search for Information in an Organizational Context

Viewed from the perspective of the organizational member, rather than the organization as a whole, organizational memory is invisible or at least muted and hazy. At a micro-level, the issue is one of information seeking by some organizational member. As such, information seeking can be considered as the process of finding the right "piece" of organizational memory. The goal, then, is to decrease the effort involved in information seeking in cases where the required information is not known to the individual involved, or to eliminate the information seeking in cases where the organization has redundant efforts. In this chapter, I will consider the social side of information seeking, that of finding the right organizational member of whom to ask one's question.

2.1. Information seeking

Information seeking has been extensively studied. Reviews of the information science literature concerning information seeking and information needs point to several hundred studies every few years. (See Allen 1969, Crawford 1978, Dervin and Nilan 1986, and Hewins 1990 for partial reviews.) I will not attempt to survey the large number of information retrieval system studies and of general population studies (e.g., Chen 1982), instead focusing on those studies of engineers' information seeking behavior. (Examples of these studies of engineers include Pinelli, Kennedy, and Barclay 1990, Rosenberg 1967, and Rosenbloom and Wolek 1970. A good review can be found in Pinelli 1991.) These studies are among the most detailed examination of information seeking in organizational settings.

The major work in the area is Allen's seminal study (Allen 1977). Among his many findings, Allen noted that an engineer's major source of information was direct contact and communication with colleagues. His finding of the impact on performance of gatekeepers, engineers who maintained contacts outside the group, is well known. Much of an organization's communication flow (for R&D engineers) is channeled through these gatekeepers, who track both the literature and organizational members who know about various subjects. Additionally, he found that engineers seldom used the formal literature, although the informal literature (trade magazines and the like) was used extensively. For these two reasons, information retrieval systems, which usually facilitate dissemination of the formal literature, held little promise in Allen's opinion.

Gerstberger and Allen (1968) noted some of the reasons why engineers would *not* go to colleagues, and instead use other information channels. This study is critical to understanding the organizational context for information seeking. It provides a basis for how organizational members view the information search process and their attitudes toward it.

Gerstberger and Allen studied 19 engineers and 154 information searches over 15 weeks. Figure 2.1 shows the discrepancy between expected usage and first-source usage. They found that these engineers did not go to the most frequently used sources first. Instead Gerstberger and Allen found that engineers go, in order of frequency as first source, to the technical literature, group members, experimentation, technical staff, vendors, company research, customers, people in other divisions, and external sources. (Their channels were large-grained.) This was unexpected since engineers, in general, go to colleagues and other people more than they turn to the literature or experimentation. However, engineers appear to turn to the literature, their immediate colleagues, and experimentation before consulting technical experts outside their immediate group.

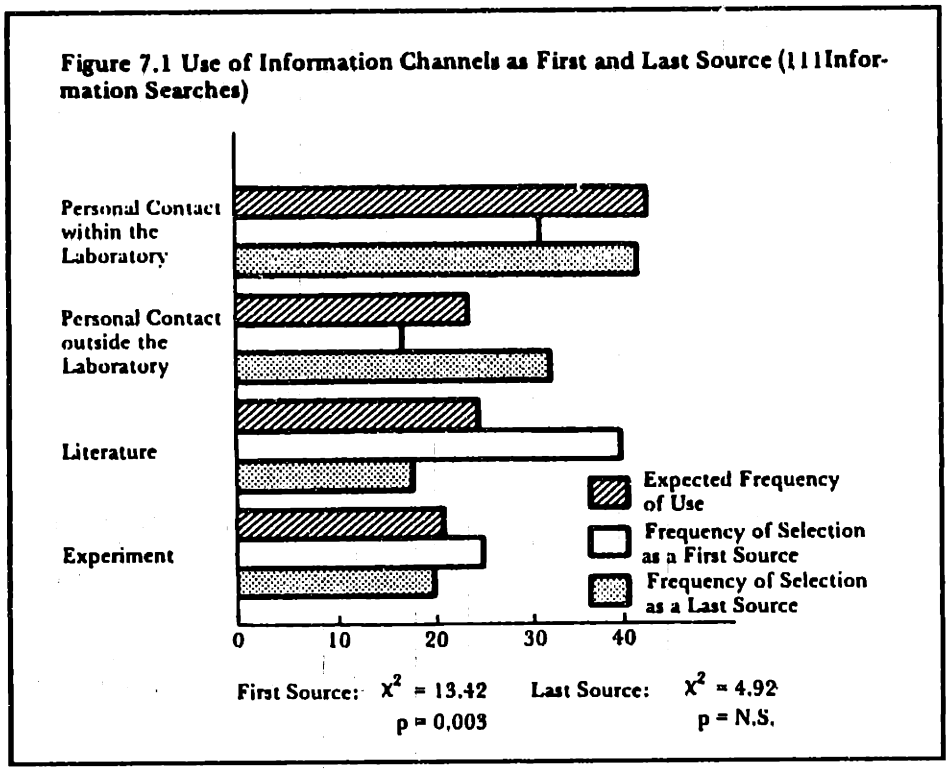

Figure 2.1: Use of Information Channels, from Allen, 1977

Moreover, they found that the engineers chose not to go the channel of the highest quality for technical information, but rather to go to the channel of highest accessibility. As Gerstberger and Allen reported:

> Apparently, in the minds of the Ss [subjects], there is some relation between their perceptions of technical quality and channel accessibility, but it is the accessibility component which almost exclusively determines frequency of use. (pp. 275-276)

Gerstberger and Allen go on to suggest that R&D engineers do not maximize gain, but rather minimize cost (or loss). Furthermore, engineers do only a local analysis; that is, they do not weigh future costs, such as searching additional channels upon failure in their information search, in making their initial analysis. (Hence the discrepancy between expected usage rates and first-source usage rates.) Gerstberger and Allen continue:

> This is an interesting phenomenon in which technical quality is recognized, but its consideration is delayed until after the channel has been selected. ...It is no wonder then that the perceived

> reliability of the source is taken into account, and that ideas are accepted or rejected, at least in part, on that basis.
>
> From a simple engineering viewpoint, this seems to be an appallingly inefficient way of doing things: using the wrong criterion early, and later correcting for it. (p. 279)

But, as they point out, just improving the quality of the channel's information is not sufficient. The channel must be used, and therefore must be considered to be high in accessibility or low in personal cost. That is, engineers do not view quality as their primary motivator; instead, they examine channels for the lowest psychological cost.

Allen furthers the analysis in his book (Allen 1977)[1] Allen argued that the psychological cost was in the status implications of admitting ignorance and of lack of reciprocity. As he states, "The costs in the consulting transaction are often perceived to be far greater than the benefits" (Allen 1977, p. 191). Allen describes these costs quite well:

> ...an engineer who very frequently seeks consultation will soon wear out his welcome, unless, of course, he has something substantive to offer in return. ...Unless a person is able to give information in return, the tendency is to cut him off before very long.
>
> ...the information seeker also sees the transaction as a costly one. This is particularly true when, as in the case of strangers, there is a very remote possibility in the information seeker's mind that the potential consultant will respond in an ego-threatening way. ...This results from the fact that a person's reputation in an organization can be severely damaged if he is too often seen in the role of an information seeker.
>
> An engineer's prestige among his colleagues is founded to a great degree upon an almost mystical characteristic called "technical competence." To admit a lack of technical competence, especially in an area central to the engineer's technological specialty, is to pay a terrible price in terms of lost prestige. Furthermore, the problem is exacerbated when the consultant

---

[1]The section in Allen's book appears to use Gerstberger and Allen's data and findings although it is not cited. The statistics reported in both are the same.

> resides in the same organization as the individual seeking help. (Allen 1977, pp. 192-193)

For the information source, the ego rewards drop with continued contact while the costs continue. For the information seeker, the ego costs are immediate and can only continue. Allen quotes one engineer as saying:

> When you come into a place as a junior engineer, you ask anybody anything and they accept the fact that you are real dumb. After you have stayed awhile, you ask fewer people fewer things. There are certain things that are expected of you and you are a little bit querulous [sic] about displaying ignorance. (Allen 1977, pp. 194-195)

Allen also reported several strategies for reducing the psychological cost to the individual: admitting ignorance (although this only works for a short while), preparing for information seeking by reviewing the literature, and asking people known socially (and therefore asking in the mode of friendship). Organizationally, Allen suggested alleviating psychological cost by finding some way to maximize benefits for both parties in the information seeking process.

2.2. Information seeking among software engineers: an exploratory study

In order to determine whether Allen's findings about information seeking extended to software engineers and what features these implied for an organizational memory system, I interviewed 31 software engineers, system administrators, and support engineers who were taking training courses in the X Window System, a user interface system for workstation-class computers (Scheifler and Gettys 1987, Swick and Ackerman 1988).[2] I found, like Allen, that information seeking occurs in a highly charged social

---

[2]There are some important differences between software engineers, especially those engaged in user interface work, and engineers in more traditional engineering disciplines. For example, user interface (UI)

environment. However, there were some interesting nuances in the data that add to Allen's findings.

2.2.1. *Status interactions and social norms*

The important status interaction was further explicated in the interviews. Information seekers, like most people, prefer going to people who will be friendly -- either because they are trusted or just because they are personable individuals. They are acutely aware of the status trade-offs involved in such an information-seeking session. As Allen found, asking questions about a new area is not unlike asking questions upon entry into the organization (Miller and Jablin 1991); it is felt to be dangerous to appear

---

engineers engage in more minute and iterative problem-solving than do aero-space engineers planning a new aircraft.

All but one of the 31 interviewed engineers were in the process of using the X Window System. They had already used X at least at a user level (and often more extensively) for at least two weeks before being interviewed. I interviewed engineers using X in order to fit with the field study described in chapter 6. The interviewees' organizations ranged from large aero-space companies to high-technology startups to universities.

All participants volunteered after my requests for interview subjects during training classes and tutorials on X. Since the engineers felt it necessary to have training, this selection emphasized people who were having some level of difficulty in obtaining information about X. However, this emphasis was discussed in the interviews and weighed in the presentation of the interview material here. In fact, these participants were all the more interesting because they highlighted some interesting dysfunctionalities in seeking and obtaining information that are not uncommon in organizations.

The interviews lasted between one-half hour and three hours. They were open-ended, but focused on what it was like to seek out information about X in their organization. To my surprise, these engineers were generally quite voluble about their problems. The interview data was then categorized, looking for patterns of information dysfunctionalities in seeking information about X (Filstead 1970). Later analysis focused on the type of knowledge being mentioned.

too "stupid." I was told of one engineer who was a company expert on their operating system and who refused to ask questions about X, telling the support engineer that he would ask them questions when he knew something.

Asking questions about an area that you are supposed to know can be more fraught with subjective danger. Part of it is the question of status, but another is going against the social norm of not bothering others without trying the documentation and other technical literature first.[3] As one support engineer put it: "I had to convince them [the engineers in charge of a product] that I tried as hard as I could to get the information on my own."

It is not easy to violate this norm; however, sometimes it is required. For example, there may not be anyone else able to answer a question.

2.2.2. *Social definition and the enactment of expertise*

When seeking information from an organizational member, one must seek it from someone who knows that information. Software engineers, like any engineer, seek this information from people with expertise in the area. Information seekers are quite aware of the skill levels of potential sources. They do not view the interaction as a novice-expert interaction where any expert can be substituted for any other expert. Instead, they

---

[3]Hauptman (1986) noted that a difference between software engineers and other R&D engineers was that software engineers were much more reluctant to ask questions of others and, in turn, to answer questions. On the other hand, software managers bore the brunt of question-asking. No interviewees reported going to a manager, perhaps because I was asking about information seeking about a particular technical system. One interviewee did say, however, that "Managers have an incentive and [the] patience to answer dumb questions." Software engineers may not feel a need for checking the technical literature and documentation and for reciprocity when asking questions of software managers. This could also be true for some support desk personnel.

are aware that various individuals have different expertise, and a particular range of expertise at that.

This is weighted with several considerations. One is that the definition of "expertise" and being an "expert" are multi-faceted.

Neither term is well defined in the research literature. While it has been widely bandied about, even in the management and computer science literatures, the nature of expertise is not well defined in those literatures. "Expertise" is defined by the dictionary (Webster's 1986) as "expertness in a particular field: know-how," while "expert" is defined as "having, involving, or displaying special skill or knowledge derived from training or experience." Adelson (1984) noted distinct differences in the ways that software experts and novices approach problems. She notes that novices are much more detail-oriented, whereas experts have developed mental models of the problem space. Experts use these mental models to guide their use and memory of detail. Guidon (1990) follows the artificial intelligence and cognitive science literatures in defining an expert as one who has a better and more efficiently executed data-driven rule base and knowledge schemas. March (1990) argues that the heart of expertise may result from being able to adapt rules over time and in various levels of appropriateness.

The interviewed engineers used "expertise" and "expert" in various, somewhat conflicting ways.[4] There was an objective sense to the terms. One particular group of people, all within one organization, pointed to a particular individual as clearly understanding a technical product. He had a clear conception of this complex product,

---

[4]While the literature displays a tension between the objective nature of expertise (e.g., Adelson 1984) with the socially constructed use of expertise (e.g., Cicourel 1990), it should be noted that the interviewees moved between these in a manner reminiscent of Bourdieu's description of "pre-logical logic" (Bourdieu 1990).

and was able to relate the details of the system to his general model in the manner that Adelson found.

There was also a relative sense to the term "expert." As one engineer said: "There isn't any formal expert [about X in his company]. People get reputations on knowing something and become the de facto expert. They may not know much more than the others." Others would joke about being the expert in their companies since they had taken a training course. To some extent, they mocked themselves or other relative experts, not considering themselves true experts. It should be noted that the interviewees would often refuse the title of "expert" but acknowledge expertise in some subject.

The final sense of the terms was a subjective sense. As mentioned above, R&D software engineers have an implicit understanding of expertise and are able to identify individual engineers as having expertise. Some engineers were acknowledged as being experts. This "everyday" knowledge is well recognized in the qualitative literature (Filstead 1970; Schwartz and Jacobs 1979; Van Maanen, Dabbs, and Faulkner 1982; Becker 1986).

The engineers recognize expertise by enacting it, to borrow Weick's (1979) terminology, through the double interacts of information seeker and information source. This meaning, partially historical and partially situational, enables both parties to suggest, defend, and develop expertise.

Expertise, for software engineers, is grounded in the particular. A person has expertise in a specific topic (or set of topics). Expertise is subjectively determined, or enacted by both expert and information seeker. This can be seen most easily in the "one-upmanship" conversation among R&D software engineers that are similar to those described by Anderson (1978) in his urban ethnography of a South-Side Chicago bar and liquor store. Anderson reported that participants made assertions of self and that other

participants tried to attack those assertions. One person might claim to have fought in the paratroopers (a high-status role), and the others might sharply question that assertion.

R&D software engineers engage in similar banter. One software engineer might make a technical assertion, and other engineers might follow with factual or design challenges. Status, and thus recognition as an expert (or relative expert), is garnered by the initial participant's ability to counter those challenges. The discussion can become quite technical and elaborate, and further status is enabled through sophisticated claims and counterclaims. Topics, in fact, can shift over several hours, as participants essentially argue over their status positions.

However, unlike the bar-room participants in Anderson's study, software engineers do not appear to engage in personal attacks as frequently. Engineers appear to value technical discussions in different ways than general discussions. While Anderson's subjects denigrated a participant's opinions in general because they felt the speaker "ignorant," the observed engineers use more finely grained distinctions. An engineer may not offer substantial opinions about sports or politics, but might still be valued for her technical abilities. This suggests that, for engineers, a person is allowed to be an expert by the willing agreement of her colleagues and according to her actions.

Furthermore, one can also see this type of collusion in dyadic interactions. For example, one expert (not on X, but on Emacs, a sophisticated text and programming editor) described a substantial repertoire of activities to help information seekers. Most maintained her expert status, often without forcing her to defend it. One such activity in her repertoire was to tell the information seeker that she was too busy to research the question, but that the answer was in a general location in the manual set. Other participants reported doing more or less based on their assessment of the information seeker's expertise.

As Allen pointed out, information seeking is not a straight-forward information transfer. For people with a career anchor in technical expertise (Schein 1978), each information seeking interaction is a double-edged sword. It provides the possibility for consorting with experts (or relative experts) and thus gaining prestige and status. On the other side, it also provides the possibility of being seen as incompetent, a possibility either so frightening or so likely that most engineers prefer using the documentation, other technical literature, or friends. This is only reinforced by the social norm of examining the documentation first and not bothering the expert; it is even more important to have non-trivial questions to ask the experts.

Any system attempting to augment information seeking (and use the organizational members as part of an organizational memory system) will need to consider this status interaction. The norm of not bothering experts, if not enforced by the organization, can be interdicted; the status interaction will likely not so easily disappear.

2.2.3. *Dysfunctionalities of information seeking*

Interviewees mentioned several dysfunctional situations for the information seeking activity.[5] The expert could be unavailable. This condition exists in startups, small companies, and organizations that have concerns for military security. They also exist in parts of very large organizations where the parts do not communicate or where the culture frowns upon sharing across organizational boundaries. User interface work is a clear example of this since user interface software engineers, usually scattered within product groups, often do not know of others in the organization attempting to solve

---

[5] I follow the use of the term "dysfunctional" in Blau 1955 and Blau and Meyer 1987. In Blau and Meyer, "dysfunctional" is defined as "some feature of a social system [that] is detracting from adaptation and adjustment - the capacity of the system to serve those functions it is intended to perform" (p. 139). The definition follows Merton's use of the term.

similar problems. In fact, this shows that an information-seeking dysfunctionality may not require that an expert *per se* be unavailable. User interface engineers may suffer from the not even being able to ask fellow user interface colleagues.

The engineers often expressed frustration at the difficulty of seeking information when they felt isolated.

> "Because I'm so isolated it's hard to figure out where to head in general."
>
> "Not having the resources or people to turn to makes life horrible, you have to turn to the manuals or books and keep going."

Information seekers also might choose not to go to an expert because he was socially difficult. Some experts have the reputation of having a difficult personality. (One interviewee just rolled her eyes when thinking of a particular X engineer.) This needs to be separated from the status interaction; it involves a related, but separate dimension of social interaction.

Software engineers, when faced with an information dysfunctionality, resort to some standard techniques. They read the documentation and books. But, often they approach the problem from trial and error. (User interface engineering is fairly amenable to this approach, given its iterative design techniques and modular engineering style.) One engineer, describing his actions when he was stumped, stated, "I try things, trial and error, okay it doesn't work, I'll attack it from another approach."

What many engineers did in this situation was either to try writing their own code or to decipher someone else's code. One engineer talked about writing lots of little test cases. Another offered that "I picked up what other people did and played." It is often the case that the engineers take portions of existing programs, found internally or on the Internet, and try to adapt them to their needs.

This trial and error, in the face of missing information, can be quite costly to an organization. One engineer, in a site with military secrecy and therefore information isolated, estimated that she spent up to 25% of her time trying to figure out how to make things work through trial and error. Other engineers' ballpark estimates were in the five to ten percent range. Over many engineers, this is a significant amount of effort, especially if the problem has already been solved elsewhere.

2.3. Information seeking and organizational memory

So far, I have concentrated in this chapter on seeking information from other organizational members. Organizational memory, however, includes both information databases and organizational members. How can a system to augment or facilitate organizational memory, either as database or as communication system, help information seekers? Interviewees offered several suggestions:

- It would be useful to have a better store of previously asked questions and answers. Said one engineer: "It seemed like we'd get runs of questions. An example? We'd get many questions about our tape drive all in a row - nothing before and after. No, it wasn't tied to the release. Then our expertise would slowly deteriorate. And then problems would come up again."

- It would be helpful to be able to explain some of the basic context behind a technical system such as the X Window System. There were multiple levels of context that would be useful.

    - Definitions. "Lots of things which make sense within [company name] seemed random outside. An example was how functions were named."

- Known problems. "A lot of the time you're trying to figure out - [is] it my problem or the system's problem?"

- The big picture. "The problem is learning what's global and what's not - it's a hard thing at first. What are the real consequences if I do this? There are answers and then there are real answers."

- Cross-effects and later implications. "I need to understand the consequences of necessary violations."

It is likely that definitions will be much easier to explain and capture than will contextual issues.

- ❑ It could be critical to tie an information seeker into the expertise network for the organization or community

  "Someone to help me? Oh yes, it would have gotten rid of all the little twigs. It's the twigs that take two days to clear up."

  and could save effort for the relative expert

  "I'm the only person with experience with X, so I looked through the code. I looked at the server for our Silicon Graphics."

- ❑ Having sample programs would help user interface engineers since they like taking other people's code and adapting it.

- ❑ It would be most advantageous for a system to reflect the fine-grained distinctions made by engineers when they seek

information. Engineers do not normally go to indiscriminate sources and they weigh the expertise level of potential information sources. Broadcast methods ignore these considerations. A typical response by an engineer was, "If I went to the net [network bulletin board], I'd get ten contradictory answers."

In summary, information seeking within an organization is a highly complex, and sometimes daunting, social activity. Substantial interaction with other people doing similar work would help. However, given the nature of the task, this is unlikely in the software engineering world (Hauptman 1986). It is even less likely with highly dispersed user interface engineers. The best possibility for resolving this dilemma is to improve the tools for accessing organizational memory.

2.4. Conclusions

In the last two chapters, I have argued that

- Information technology can support organizational memory in two ways, either by making recorded knowledge retrievable or by making individuals with knowledge accessible. To do so, an information system would need to incorporate elements of information databases and communication systems.

- Engineers currently go to the literature and experimentation first, as a way of avoiding the psychological cost of status interactions in information seeking. Additionally, they are quite sensitive to the norm of reciprocity in information exchange.

- In order to facilitate information seeking in the communication system, one should decrease the status implications and need for reciprocity.

In the following chapter, I will survey the technical literature to determine what types of systems might meet these criteria.

Chapter 3: Software Paradigms

Given that one might wish to augment the use of organizational memory, what software paradigms will help do so? Systems that could augment organizational memory must make recorded knowledge retrievable or make individuals with knowledge accessible. I will briefly survey four possible software paradigms that can be used for these purposes: information retrieval systems, bulletin boards and conferencing systems, help and consulting systems, and some computer-supported cooperative work systems.

3.1. Information retrieval systems

Information retrieval research is quite broad: The following section surveys the information retrieval literature in order to examine the role of information retrieval in an organizational memory system, but it does not attempt to detail the field.

This section is divided into three parts. First, I will describe the standard information retrieval model that has existed for at least 20 years. Second, I will turn that model into a user-centered model, making it easier to discuss the user's information seeking through the system. By making the model user-centered, I also attempt to escape the notion that information retrieval systems are primarily concerned with the retrieval engine, showing that other functions are also critical to their success. Next, I expand that model to include organizational considerations, noting the limitations of the information retrieval model in considering organizational memory.

3.1.1. What is information retrieval?

Salton (1989), as well as many other authors, argues that information retrieval systems are different from other types of information systems, especially database

management systems. Retrieval in database systems, Salton argues, is based on formal attributes. Information retrieval systems, on the other hand, deal with a set of identifiers, or index terms, based on the content of the document. He provides the following argument:

> ...Each index term is assumed to describe the text content only to some extent, not completely or unequivocally, and large numbers of different index terms may be attached to each particular document. ...it may not be useful to insist on a complete match between query and document terms.... Instead, the retrieval of an item may depend on a sufficient degree of coincidence between the sets of identifiers attached to queries and documents, produced by some approximate or partial matching method. (p. 230)

Other authors follow similar arguments. For example, Kraft (1985) argued that the determination of relevancy of the retrieved items to the query is less certain than in database or expert systems. Bartschi (1985) distinguishes database systems from information retrieval systems in that the former include formatted data and the latter include both unformatted and formatted data.

Interestingly, all three authors argue that information retrieval systems are a specialization of an information system, and that information systems can include database systems, management information systems, expert systems, and the like. They then distinguish information retrieval systems on the basis of the matching techniques. This may muddle the distinction between the technology of the information system and the context of usage.

3.1.2. The standard information retrieval model

Saracevic (1970) offered the following statement in 1970 about the nature of information retrieval systems (p. xxii):

> Here, "functions" are taken as being the major classes of action or, more precisely, the order of processes. It is believed that all information systems have the following five basic functions (loosely defined):

(a)	Acquisition	getting the material ... which at a minimum implies some selection process.
(b)	Information Representation	conceptual handling of the acquired material in some representative form and structure, which at a minimum implies a language (natural, artificial, indexing, classification, coding, etc.) or some combination of languages.
(c)	Organization of Files	storing of materials and/or storing of their representations, which at a minimum implies physical arrangement.
(d)	Question Handling and Search Procedures	getting whatever is in the system out in some organized and delineated fashion, which at a minimum implies the ability to search on a restrictive basis; i.e., it implies an output selection process.
(e)	Dissemination	distributing of the output, or displaying it in some organized form, which at a minimum implies arrangement of both a conceptual and physical nature and distribution patterns.

Such a model is a five-step batch system suitable for mainframes. Lancaster and Fayen (1973) modified the above model for on-line systems, offering the loop shown in Figure 3.1.

In Lancaster and Fayen's model, the user can use the previous retrieval's output to determine whether his information need has been satisified. If he is not satisfied, the model notes the possibility of refining the requirement and request to the system. In their diagram, the user is merely one of the steps in this process, but it would require only a slight modification to have the user driving the process.

An important difference between Saracevic's model and Lancaster and Fayen's model is that in Lancaster and Fayen's model, the retrieval process is limited to the collection at hand. Saracevic includes the acquisition of materials as intrinsic to the information system, although acquisition is an autonomous process.

Other writers, such as Kraft (1985), Bartschi (1985), Salton (1989), and Lesk (1991), stay with essentially Lancaster and Fayen's model (albeit providing more detail to the model). Kraft notes Saracevic's five step model as being appropriate for an information system, but then ignores the acquisition and dissemination steps in the rest of his review article of information retrieval systems. Lesk follows the Lancaster and Fayen

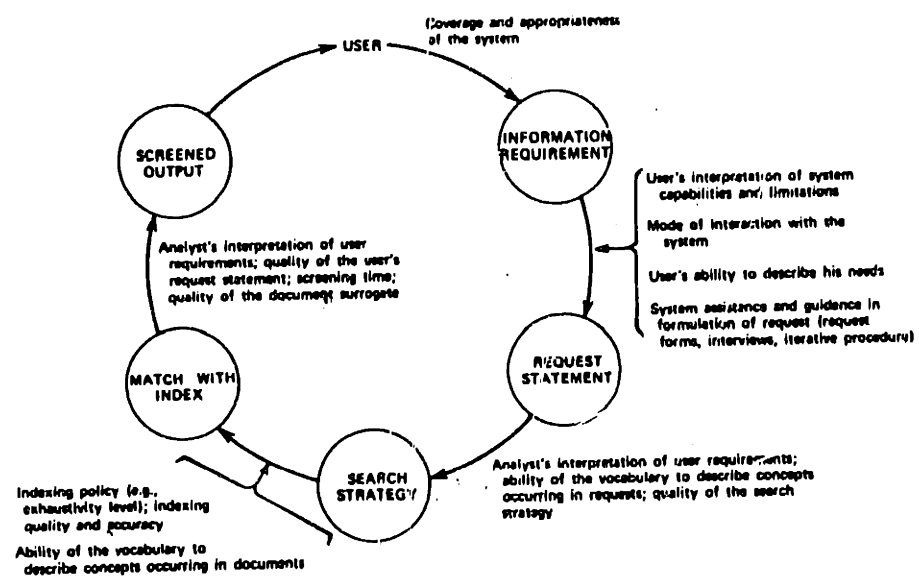

Figure 1.1. Factors affecting performance of an information retrieval system used in a delegated search mode.

Figure 3.1: Lancaster and Fayen model of information retrieval (Lancaster and Fayen 1973)

model implicitly, although he argues that extensions of the basic model will be critical if the research field is to remain vital.

3.1.3. A user-centered information retrieval model

A re-orientation towards the user's perspective lets one see the functionality required for an information retrieval system. The model in Figure 3.2 includes same functions as the Lancaster and Fayen model; I have rearranged the functionality as though being viewed by the user.

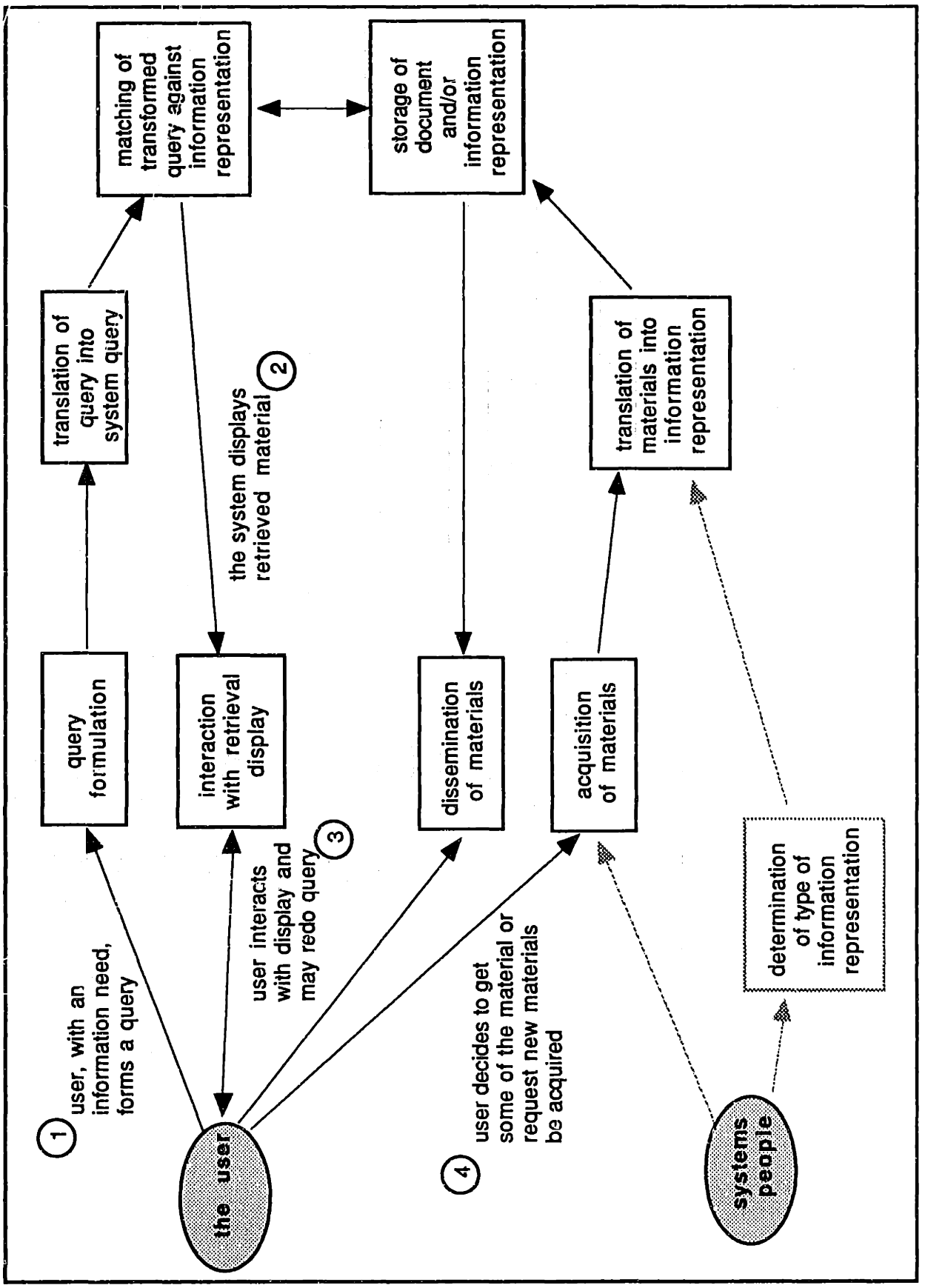

Figure 3.2: A user-centered model of information retrieval

In this model, the user can initiate one or more activities based on his information needs. Some of those activities can be system activities such as initiating a query process, interactively viewing an information display, and requesting the selected material. In other words, the user, in his mind, is formulating queries and seeing finished retrievals. How those retrievals are performed is extraneous; the technical components can be changed. Again, this model of the retrieval process is iterative -- no assumption is made about an ending step.

The standard model has been extended here to include a beginning and ending step. Before the process is begun, someone must acquire materials, and after the process is complete, the materials should be received by the user. I noted above that Saracevic and Kraft included these steps.

The contents of most of the boxes in the model are not important for the discussion here. For those interested, I briefly explicate each box in the figure in *Appendix A: A User-Centered Information Retrieval Model.*

One small extension to the standard model is easily obtained. If we assume that the acquisition and dissemination steps also hold iterative possibilities, the user should be able to ask for new information at any time. In fact, this raises an important question: how dynamic are the information store and other parts of the system? The user's actions are clearly dynamic and iterative. The standard information retrieval model views its collection as essentially static. This is not necessary; it is merely an implicit assumption following from the relatively static collection of library materials. An extension of the standard information retrieval model, then, is to make the information store substantially more dynamic and iteratively acquired.

3.1.4. An organization-centered model

Another extension to the standard information retrieval model is to consider an information retrieval system in the context of organizational use. Only a small portion of the information retrieval research literture addresses organizational issues (for example, Kwasnik 1989, Lansdale 1988, and Anick, Flynn, and Hanssen 1991). Figure 3.3 extends the above model to consider large numbers of organizational members using a retrieval process simultaneously or during some short period. The retrieval box in the organizational model follows directly from the user-centered model. It assumes that each member (or possibly teams of members) queries, retrieves, and views information from some information store.

However, there are additional functional components required when considering the organization as a whole, namely large-scale methods of acquiring and disseminating materials. Because this model no longer considers only a single user, and because the organizational members may not even know of one another, a suitable model requires methods for collectively authoring, storing, advertising, and communicating results. The advertising of organizational features is an important consideration since information may not be obvious to all information members. In terms of system components, the information retrieval paradigm, extended to organizational use, must include an authoring and production system, information storage, retrieval engines, and the organizational communication system.

Once information retrieval is extended to the organizational level, one can also argue that information seeking should not be restricted to information objects such as documents. Information seeking, as was shown in the last chapter, includes people. Our retrieval process, portrayed above as retrieving from a system, should include both retrieval systems and organizational members.

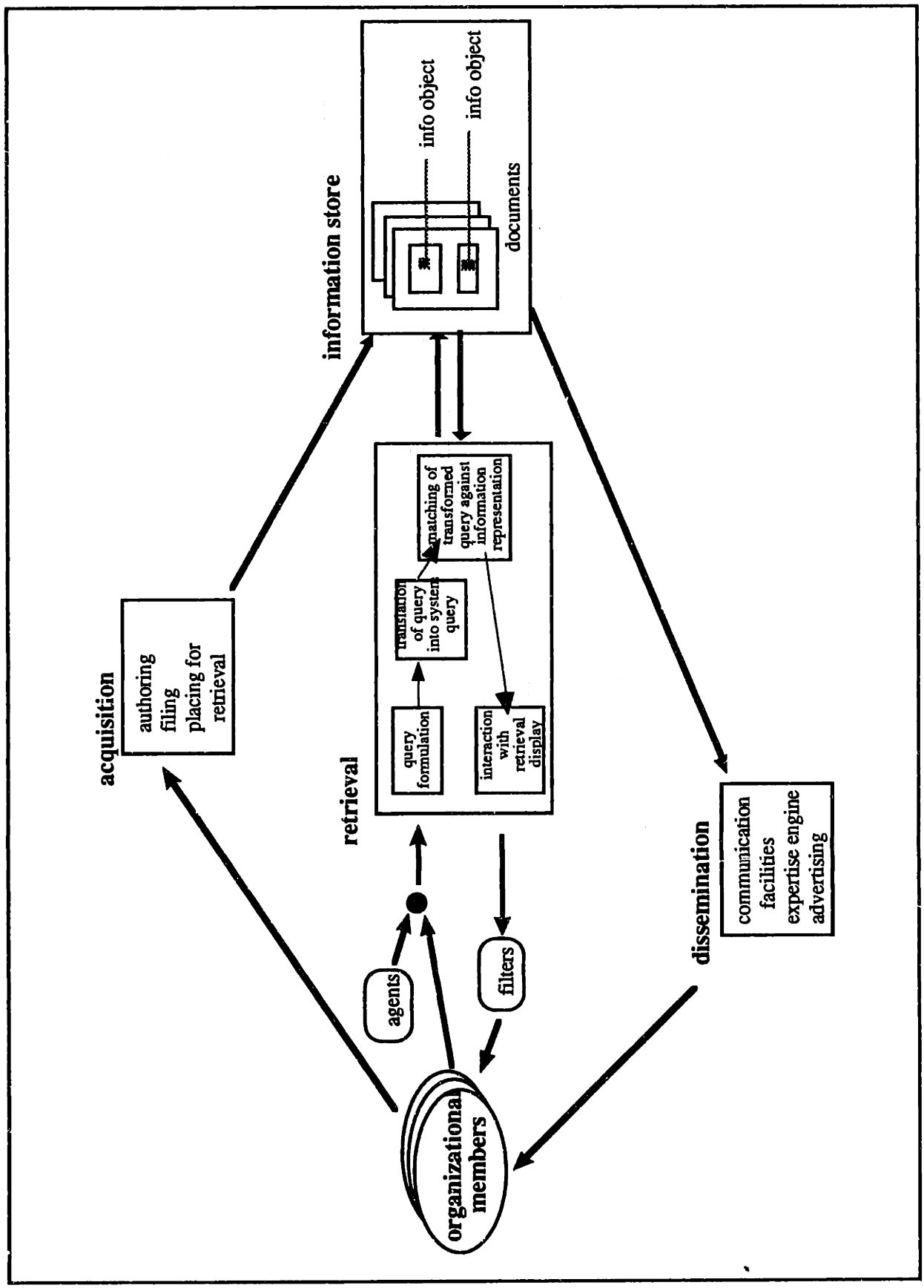

Figure 3.3: An organization-centered model of information retrieval

The standard information retrieval model, then, must be extended in order to handle organizational memory considerations. The most important extended feature is to include communication facilities. In addition, it must be also extended to handle iterative acquisition, hetereogenous information objects, and production processes.

3.2. Bulletin board and conferencing systems

Bulletin board and conferencing systems, such as the Usenet bulletin boards, approach organizational memory from the other side. They have communication facilities to help locate individuals with knowledge, but they lack the generalized memory and archival capabilities that are necessary to store and retrieve information objects for future reference. Examples of bulletin board and conferencing systems include the Usenet bulletin board system (Horton and Adams 1987), Confer (Advertel Communication Systems 1988), TEAMate (MMB Development Corporation 1991), VAXnotes (Essick et al. 1983), Discuss (Raeburn et al. 1989), and EIES (Hiltz and Turoff 1981).[1]

Bulletin board systems are typically arranged into topics. An example from Usenet is shown in Figure 3.4. Within these topics are messages (Figure 3.5), which are the basic data units of bulletin boards and conferencing systems. Related messages within a topic may be arranged into "threads." The topics and threads can be arranged in a tree structure (usually represented on the screen as an outline or menu) with topics clustered into categories created by the system administrator. Communication is usually

---

[1] Bulletin board and conferencing systems are one form of computer-mediated communication systems (CMCS). Unlike the information retrieval community, there is no CMCS *technical* community. Although there is a flourishing research literature studying these systems from a social perspective, there are relatively few technical papers. I have, therefore, abstracted a paradigm from representative systems.

```
journeyman:/afs/athena.mit.edu/user/a/c/ackerman
********   5 unread articles in athena.housing--read now? [ynq]
********   1 unread article in mit.lcs.announce--read now? [ynq]
********  12 unread articles in mit.lcs.misc--read now? [ynq]
********   4 unread articles in mit.lcs.seminar--read now? [ynq]
********  45 unread articles in comp.archives--read now? [ynq]
********  14 unread articles in comp.archives.admin--read now? [ynq]
******** 382 unread articles in comp.bbs.misc--read now? [ynq]
******** 438 unread articles in comp.bbs.waffle--read now? [ynq]
********  15 unread articles in comp.cog-eng--read now? [ynq]
******** 242 unread articles in comp.human-factors--read now? [ynq]
******** 194 unread articles in comp.infosystems.gis--read now? [ynq]
******** 126 unread articles in comp.infosystems.gopher--read now? [ynq]
********  53 unread articles in comp.infosystems.wais--read now? [ynq]
********   7 unread articles in comp.internet.library--read now? [ynq]
********   3 unread articles in comp.ivideodisc--read now? [ynq]
******** 134 unread articles in comp.multimedia--read now? [ynq]
********  33 unread articles in comp.newprod--read now? [ynq]
********   9 unread articles in comp.patents--read now? [ynq]
********  12 unread articles in comp.research.japan--read now? [ynq]
********   7 unread articles in comp.society--read now? [ynq]
********  20 unread articles in comp.society.futures--read now? [ynq]
********  81 unread articles in comp.society.privacy--read now? [ynq]
********  12 unread articles in comp.soft-sys.nextstep--read now? [ynq]
******** 165 unread articles in comp.sys.laptops--read now? [ynq]
```

Figure 3.4: News groups, or topics, in Usenet

asynchronous, but in some conferencing systems, there may be synchronous or "chat" capabilities.

Entries in topics can include other types of information, which can be downloaded to the user in various ways. It is often, but not necessarily, the case that multimedia information cannot be accessed directly in the bulletin board, but must be downloaded to the individual's computer for viewing and manipulation. None of the examined systems allows object modules or other active information to be run in-place within the bulletin board system, but this would be a minor extension of the software paradigm.

The basic distinction between information retrieval systems and bulletin board systems is the emphasis on messaging in the latter. While some bulletin board systems include full-text searching, they do not emphasize retrieval of discrete and segmented

```
journeyman:/afs/athena.mit.edu/user/a/c/ackerman
Article 58825 (534 more) in comp.windows.x:
From: bmott@garfield.catt.ncsu.edu (Bradford Mott)
Subject: Security
Organization: Computer and Technologies Theme Program, NCSU, Raleigh
Distribution: usa
Date: Tue, 22 Sep 1992 13:50:53 GMT
Lines: 16


        I was trying to use the MIT-MAGIC-COOKIE-1 security mechanism
on a DECstation, but I ran into a problem.  Some X window programs
will not run unless you do an 'xhost + localhost'.  Is there an easy
way to fix these clients to make them work with the MIT-MAGIC-COOKIE-1
security system?  I really need to get xv working.

Any help will be greatly appreciated.


Brad

-----------------------------------------------------------------------------
     Bradford W. Mott                                bmott@catt.ncsu.edu
Vice-President, Computer and Technologies Theme Program, NC State University
--MORE--(93%)
```

Figure 3.5: A message about the X Window System from Usenet

information. Nor do they use advanced information retrieval techniques. Instead, the emphasis is on the topic or thread, which forms a conversation among users.

Because of their emphasis on messaging, bulletin board systems are inherently multi-user. Nonetheless, while bulletin boards do "narrow-cast" messages to selected audiences, these audiences self-select themselves, thus making the system difficult to use for information seeking or organizational memory. If an organizational member wishes to ignore a question, or does not belong to a bulletin board group, the query goes unanswered.

Users of bulletin board systems have two common complaints. First, when one has asked a question, there is seldom a way to distinguish a correct answer from incorrect responses. Moreover, there are usually many incorrect responses for each correct one.

As one interviewee said: "I would post [my question] to the net and wait a month .... [then] read my 300 wrong answers."

Second, the discussions and questions tend to be repetitious over time. These problems become significant as the number of users on the system grows, and is a frequently mentioned criticism of the Usenet bulletin board system. Two responses have arisen. First, some topics have moderators who monitor discussions. On some bulletin board systems (although not on Usenet), moderators may remove incorrect responses. Second, in the last two years, individual users on the Usenet bulletin board have begun to compile Frequently Asked Questions (FAQ) lists. The FAQs are posted approximately every month on a topic. An excerpt from the Usenet bulletin board topic concerning the Open Look flavor of the X Window System is shown in Figure 3.6. Recently, Usenet has begun a new topic that just reposts FAQ lists from the other Usenet topics.

3.3. Consulting systems

Another possibility for organizational information seeking is consulting and help systems. Typically, help systems are either part of other applications, such as word processing systems, or they are stand-alone documentation systems. At the lowest level, help systems do not vary considerably in their paradigm from information retrieval systems. The emphasis, however, is on help to users, and thus, systems carry more tutorial material and material is often educationally arranged (Borenstein 1985). These systems are also functionally similar to document (manual) browsing systems (e.g., Walker 1987, Campagnoni and Ehrlich 1989).

Research in help and consulting systems has been active in the computer-human interaction community; the emphasis was on providing user assistance through expert systems technology (see for example, Carroll and McKendree 1987 and Pollack 1985). An example of this type of consulting system can be found in Gwei and Foxley 1990.

```
journeyman:/afs/athena.mit.edu/user/a/c/ackerman
Article 59276 (533 more) in comp.windows.x:
Newsgroups: comp.windows.open-look,comp.windows.x,alt.toolkits.xview,comp.window
s.x.intrinsics
From: lee@sq.sq.com (Liam R. E. Quin)
Subject: OPEN LOOK GUI frequently asked questions
Organization: SoftQuad Inc., Toronto, Canada
X-Feet: bare
Date: Sat, 3 Oct 92 23:49:28 GMT
Lines: 1014


Frequently-asked questions about the OPEN LOOK Graphical User Interface.

If your question isn't here, please try (at *least*)
        man openwin
        man xnews
        man 7 xview
and looking in the appropriate manuals listed in the Bibliography below
before posting to the comp.windows.open-look, comp.windows.x or
alt.toolkits.xview newsgroups and/or their corresponding mailing lists.

Frequently Asked Questions for X11 are posted to comp.windows.x monthly.

This is version: $Revision: 1.38 $; Last posted version:  1.34
I am posting the list of differences since last post separately this month.

Contents: (in rn and trn you can use control-G to go to the next topic)

    Subject: Terminology: OPEN LOOK, OpenWindows, X11, XView, (MO)OLIT, Motif
    Subject: Window Managers -- olwm, olvwm
    Subject: OpenWindows, Terminals, and Other Displays
    Subject: Configuration Files: Getting started with OpenWindows
    Subject: Key Bindings, Cut and Paste
    Subject: Applications: Finding Out...
    Subject: Troubleshooting: It Won't Let Me Type
    Subject: Troubleshooting: Not authorized to use display
    Subject: Trouble Shooting: other common problems
    Subject: Trouble Shooting: XView problems
    Subject: Fonts
    Subject: Environment Variables
    Subject: Where Can I get It? Ftp, implementations, etc...
    Subject: Bibliography -- books, manuals, journals, papers, beer-mats
    Subject: Getting this File, Revision History, Recent Changes
--MORE--(4%)
```

Figure 3.6: FAQ from comp.windows.x on Usenet

Gwei and Foxley describe a system that used user models and natural language to provide additional context for users' questions. Other similar systems include Fischer's design critics, which serve a consulting role in architectural design (Fischer and Lemke 1988).

A notable exception to the expert consulting system paradigm is the On-Line Consulting System (Coppeto, Anderson, and Geer 1989; Acevedo 1991). OLC does not attempt to automate the expert's role in the consulting process; instead, it attempts to automate the mechanics of placing a question and receiving an answer in a distributed computing environment. OLC is modeled on a student consulting desk, with a line of users stretching out the door and a few advanced students answering questions. The system has a relatively simple front-end, allowing the user to write messages, read answers, and mark inquiries as finished. The users determine the category for their inquiry, selecting from a menu. Figure 3.7 shows the initial menu in the X interface. (A terminal-based interface also exists.) As questions come into the system, consultants select what questions they wish from the system queue.

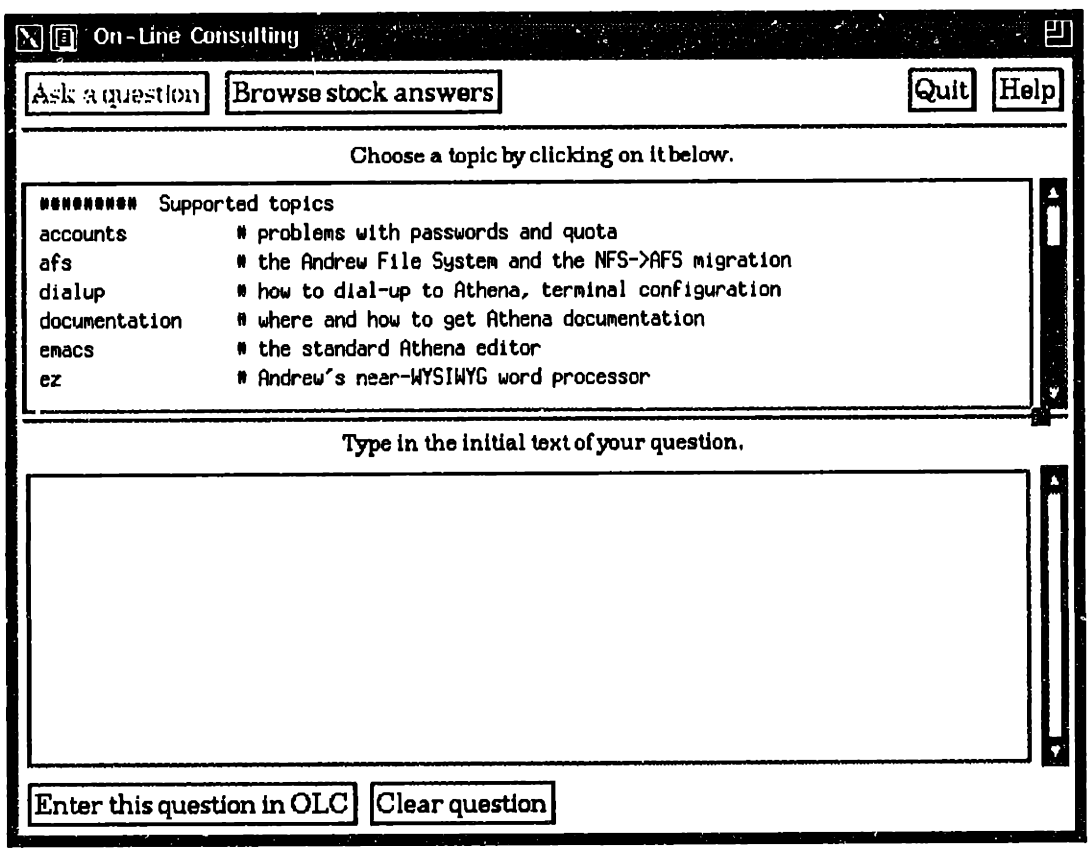

Figure 3.7: Initial OLC menu

There is also the ability to see the most commonly asked questions in each category. In the MIT installation of OLC, there are currently 27 categories, with approximately 10 to 30 questions and answers per category; these numbers have slowly risen over time. Figure 3.8 (X interface) and Figure 3.9 (terminal interface) show some of the stock questions and answers for the X Window System. In addition, the consultants maintain their own databases with additional information.

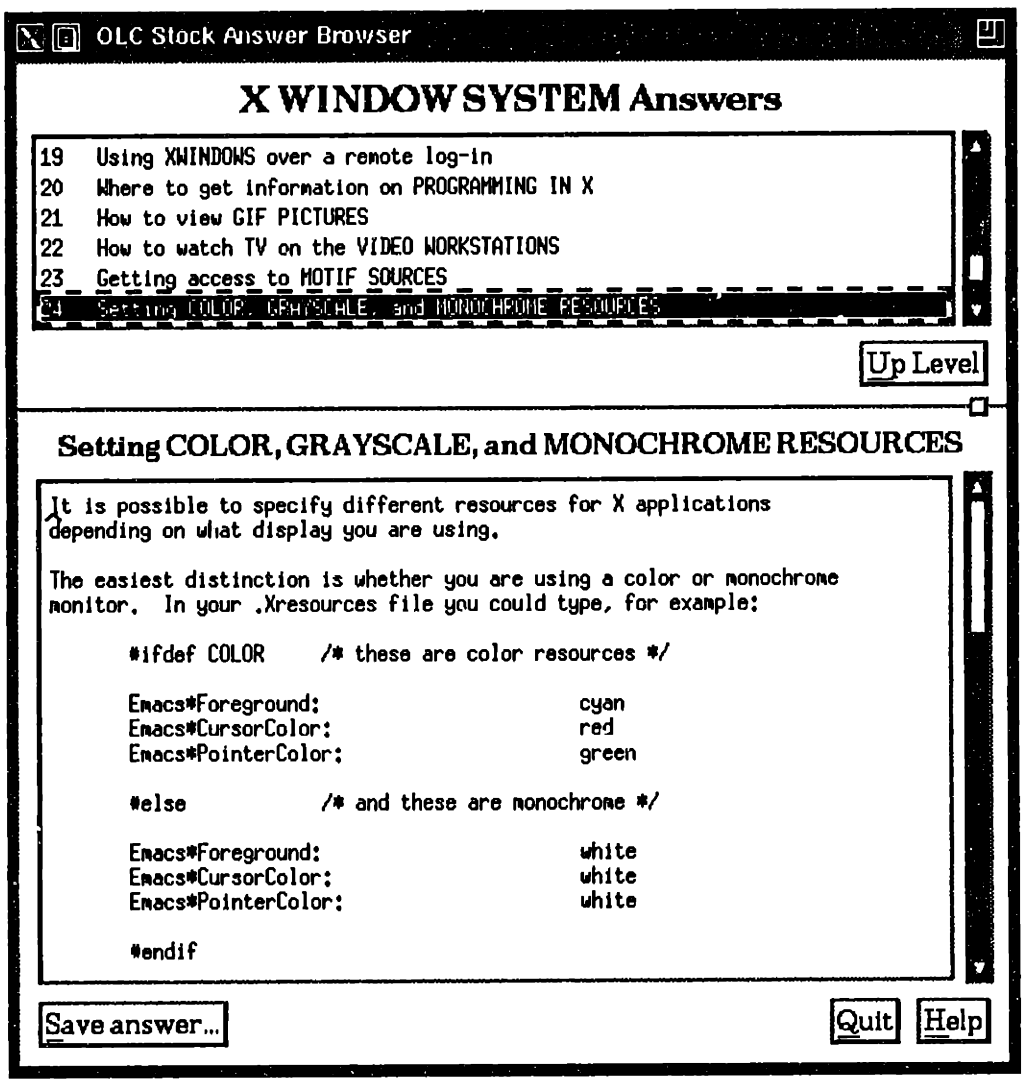

Figure 3.8: Stock answers about the X Window System in OLC (X interface)

OLC has some of the features one would like in an organizational memory system. It has a store of commonly used information for a computer site. However, it lacks several critical features. OLC does not allow the end user to query most of the

information, relying instead on trained intermediaries. Its emphasis is not on building an organizational memory since its result is not intended to be a retrievable corpus of information or accessible individuals. Rather, OLC concentrates on the mechanics of getting users answers to their questions from help desk personnel. In doing this, it uses the communication facilities of the organization in only a very primitive manner.

```
journeyman:/afs/athena.mit.edu/user/a/c/ackerman

                    On-Line Consulting Browser
                /mit/olc-stock/stock_answers/xwindows

          1  What is MWM?
          2  How to CREATE a window
          3  MOVING windows with MWM
          4  How to get a CLOCK on the screen
          5  How to configure the new XLOGOUT
          6  Working with the new CONSOLE WINDOW
          7  How to get a LOAD average graph
          8  How to get XSCREENSAVER
          9  How to make an xterm REVERSE VIDEO
         10  How to specify the LOCATION of windows
         11  Changing the KEYCLICK volume
         12  xdpr gives 'PARAMETER MISMATCH'
         13  Using CUT & PASTE between windows
         14  Using X FONTS
         15  X Font name FIELDS
         16  How to use XSETROOT to change root window
         17  How to use XPIX
         18  Getting a SCREENDUMP or WINDOW DUMP
         19  Using XWINDOWS over a remote log-in
         20  Where to get information on PROGRAMMING IN X
         21  How to view GIF PICTURES
         22  How to watch TV on the VIDEO WORKSTATIONS
         23  Getting access to MOTIF SOURCES
         24  Setting COLOR, GRAYSCALE, and MONOCHROME RESOURCES
         25  How to use COLOR in applications
         26  How to make ALT FUNCTION work like COMPOSE CHARACTER
         27  How to display a current weather wap

                        ** End of Index **

olc_browser>
```

Figure 3.9: Stock answers about the X Window System in OLC (terminal interface)

A consulting system that uses both expert-systems technology as well as human consultants was reported in Ram et al. 1989 and Ram, Hayne, and Carlson 1992. ICE/H is an intelligent help system combined with a consulting system. At the core of ICE/H is

its knowledge base that attempts to emulate human consultants in particular computing domains. However, if the problem is beyond the knowledge base, the user can ask an appropriate human consultant through electronic mail. While communication with the human consultants is possible, the emphasis in ICE/H appears to be the knowledge base and expert system. For new problems, knowledge engineers are required. ICE/H is intriguing because it combines elements of communication with a help system in order to relieve some of the consultants' load. Unfortunately, it does not lead to an organizational memory without the extensive intervention of experts and highly trained intermediaries. ICE/H is apparently used in several internal IBM sites.

3.4. CSCW notebook systems

There is a set of systems that fall under the rubric of computer-supported cooperative work (CSCW). Below I discuss four particularly interesting systems for supporting organizational information seeking and organizational memory. This set includes Lotus Notes (Lotus Development Corporation 1991), the Virtual Notebook System (VNS) (Gorry et al. 1988), gIBIS (Conklin and Begeman 1988, Conklin 1992) and Oval (Lai, Malone, and Yu 1988).

These systems are quite difficult to sort into one specific paradigm. Two of them, Oval and Lotus Notes, are general-purpose development platforms for group or organizational applications; the others are specialized systems. Two of them, VNS and gIBIS, are specifically designed to augment information publishing; the other two can be used for the same. Three of them, VNS, Oval, and gIBIS, use hypertext technology (or extensions of that for Oval). Instead of offering a general paradigm for the four systems, I will, instead, discuss each in turn.

The Virtual Notebook System (VNS) is, as Gorry et al. state, "...a technologically-extended analog to the ordinary notebook ... to help the members of a biomedical group

coordinate efforts and share information and to improve group functioning through the automatic importation of relevant information from external sources such as libraries" (p. 39). VNS data can consist of text or images, and can be shared or private. Functionally, VNS is similar to the information sharing characteristics of a bulletin board system; technically, however, it is more similar to a hypermedia system.

Lotus Notes is a relatively new commercial product, innovatively combining elements of bulletin board systems and information managers in a networked environment. Notes' most important data abstraction is the information object, which could be a message, form, or document. Notes can also handle linkages through Microsoft's Dynamic Data Exchange mechanism. Information objects can be collected into databases. Information objects are displayed according to presentation objects called Forms for individual objects or are summarized in Views. Views are limited essentially to an outline presentation, sorted according to set of categorizations, including any field data or calculations. The current version, version 2, does not allow users or authors to extend the user interface presentation or authoring components. Users and authors must use the outline presentation; however, there are mail links both into and from Notes. Version 3 should have more capability for external "hooks."

From these simple data abstractions, one can compose a large number of applications. Potential applications prominently promoted include organizational communications (such as a Rumor Hotline), bulletin boards, technical support (the Help Desk), expertise sharing, information publishing, and organizational directories.

gIBIS is a design rationale system that utilizes hypertext features to document design considerations over time. The tool is a specialized hypertext that attempts to capture "design problems, alternative resolutions (including those which are later rejected), tradeoff analysis among these alternatives, and records of the tentative and firm

commitments that were made as the problem was discussed and resolved" (Conklin and Begeman, p. 140).

The system uses a rhetorical model called IBIS that decomposes design into a set of Issues, Positions, and Arguments. The Issues can include "any problem, concern, or question..." (p. 140). The Positions are statements that offer potential resolutions of the Issue, and the Arguments provide supporting or contradicting evidence. A specific Issue and its Positions and Arguments form a tree, and the links among the parents and children are typed according to the IBIS method.

Oval is based upon a system of objects, views, agents, and links. From these four primitives, a large number of applications can be constructed including most of gIBIS and Lotus Notes (Malone, Lai, and Fry 1992). (The single user functionality of VNS could also be emulated.) While Oval lacks some of the information publishing mechanisms and facilities of Lotus Notes and VNS, it provides the additional abstraction of an object-oriented information environment as well as user-defined agents. Since Lotus Notes allows the construction of a large number of applications potentially useful for organizational memory, Oval, by extension, also facilitates these applications.

These four CSCW systems provide functionalities that will be required in a system to facilitate organizational memory. Combining elements of information retrieval and bulletin board systems, they add information publishing mechanisms, semi-structured objects, considerations of information authoring, and the like. However, system that offered direct support of organizational memory for information seeking would need the information publishing and organizational elements of Lotus Notes and VNS as well as the object-oriented flexibility of Oval. One would also like to capture organizational memory in a way that was as easy and seamless as gIBIS claims in its promotional literature.

3.5. Summary

In this chapter, I have reviewed four types of systems. Information retrieval systems were seen to ignore organizational considerations including information acquisition, information publication (and publicity), information-seeking from organizational members, and ties to organizational communications. Bulletin board and conferencing systems were seen to ignore information retrieval and access. Consulting and help systems were seen to be specializations of either information retrieval systems or bulletin board systems.

The four CSCW systems, while lacking some of the coherence of a software paradigm, included elements of information retrieval and bulletin board systems to provide the basic facilities for organizational memory applications. In the next chapter, I consider some new organizational memory applications. In the chapters following, I discuss a system that includes the features required to implement those applications.

Chapter 4: Three Organizational Memory Applications

I argued in chapters 1 and 2 that a useful organizational memory system would need to combine elements of information retrieval and communication systems. In addition, it would be useful for an organizational memory system to facilitate some of the social interactions in an organization in order to promote better information seeking. In the last chapter, I examined several software paradigms to examine their support for these considerations. In this chapter, I describe three organizational memory applications, all of which have, to varying extents, support for these criteria.

The following three organizational memory applications were all built from a common underlying system, the Answer Garden Substrate (AGS). An overview of AGS follows in chapter 5.

It should be mentioned that all three applications were built using only AGS. They are, therefore, written in C, using only the X Window System and standard Unix routines. In addition, the first two have been extensively field tested; in fact, both are in daily use world-wide. The third exists in prototype.

All of these applications lead the users to build organizational memories as by-products of their normal activities.

4.1. Application 1: Answer Garden

Answer Garden (Ackerman and Malone 1990) is a system for allowing users to find answers to commonly asked questions. The information in an Answer Garden grows "organically" over time as the result of users and experts interacting through the system,

resulting in an organizational memory. Answer Garden is most useful in situations where there is a repetitive stream of questions, many of which the organization as a whole has seen before, but some of which are new.

Let me start the description of Answer Garden by offering a few common organizational situations:

- You have just traveled to a foreign country. Dimly you remember that the reimbursement form to fill out is different. What form is it, and where can you find it?

- You cannot get your MIS-approved word processor to wrap text in two columns below a banner graphic. Where do you find the answer to your specific problem?

- You are working on a proposal for your consulting company. You suspect you are not the only person who has done this type of proposal in your company. How do you find the other people who have relevant expertise?

All of these situations have some common characteristics: They are all situations where there is a commonly asked question. In each situation, someone needs directions to specific information. The directions may be to people or to data and text, but they are all pointers that the information seeker does not have. Moreover, the questions as a whole are unpredictable and new to each individual; to the organization, however, they are repetitious. Finally, they are all situations that are dynamic; the information can change or be modified rapidly.

4.1.1. *An Answer Garden session*

A typical Answer Garden session is shown in Figures 4.1 through 4.4. Figure 4.1 shows the user's initial view of Answer Garden. This first node describes the particular information database being used; in this situation, the information database is about the X Window System.[1] The warning to the user about her being monitored is required by MIT rules.

There are several different ways to find information in Answer Garden. One way is to travel down a set of diagnostic questions that lead the user to the information sought (Figure 4.2). It is similar to the system playing the game "Twenty Questions." The user traverses the question nodes, selecting the appropriate button with his mouse. Figure 4.2 shows the screen after the user has selected several questions. At the end of these diagnostic questions, the answer is a particular piece of information. In this example, the user is looking for the name of an X graphics program.

In this particular information database, answers can either be specific pieces of information (as in Figure 4.2), or they can be more general groupings of opinions, tutorials, and code examples. However, there is no restriction on the type of information or display for the node. The end node could also be a picture of a person, a program segment to dynamically create or retrieve the information, a set of questions and answers, an advertisement for a distributed service, a tutorial, or anything else. (Answer Garden is quite extensible by application programmers.)

---

[1]The information database about X is only a sample database. Two computer science departments are building databases about their classes and requirements, and various organizations have built information databases about system administration. There are also databases on a hardware product and on a software system.

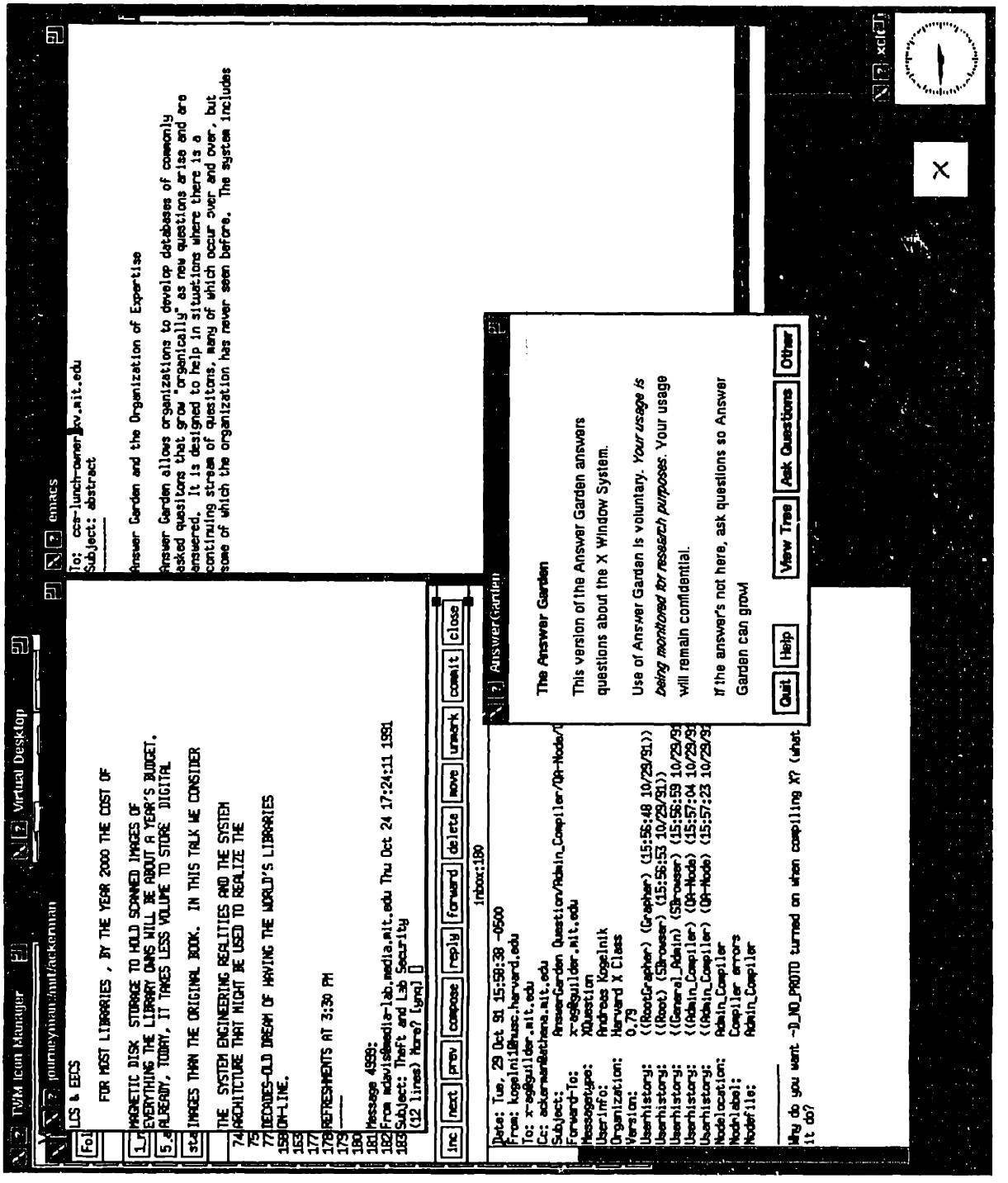

Figure 4.1: Initial screen for the Answer Garden application

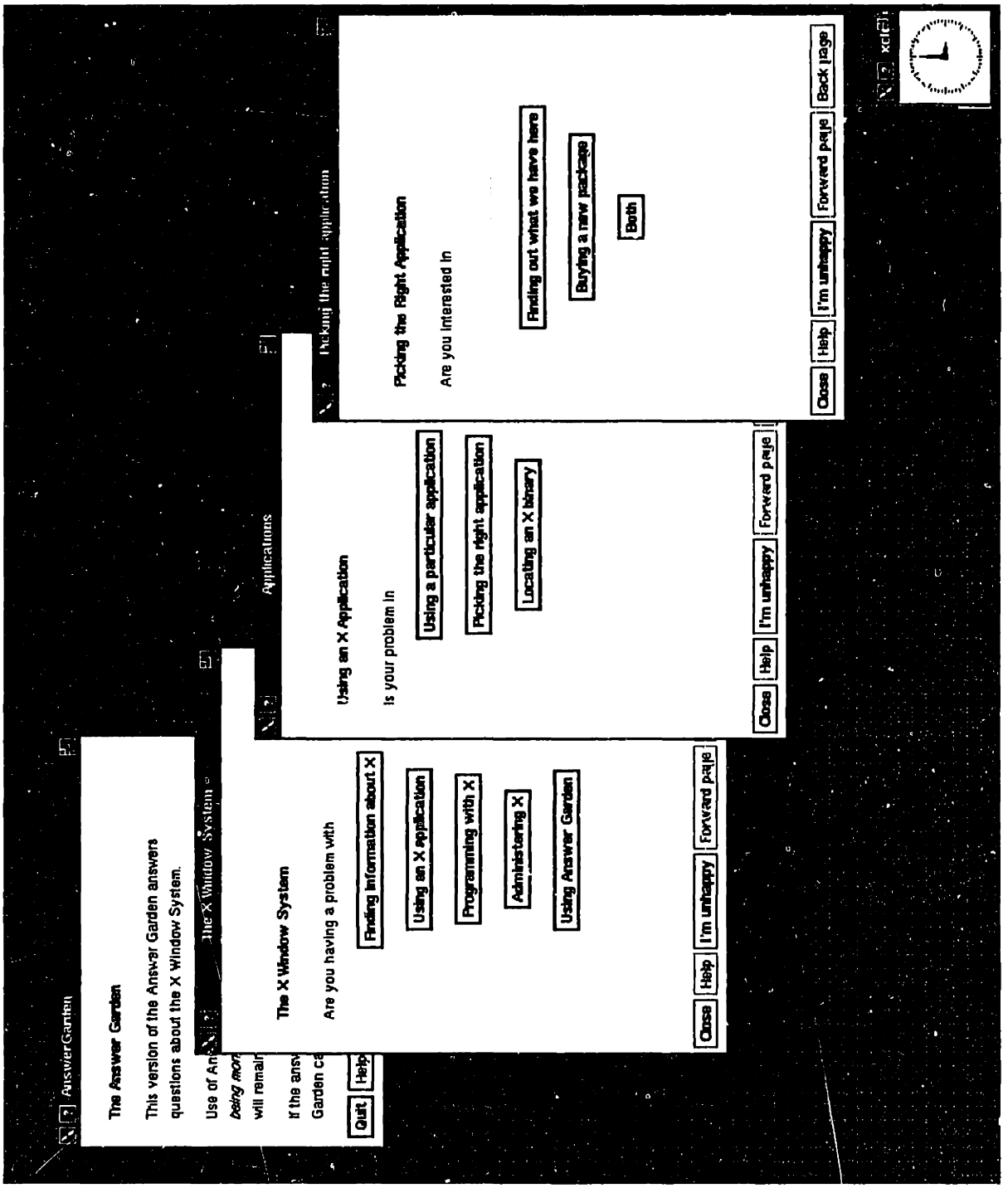

Figure 4.2: Diagnostic questions in an Answer Garden session

Figure 4.3 shows a second way of searching for the information. More experienced users may wish to go directly to the information, and they can use the grapher view of the same diagnostic questions. (In some Answer Garden applications, the grapher view is a reference view, and therefore offers an alternative to the diagnostic questions.) The network of diagnostic questions is projected into a tree to ease navigation for the user. The "..." indicator in some tree items indicates there is a subtree as well. Figure 4.3 shows a situation where the user has selected the Answer Garden tree, and then selected a specific answer. (It is the same answer he would have selected from the diagnostic question series.)

From the "Other" menu on the beginning node, the user can select additional information retrieval engines. Currently, Answer Garden has a free text retrieval engine, a keyword engine, and an adaptive retrieval engine. The structure of Answer Garden makes it possible to include almost any information retrieval engine for use.

A feature that makes Answer Garden unique is the "I'm Unhappy" button on each node. If the user cannot find the answer, does not understand the diagnostic questions, or is lost, the "I'm Unhappy" button serves as a way to seek human assistance. In the case of an unanswered question, the user asks her question in the popup as shown in Figure 4.4.[2] The user asks her question. Notice that the node expert is anonymous for the user. The header for the electronic mail message is replaced before being sent with the electronic mail address of the correct expert or set of experts, a user history so the expert can

---

[2]Answer Garden also has synchronous communication using the Zephyr communications package at MIT. Answer Garden, at MIT, attempts to contact the expert synchronously; if that fails, the expert is sent the same message via e-mail. Telephone and voice mail support are possible extensions, again blurring the line between asynchronous and synchronous communication in the system.

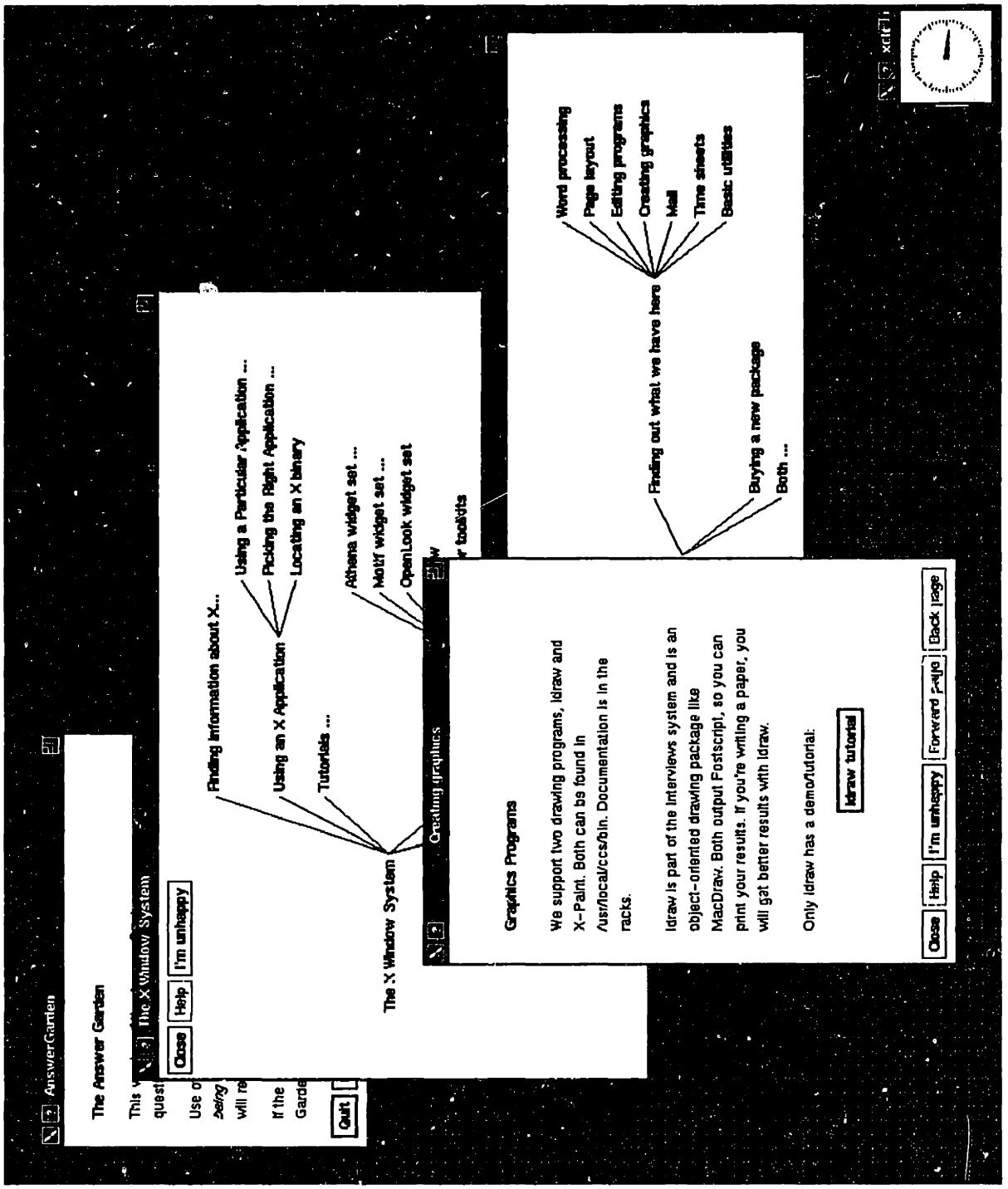

Figure 4.3: The diagnostic tree in Answer Garden

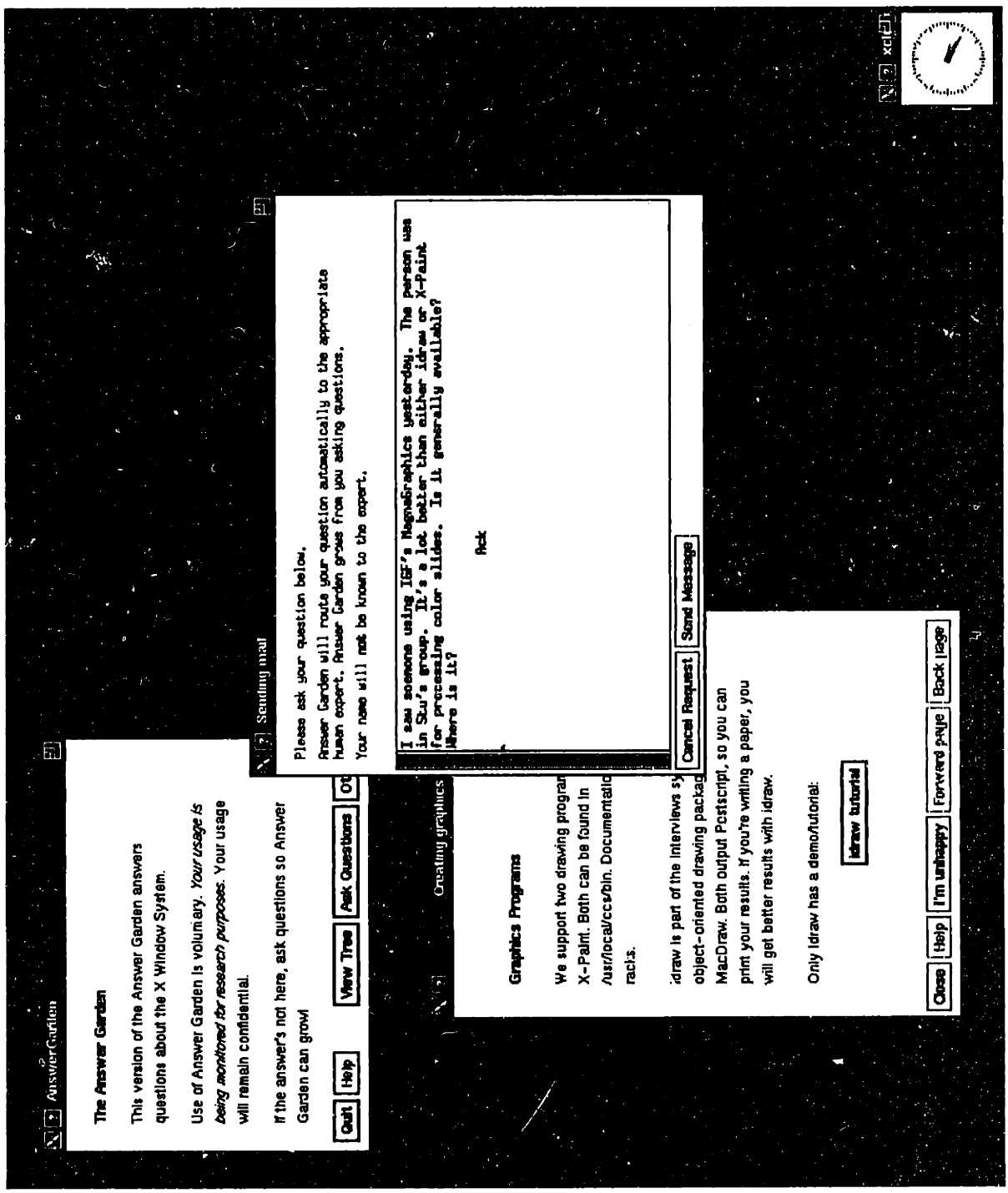

Figure 4.4: Mailing a question in an Answer Garden session

determine where the user has been and what she has seen, as well as other, miscellaneous information.

Answer Garden then routes the user's question to an appropriate human expert. The human expert then answers her directly.[3] If the question is a common one, the expert can also insert the answer into the database. As well, the expert can add any diagnostic questions that he feels might be necessary.

Answer Garden thus provides a mechanism for growing a body of information over time. (Hence the name.) Both experts and users perform their normal duties, and Answer Garden provides incentives for both the experts and the users. While users have to first browse the database to find the answer, they gain the ability to find the right person of whom to ask their question. This is often a considerable problem in many organizations. While the experts must structure the database, they gain the capability to rid themselves of the most commonly asked questions. As a whole, the organization, group, or community gains a corpus of information, an organizational memory.

4.2. Application 2: the ASSIST

The ASSIST is an application, developed by the Harvard-Smithsonian Astrophysical Observatory, that allows astrophysicists to easily find documents, data files, and software components (Mandel et al. 1992, Brissenden 1992). Parts of the ASSIST are shown in Figures 4.5 through 4.8. Figure 4.5 shows part of the interface for locating and using about one thousand software components (mostly for data analysis)

---

[3]The expert normally answers via electronic mail. Usually, the user is anonymous. In that case, the question and the expert's answer go through an intermediate layer. If the user is not anonymous, the expert could also phone or even show up in person.

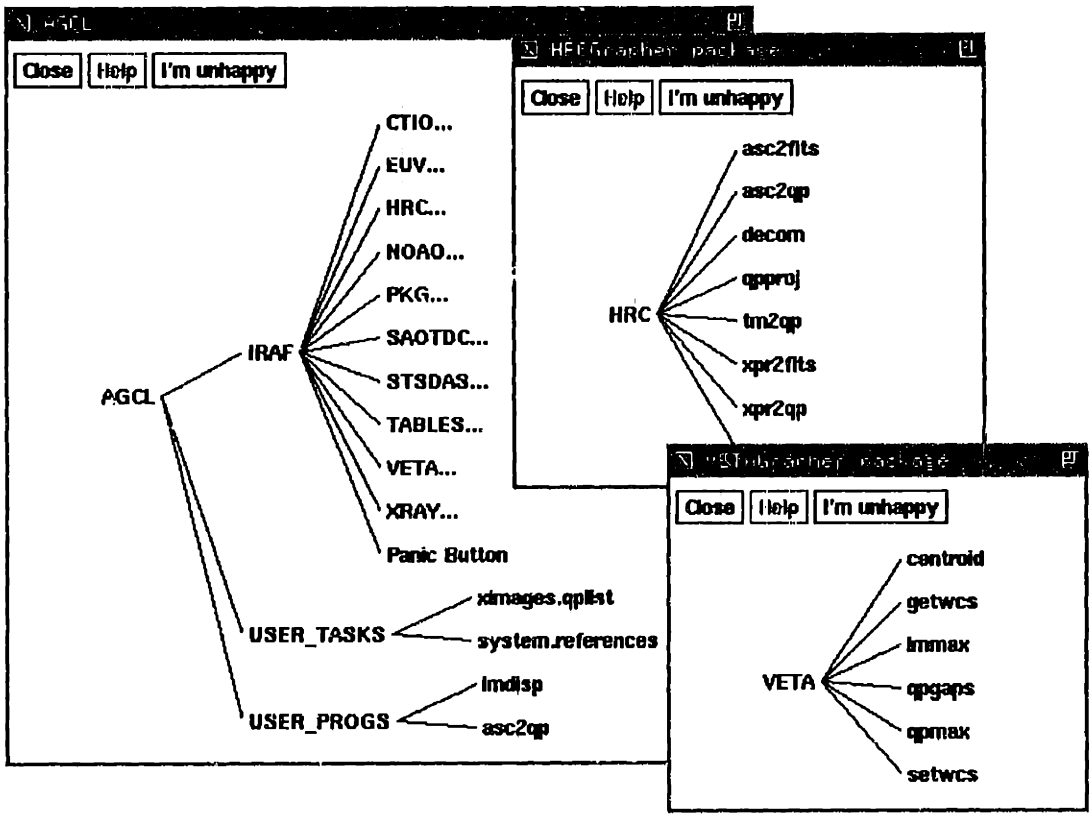

Figure 4.5: Interface within the ASSIST for using data analysis modules

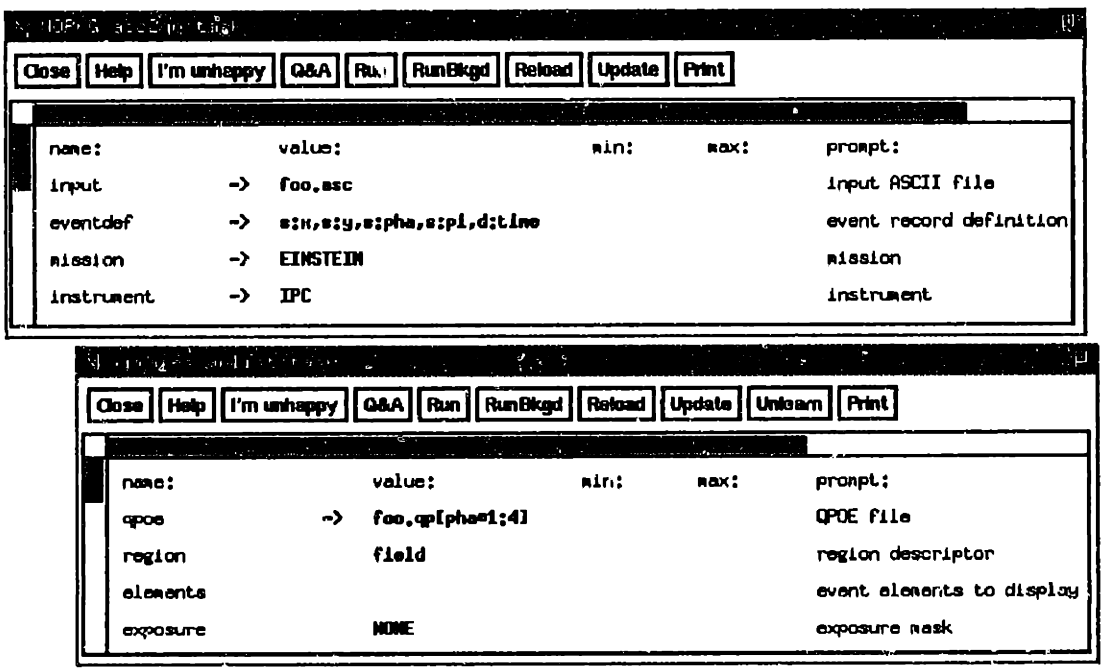

Figure 4.6: ASSIST's parameter editors for controlling software modules

and the associated documentation. The tree structures include the various tasks that the astrophysicist can use. The parameter editors (shown in Figure 4.6) are the graphical representation of the system tasks; the window for inspecting the results is not shown.

In essence, the application writers embedded the data analysis software components within a surrounding help and documentation system. To the ASSIST user, information objects consist not only of questions and answers, documentation, help, and other standard information objects; they also include data analysis packages (at the subroutine level), parameter editors, and testing software. Figure 4.7 shows the interface to part of a document collection about these software modules.

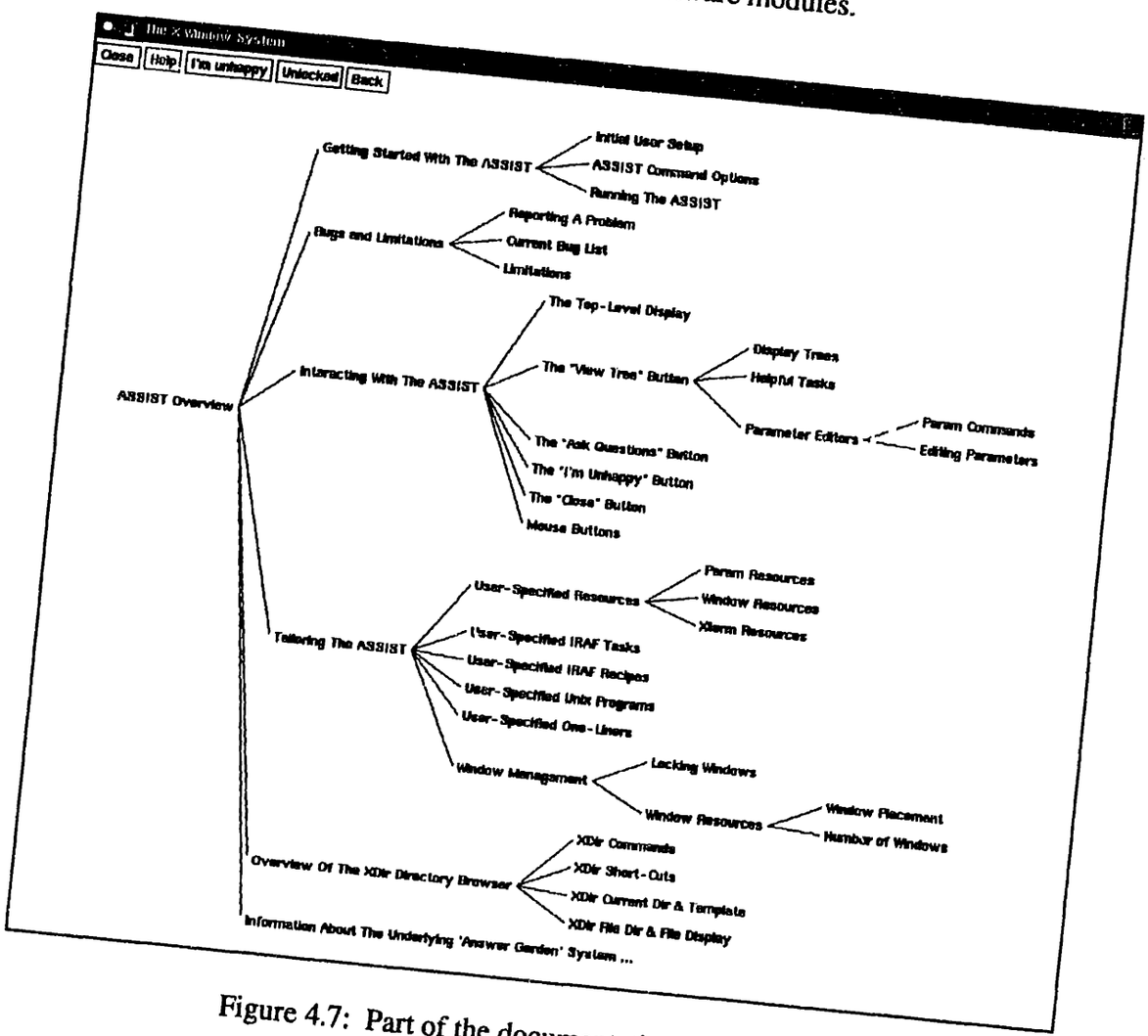

Figure 4.7: Part of the documentation in the ASSIST

Again, the information in the ASSIST grows over time. In the ASSIST, the authored material can include new documents, data, or data analysis software. The application also includes questions and answers about software or documentation, bug reports, feature requests, and help from other users (Figure 4.8). Some of these information objects are built dynamically.

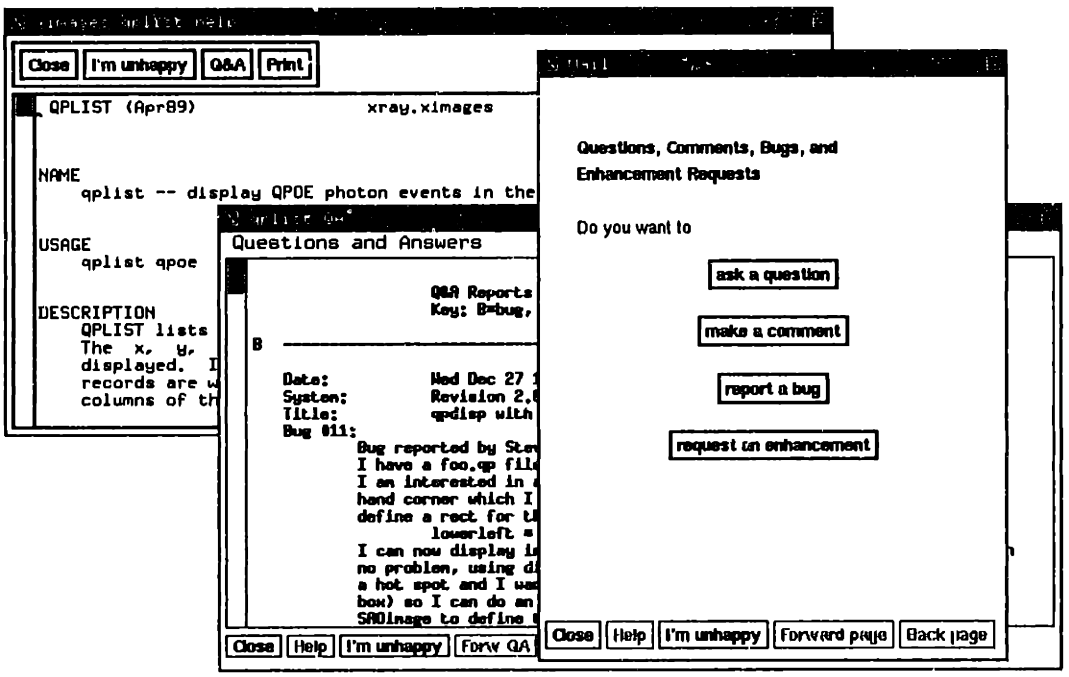

Figure 4.8: Various types of help available in the ASSIST

The most innovative memory component, however, is what the Smithsonian scientists call a "living cookbook." Astrophysicists can provide their data analysis recipes -- previously shared by pulling out their spiral notebooks -- for others or for their own personal use through the ASSIST application. Figure 4.9 shows an example of a user recipe as well as a tutorial. Unlike paper recipes, the links between the recipes and software modules are live; pressing a button in the recipe results in a program action. Thus, using ASSIST to follow the recipes allows the scientists to ignore the details of the data analysis software and to concentrate on the actual analysis: Having the software components and the recipes in the same memory application provides much more power

to the user. Sharing the recipes allows a common method of acquiring and retrieving data analysis methods, something they did not have before.

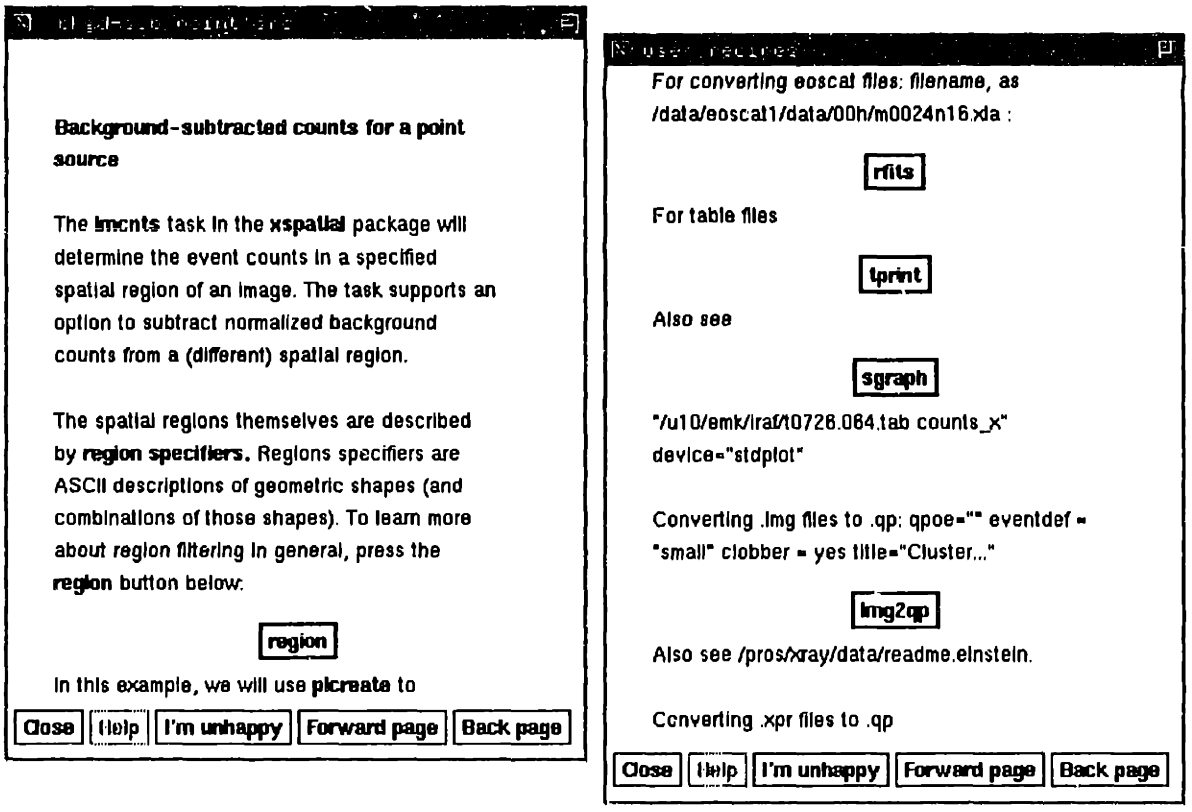

Figure 4.9: User recipes and tutorials in the ASSIST

4.3. Application 3: LiveDoc

Documentation is ever-present in organizations. From technical documentation to human resource manuals to "how-to" knowledge in functional groups, documentation is relied upon to facilitate and even establish organizational memory.

A standard problem with documentation is that it is inherently incomplete. The writers may not have understood the issues involved, or they may have reflected the official version of the issues. As one interviewed engineer said, "What is expertise? The usual things that tools [documentation and help systems] don't tell you and you have to run into and find answers makes you an expert."

Additionally, documentation is slow to change. When on paper, documentation is difficult to change; changes must be printed and interwoven with the existing manual set. When on CD-ROM, a new disk must be mastered and sent to users. What is more important, there are few feedback channels from users to documentation specialists.

The LiveDoc application is one mechanism for avoiding most of these problems. Consisting of a manual or any formal documentation, it uses "side notes" to mark and answer any additional issues or questions that may arise in use.

Figure 4.10 shows the beginning screen for the LiveDoc application. The user came to LiveDoc to examine a document to either browse it or to answer a specific question. This particular document is about LiveDoc itself. The user can search through the document in the usual manner, either by examination or by a full-text search. The icons in Figure 4.10 represent "side notes" and are used to indicate additional information. This additional information can include full discussions by organizational members, or questions and answers about the documentation or underlying process.

Figure 4.11 shows a side note. This particular side note answers some additional questions about this section of the document. Other possible side notes include additional questions and answers about underlying process, full discussions by organizational members about particular points of information, or additional information about the topic. For a reference document, the side notes may include tutorials to help beginners, such as new organizational members or people just beginning to learn a new skill.

All of the notes (as well as the original document) include the familiar "I'm unhappy" button to connect the user immediately with electronic mail (as shown in Figure 4.12). If the user has a question or problem with a particular section of the

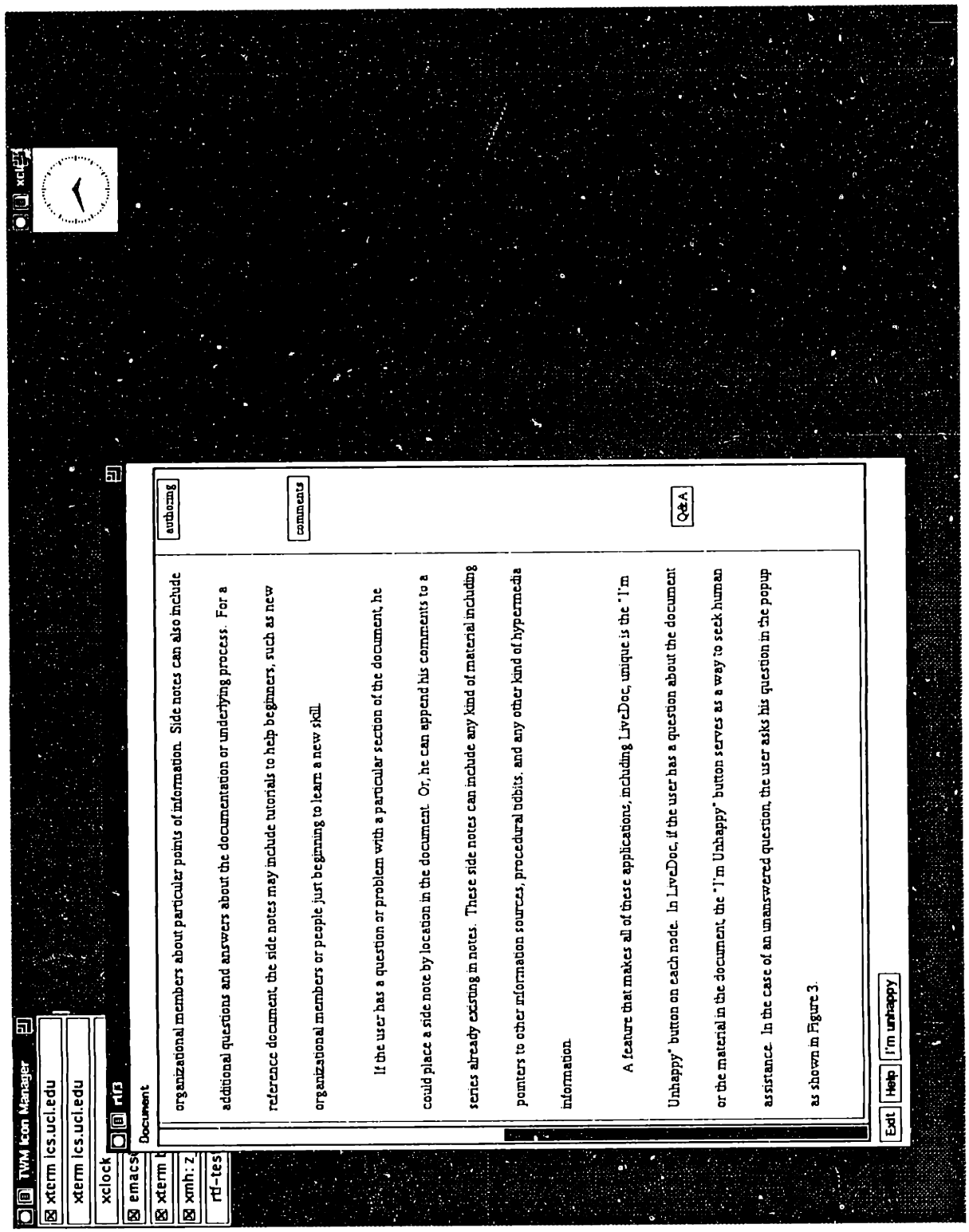

Figure 4.10: Initial screen in LiveDoc

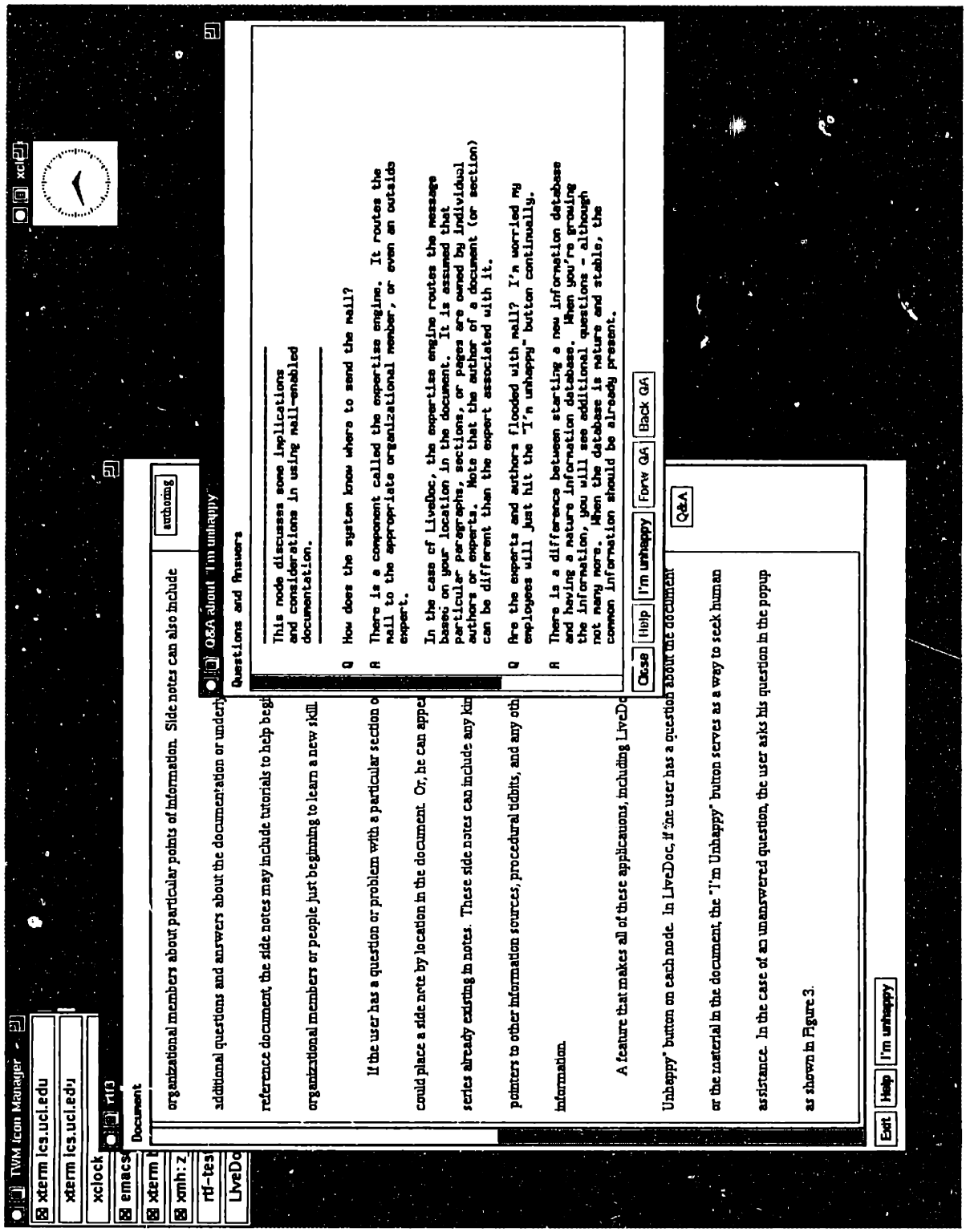

Figure 4.11: Auxiliary questions and answers in LiveDoc

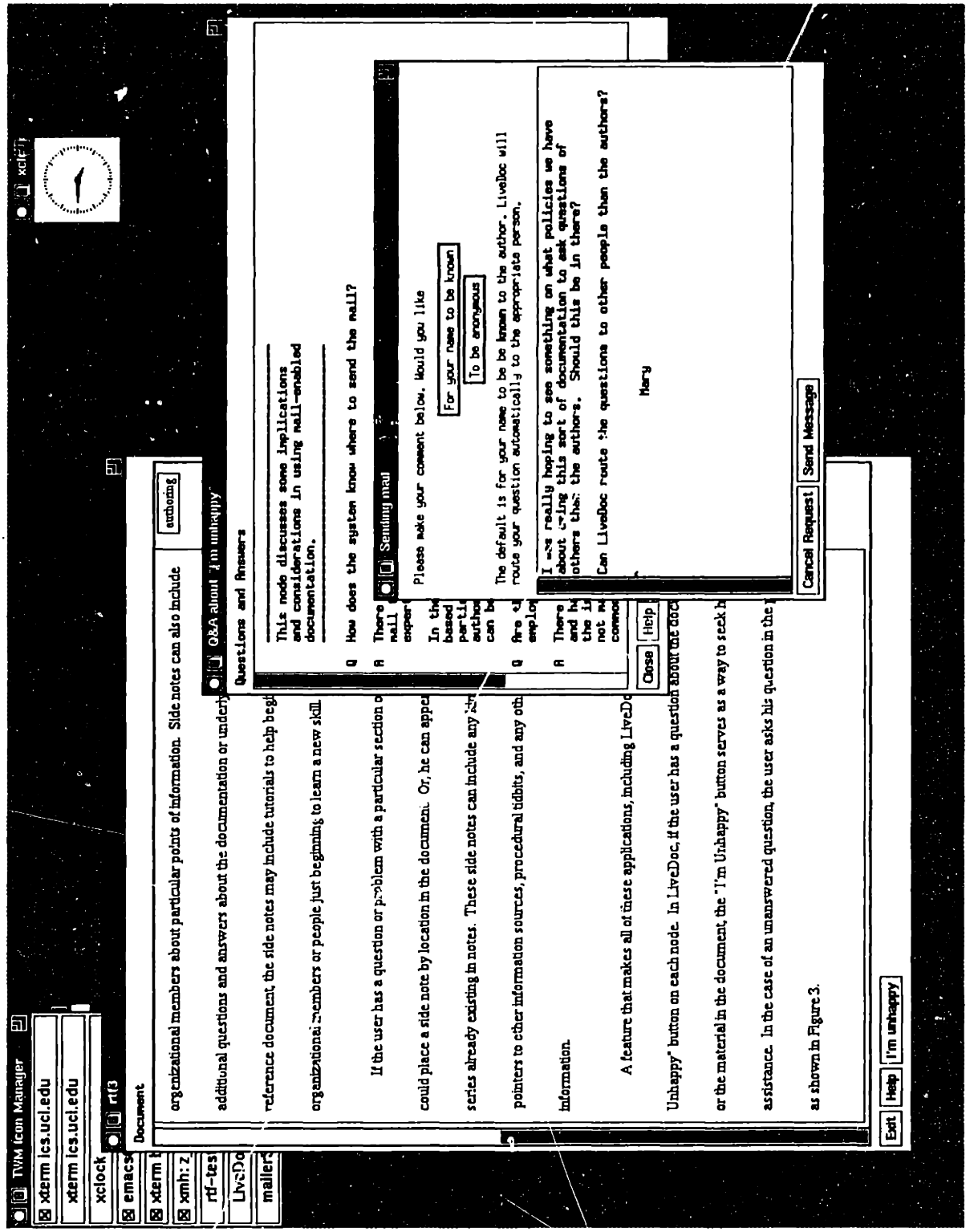

Figure 4.12: Asking a question in LiveDoc

document, the side note can be placed by location in the document. Or, the user's question or problem may be appended to a series already existing in notes.

If the underlying document were programming code, the side notes could be used for design rationales. Because of the iterative nature of LiveDoc, the design rationale could be constructed iteratively, as it was needed. For example, if a maintenance engineer did not understand internal comments, or felt that the comments were incorrect, he could ask the author of the system or other, more expert engineers.

The LiveDoc application is very similar to existing hypermedia systems; the major difference is the emphasis on iteratively adding new information and on allowing users to ask questions of the writers and experts.

Chapter 5: The Answer Garden Substrate (AGS)

The Answer Garden Substrate (AGS) is the underlying system beneath the Answer Garden, ASSIST, and LiveDoc.[1] It allows these, and similar, applications to be easily created.

This chapter provides an overview of AGS. Readers who wish merely a summary of the design assertions behind AGS and the Answer Garden application may safely skip to the next chapter. Those wanting an in-depth technical discussion of the Answer Garden Substrate should skip to *Appendix B: Answer Garden Substrate Overview* and *Appendix C: Answer Garden Substrate Internals*. The two appendices cover the same material as this chapter, but in greater technical detail.

5.1. Introduction

Why build an underlying system? Quite simply, it was to allow the easy construction of additional applications, some of which might deviate from the Answer Garden application slightly and some by substantial amounts. AGS is an attempt to generalize the Answer Garden functionality.

Historically, AGS was required when software engineers at the Harvard-Smithsonian Astrophysical Observatory wished to build the ASSIST application (described above). They wanted to add a substantial amount of functionality and

---

[1] As with any system, care must be taken to distinguish among the applications of a system and the system itself. In this chapter, I will refer to *applications* when I speak of programs that one can build using AGS. and *system* when I refer to AGS itself.

flexibility to what was then a system that could handle only the Answer Garden requirements. I have generalized almost all of the functionality in the original Answer Garden system so as to allow many different types of organizational memory applications. This generalization into a system-level framework constitutes the AGS system.

The goals, then, of AGS were to provide the functionality to:

- Create the Answer Garden application.
- Easily enhance or tailor existing applications (especially the Answer Garden application) by hooking in additional user interface objects, information retrieval objects, communication channels, message types, and file access methods.
- Create new organizational memory applications.

New applications are created by application programmers. While end-users and site managers can accomplish a substantial amount of tailoring, I have assumed that programmers would be required to build additional services and user interface objects.

However, much of the work for an existing application is done by information authors, whose responsibility it is to structure the information database and to author the actual information. For example, some Answer Garden author must write the information as well as formulate the diagnostic tree structure. A goal of AGS was to make most of the internal mechanisms transparent to the information author.

5.2. System overview

Broadly speaking, AGS can be divided functionally into four capabilities:

1. A set of object classes that manipulate and process information and/or display that information. One uses these object classes to build or modify an AGS information database; for example, to author some information for users.

   In AGS, these object classes are called *node types*, and the object instances are *nodes*. Nodes are generally visible and separately moveable on the user's screen, although this is not necessarily the case.

2. General support services for AGS applications. These include editing, screen management for the user, information retrieval mechanisms, and communications with organizational members.

   Additionally, AGS contains mechanisms to allow site managers or authors to customize existing applications.

3. Facilities for adding new object classes (or node types) to the system. New organizational memory applications may require new types of information or information display. Applications can add new node types if they wish to extend AGS's repertoire of information objects or display capabilities. Adding new node types requires a minimal understanding of AGS internals.

4. Facilities for adding or modifying the general support services for AGS applications. For full extensibility, AGS allows the base set of services to be modified or supplemented. Examples might include adding new information retrieval mechanisms or file access methods, as well as other, unforeseen services such as database or multimedia services.

5.3. General comments

AGS is an X Window System toolkit (Xt) application (Scheifler and Gettys 1987, Swick and Ackerman 1988). The Xt toolkit imposes a certain software architecture on its applications, and AGS is no exception. Specifically, all user interactions are handled through the Xt toolkit and its objects. These Xt objects, called *widgets*, form a separate object system from that in AGS (although some of AGS functionality is provided by new widgets).[2]

5.4. Pre-existing node types: authoring an information database

The node types, which in AGS are largely synonymous with both the user interface objects and the information objects, are the most critical components of AGS. An author of an information database uses the existing node types as his building blocks. The node types that currently come with AGS are the SBrowser, Grapher, QA-Node, Discussion, Code, and Ascii node-types. These were the base node types required to implement the Answer Garden application. Each has anywhere from a slightly to a radically different interface presentation.

5.4.1. SBrowsers (structured browsers)

SBrowsers can contain text, formatting commands, and system commands. There is a limited number of formatting commands implemented at this time; the capabilities

---

[2]A number of flavors of the Xt toolkit exist, including the Athena, Motif, and OpenLook varieties. AGS uses the Athena widget set (Xaw) currently since it is the most readily available. I will use Xt synonymously with Xaw in this chapter since these statements will also be true when (and if) AGS is ported to another Xt widget set such as Motif or OpenLook.

have been growing slowly as needed. Additionally, SBrowsers can contain links to any type of node.[3] The SBrowsers can thereby serve as menus, answers, tutorials, and so on.

5.4.2. Graphers

The Graphers display a tree of nodes. When the user selects a label in the tree, the Grapher will open the appropriate node.[4]

In the Answer Garden application, the Graphers form an alternative view to the tree specified through the SBrowsers. In fact, Graphers were written to provide an overview picture, replacing many SBrowsers. However, AGS applications need not conform to this model. The ASSIST application uses the Graphers for a reference view and the SBrowsers for a tutorial and problem solving view.

5.4.3. QA nodes

QAs (sometimes called QA-Nodes) are used for situations where many pieces of information are placed in the same node. The canonical example is a series of questions and answers.

The QA-Node was created to allow experts to throw many questions and answers together in one node without having to structure them. In the Answer Garden

---

[3]Note that while the user views a tree, the links actually form a full network. In a typical AGS application, the network will be a non-cyclical directed graph since the user is trying to narrow down a question or to find a specific item. However, there is nothing in AGS to prevent cycles. (An authoring tool is provided to check for cyclicality.)

[4]Both the SBrowsers and Graphers actually trigger a system command or set of commands on the mouse click. The usual action is to open the specified node, but other actions are also possible. For example, a database author might want to start an external tutorial program.

application, it was hoped that the diagnostic questions specified in the SBrowser would lead to atomic answers provided in other SBrowsers. However, a mechanism for allowing questions and answers to be quickly placed in an information database was required, and this requirement led to the QA node type. Other AGS applications use QA-Nodes for bug reports, tutorials, and enhancement requests.

5.4.4. Discussion nodes

Discussion nodes are simply compilations of Ascii data, such as a series of electronic mail messages. This node type allows the visual discrimination of nodes that contain a series of opinions. In the Answer Garden application, as well as others, users will want to be able to quickly distinguish situations where the expert could provide a definitive answer (or at least an approximation of one) from situations where there were many opinions about an answer. Examples of questions leading to such a node might be "What is the best way to deal with statistical outliers?" or "Which is better for coding an application, Lisp or C?"

5.4.5. Code nodes

Code nodes are essentially the same as Discussion nodes. The label at the top is slightly different to allow users to quickly determine that a node contains program code segments. It was required in the Answer Garden field study, where users wanted to distinguish between working code and answers. Users can cut and paste example code into their work from Code nodes.

5.4.6. Ascii nodes

Ascii nodes provide ready access to Ascii files. They were provided to allow the quick inclusion of pre-existing files or files that are to be edited on a frequent basis as part of normal operating procedures.

5.4.7. Other node types

New applications will often need to add to the existing set of node types to provide new types of information or information display. For example, the ASSIST application has four node types specific to itself, and the LiveDoc application has two.

One of the goals of AGS was to be able to quickly design and implement new node types. To do this, little about AGS needs to be understood. A node type writer merely writes fairly routine Xt toolkit code. A number of support functions exist within AGS, if they are desired, for accessing physical storage or other nodes, communications, and the like.

New node types that have been written include a database access node type, an outline node type, a journal entry node type, a parameter editor node type, and a help node type. The database access node type allows an AGS application to access remote relational databases.

5.4.8. The information in a node

The actual physical storage mechanism is not specified by AGS, and so no specific file format is required. Each physical storage mechanism may have its own format. In addition, each node type may have its own fields and formatting requirements.

Moreover, nodes need not even have an associated physical file. It is possible for a node to be *virtual*, and thus, built dynamically. An example of such a node might be a manual page built specifically upon request or a node built from a relational database query.

Virtual nodes not withstanding, AGS prefers to have a set of information for each node. Of particular interest are the expiration date, the associated expert or set of experts, and author's requested publicity level. The expiration date is used to determine whether

the information in the node has become obsolete. Information in organizations has a shelf-life, and this allows an authoring tool to tell the information author to check his information. The associated expert's name is used in the routing of communications.

The final field of interest is a field used to determine whether the name of the author should be published on the node. Users identify the information's quality in part by the author of the information. However, some authors do not wish to be identified by name for fear that they will be inundated with queries.[5] On the other hand, some authors, such as consultants, want to be identified. AGS provides the capability for authors to identify whether they want to be known by name, known by their organization, or not known at all.

Each node, then, is semi-structured, having some structured fields as well as text or other unstructured information (Malone et al. 1988). New node types can have additional header fields, permitting the nodes to contain additional structured data, such as weighted keywords, for information retrieval or data manipulation purposes.

5.5. AGS Services

As mentioned, the node types are the heart of AGS. But, in order to make an AGS application work effectively as an organizational memory system, it was necessary to include other facilities. AGS is really the confederation of nine Services.

The Services are:

---

[5]Users often wish to have the expert identified so they can evaluate the quality of the information. On the other hand, the first time Answer Garden was ever shown to X experts (who were to participate in the Answer Garden field study), all quickly requested that their name be removed from the node. On further investigation, I found that this could be resolved by removing the electronic mail addresses of the experts.

1. A set of UI presentation objects. These are the node types described above, and they present information to the user.

2. A Communication Service. An organizational memory system needs to handle outgoing electronic mail or other communication mechanisms. The Communication Service also includes the capability, called the *expertise engine*, to route the communications to the appropriate organizational member.[6]

   The Communication Service handles the routing transparently to the user and to the information author. Outgoing messages are semi-structured, containing data about the user, the type of question, the user's position in the database (including a history list of where the user has been in the last 15 minutes), and other information useful to the expert answering the question.

   The current expertise engine uses a fairly straightforward heuristic. There can be experts for each node, as well as local experts (such as a help desk) and global, backup experts.

3. An Information Retrieval Service. There is considerable controversy in the information retrieval (IR) community as to the best information retrieval engine since each engine has slightly different retrieval properties. AGS is not didactic about what engine must be used. Instead, AGS allows a wide variety of

---

[6]The expertise engine has also been called an expert-locator in Lesk 1991. Other approaches to finding an appropriate expert are covered in Streeter and Lochbaum 1988, Kraut and Streeter 1990, and Kass and Stadnyk 1991.

engines to be included in an application. Several IR engines are provided.

4. A Node Service. Because AGS uses a Node Service, information authors need to use only the node name (or a direct manipulation surrogate) in their text. The link to physical storage is handled automatically.

5. A set of File Services. Each File Service handles one type of physical access, such as flat files, an SQL database, or a distributed infobase server. This allows the information author to not worry about physical storage requirements and constraints.

6. Authoring Service. AGS provides a base level of authoring tools, including a built-in editor for node level information as well as the actual information content. It also provides a small set of independent authoring tools. In general, users do not have access to the Editing Service and tools, although there is nothing in AGS inherently preventing them from having such access.

These Services are essential to an organizational memory system. The inclusion of the Communication Service ties the system into the communications network of the organization. While the user interface objects can present a direct manipulation interface to the information in the database, the inclusion of the Information Retrieval Service provides additional capability to retrieve from large amounts of information. The other three Services provide the necessary, but not very glamorous, book-keeping and information production capabilities in such an organizational memory system.

In addition, there is a set of minor services available anywhere within AGS:

7. Window management. AGS can put up many windows, too many for most users. These windows can be managed through the standard X mechanisms, such as the user's window manager, or they may be handled within AGS through "window stacks."

8. Function Service. This Service allows nodes to use operating system features, including starting or communicating with other applications. In addition, it provides a command set that can be used in any node to compute and assign values to system variables, parse new commands, and the like. These can be used by the information author as well as the node type writer.

9. Blackboard Service. AGS provides a two-level blackboard architecture so that information authors can maintain internal memory between AGS components as well as external programs or agents. For example, an author can ask the user to provide information about his work environment for use in dynamically configuring later answers.

In addition, AGS contains a relatively small number of "core" procedures that initialize and drive the various Services, keep user statistics, and put nodes up on the screen.

5.6. Extending AGS

AGS can be extended in a number of ways. One may wish to add new node types to allow the users to view information in a new way or to access new types of information. For example, the ASSIST application added a number of node types to facilitate the use of the data analysis packages inside the application. In this way, AGS can handle heterogeneous information that was not anticipated by the system designer.

Additionally, one can extend AGS's existing set of Services. One might do this to add a new file access mechanism, a new communications mechanism, or a new information retrieval engine. For example, application writers can easily make new forms for outgoing communications, change the mail characteristics at any node, or even add new communication mechanisms by using the existing Communication Service building blocks.

Finally, an application writer can add to the list of AGS Services. One of the requirements for any organizational memory platform is the extensibility to provide new forms of organizational support as they become necessary. Therefore, the ability to add new Services to AGS was of considerable importance and is fully supported. AGS allows application programmers to add in new Services; examples include multimedia or workflow capabilities.

5.7. Summary

The Answer Garden Substrate (AGS) is the underlying system beneath the Answer Garden, ASSIST, and other applications. It provides a set of functionality to allow these, and similar, applications. To do so, it provides many types of information objects as well as support for a linked network of information, organizational communications, information retrieval mechanisms, and file access. The important characteristics of the Answer Garden Substrate include heterogeneous information objects and easy extensibility, as well as support for a number of technologies and organizational services.

Chapter 6: Design Assertions

Behind the Answer Garden Substrate and the three representative applications, there are a number of design assertions. I concentrate largely on the assertions governing the design of the Answer Garden application.

Assertion 1: An effective organizational memory system could combine information retrieval and communication capabilities.

The major conclusion in Chapter 1 was that one possibility for an effective organizational memory system could contain ways to retrieve documents or other information objects and to access individuals within the organization. AGS provides mechanisms for doing both of these with its ability to include heterogeneous information objects, to retrieve documents, and to send and route messages to organizational members. By including communication capabilities in AGS, people can serve as information repositories as much as can text documents, SQL databases or the like.

Answer Garden makes heavy use of these communication capabilities. To connect to the social network in a useful way, a routing system is built into Answer Garden. Thus, Answer Garden does not broadcast requests for information to all members of the organization. Instead, the proper individuals are selected according to some criteria. In the current implementation, the criterion is expertise, and the system functions as an expert locator system similar to that in Who Knows (Streeter and Lochbaum 1988) and Who Do I Tell (Kass and Stadnyk 1991). One could imagine other criteria such as availability, geographical location, or reporting structure.

Assertion 2: An effective organizational memory system should provide incentives for use.

I also pointed out in Chapter 1 that organizational memory is most beneficial in work processes that offer an immediate effect for the organization. Since an organization has limited resources, it will be most likely to employ organizational memory when the effects are immediate and the cost is low. AGS offers a number of mechanisms that are useful here, but it is the applications that offer policies effecting immediacy and cost.

Answer Garden attempts to offer a number of important incentives towards its use. First, the purpose is aimed at problem-solving in an organization, something of immediate impact for many organizational members. For users, offering solutions to commonly asked questions provides the potential of immediate relief to pressing problems. For the experts (or others answering questions), it offers the possibility of immediate relief to their work load or, at least, reducing repetition in their workload.

Second, Answer Garden attempts to reduce the cost of providing the information. Users and experts exchange electronic mail as they normally do. As they do so, the corpus of information and its classification structure grows iteratively. This emphasis on iterative construction has several important repercussions:

- Answer Garden allows user feedback for the correction of mistakes. With standard information systems, there is little ability for the author or system maintainers to get feedback from users, and there is little ability for users to say where there were problems. The feedback mechanisms in Answer Garden allow expert-authors to correct mistakes both in content as well as in indexing and navigation. One such mechanism is that questions are mailed with a semi-structured header including a full user

history for the last 15 minutes. Thus the expert can determine whether the diagnostic questions or other indexing schemes need to be modified or supplemented.

- More importantly, the design of Answer Garden allows for the *production of information on demand.* As was argued in Chapter 3, the standard information retrieval model emphasizes the retrieval of a pre-existing collection of information. If the collection is incomplete, the user has little or no recourse. In Answer Garden, on the other hand, the user can ask for the expert-author to produce information that is missing. (It is likely that the effort requested must be kept to a minimum in order for the expert to acquiesce.) Thus, Answer Garden assumes that the information database will be iteratively designed and built.

- The Answer Garden information database is built up where users have questions. Answer Garden thus attempts to satisfy users, preventing the excess effort of authoring and gathering information that would not be valuable to users.

Both the ASSIST and LiveDoc applications also carry this emphasis on immediacy and low cost. The ASSIST application will be used in the everyday context of data analysis. For the scientists using the system, it must be relatively easy to share analysis recipes and ask for information. The users of LiveDoc use the application as they would normal documentation. Both systems offer the ability to correct problems and mistakes and to grow the information corpus over time.

Assertion 3: An effective organizational memory system, of the sort outlined in assertion 1, should reduce status implications and the need for reciprocity in information seeking.

The final assertion is based on the discussion in Chapter 2. I argued that it would be useful to augment the expertise network by reducing the status implications in asking questions. This would be especially true when asking common questions, the type solicited in the Answer Garden. These common questions might be viewed as "stupid" by the information seeker, and therefore, not asked of the expert. However, with Answer Garden "stupid" questions are desired; they build up the database.

One way to ameliorate the status implications is to have users ask their questions anonymously. In Answer Garden, questions can be sent to experts anonymously. This is reinforced when the users are notified that their question has been forwarded to an expert and when they receive a reply. A potential side-effect of this policy, however, is the possibility that some users might abuse the system, either by flooding the system with questions or by not examining the database of answers first.

I also argued there is a professional norm of reciprocity in information providing, especially for people with technical career-anchors. Information seekers often feel as though they must answer questions for the information provider if the relationship is to be comfortable. This norm must be curtailed to have people comfortably ask questions of experts through Answer Garden since the relationship is changed by the nature of the system. In Answer Garden, the experts gain not by receiving different information in exchange; they gain by removing the burden of answering similar questions over and over. The users of Answer Garden must be reassured that these interactions are desired. On the other hand, such asking is not free, so asking questions without looking for a pre-existing answer needs to be kept a taboo. Fortunately, this taboo already exists; it is the standard norm (in technical organizations) of being informed before asking.

Chapter 7: Answer Garden in the Field

Answer Garden offers an alluring application, providing the possibility of capturing portions of an organization's memory at a relatively low cost to that organization. But, would it work in practice? The previous chapters have argued that it should. I undertook a field study of its use to determine whether, in fact, it would.

This chapter proceeds in three major sections. At first, I will discuss my research method including additional assumptions about Answer Garden's design in the study. Next, I will describe the research sites and data collection for the field study. The analysis section proceeds in several parts: an analysis of system usage, user evaluations, authoring observations, and then technical findings.

7.1. Research method

In order to determine how Answer Garden was used in an organizational context, I undertook an exploratory field study using multiple methods and multiple sites. Users' experiences with Answer Garden with the sample information database about the X Window System were extensively studied in two sites over three months. Four additional sites also provided supplemental data. Data for the two main sites were collected through initial interviews, semi-structured critical-incident interviews, usage logs, electronic mail exchanges, and direct observation.

7.1.1. Study group

I chose to study software engineers using the X Window System, a user interface system for workstation-class computers (Scheifler and Gettys 1987, Swick and Ackerman 1988). I did this for a variety of reasons:

- Software engineering is of special interest for an examination of expertise and information seeking because there is no body of commonly accepted knowledge (aside from low-level data abstractions and algorithms which all programmers are expected to know). Furthermore, the software world is undergoing constant revolution, preventing an individual engineer from acquiring lasting expertise.

- User interface work tends to be iterative in nature, resulting in substantial problem solving after the initial design phase. For example, the design of a user interface is subject to continuous testing and refinement. It cannot be easily specified in advance of testing. In addition, user interface engineers tend to be isolated within groups of software engineers doing other tasks. As such, they may need more assistance than other software engineers.

- X, like other windowing systems, is quite complicated, requiring either substantial technical expertise or access to expert information. Many of the quirks and workarounds are not well documented. Furthermore, users of the X Window System are computer-sophisticated, and the X world communicates, in large part, through electronic mail.

Software engineers using the X Window System, then, were ideal candidates for information seeking through electronic means such as Answer Garden. Additionally, building an information database of commonly asked questions and answers about the X Window System was helpful for the general X world.

7.1.2. Field study assumptions

Chapter 6 reviewed the design assertions behind Answer Garden. There were some additional assumptions and simplifications in the field study:

- It should be noted that the normal computer-human interaction standards of usability were not the ultimate goal of Answer Garden. (Or the notion of usability needed to be expanded.) Usability was only one necessary condition for Answer Garden's use. Answer Garden, of course, needed to be usable quickly by any user. (My goal was the video-game ideal of the self-evident application, where one has only 30 seconds to convince and train the user to operate the program.[1]) In addition to being usable at an individual level, however, the application needed to be usable as a social tool if it were to be anything but a standard hypertext reference system. Therefore, the sufficient condition for its usability was its use in place or in its social context.

  Moreover, the field study assumed the users were a computer-sophisticated, but domain-naive group. That is, the users of Answer Garden with a specific information database would know how to operate the computing system in general, but would not

---

[1]To this end, Answer Garden has few control mechanisms and all of these are visible. Some additional "bells and whistles", such as additional information retrieval engines, are available from the top level node through a pop-up menu. Such bells and whistles are not necessary to run the program at first.

Instead of running a usability test with end-users, I had a number of CHI experts run the system and offer critiques. This method of using CHI experts to evaluate usability has been shown to be almost as effective as large-scale and costly user tests (Nielsen 1992).

know everything, if anything, about the domain of inquiry. In this sense, Answer Garden was designed to be used as an infrequent tool, something to be consulted on an infrequent basis. Usability, then, would result from one of two things. Either the user would achieve extra efficiency by using the same tool for many knowledge domains, or the user would gain an efficiency from having an application that required relatively little training but offered paybacks in his information seeking. Answer Garden, therefore, changed the normal roles of naive and expert users, placing them in a more everyday context of use where users may be expert at some activities and naive in others.

- Several simplifications were made in the field study, and these constitute additional assumptions within the study itself. An implicit assumption was that there were separate groups of experts and of information seekers. This is clearly not the case in organizational life, where people are arrayed along many continua of expertise over many subject domains. Those with little expertise in one area go to those with somewhat greater expertise, and so forth. (This was one of the premises behind the discussion of expertise networks in chapter 1.) However, to simplify the examination of Answer Garden in use, the field study assumed separate groups of experts and information seekers. In other settings or other information databases, one might have some people be both experts in one area and information seekers in others. Or, the questions could go to a general distribution list where everyone served in both roles.

Additional simplifications in the field study were made about the type of information to be captured in the information database. An implicit assumption behind Answer Garden is that the information within an information database should be easily classifiable and easily broken into node-sized "chunks." While complex subjects, or ones with highly interdependent answers, are possible to capture in an Answer Garden, the application is more amenable to answers that can be classified along a set of easily decomposed diagnostic questions. In addition, the information seeking characteristics detailed in chapter 2 assume that there is some answer, at least one that is good enough for the purposes of solving the problem. A technical domain, such as the X Window System, is well-suited to these characteristics. It is, of course, possible to have a domain that is merely a set of opinions. An example of such a domain might be a database about wines or software design practices. There were some nodes containing opinions in the field version, but they were few.

- A corollary to this is that I modeled the authoring process as one of building a set of correct answers. To do this, I designed the authoring process assuming a set of experts, each of whom had particular areas of expertise. (These areas might overlap, however.) Each expert, or set of experts, was to be responsible for subtrees in the Answer Garden network of questions and answers. This turned out not to be the correct model for the experts in the field study, and fortunately, Answer Garden was flexible enough to handle the change.

7.1.3. Research system

The field study used Answer Garden version 0.76 and version 0.79, with version 0.79 being in place by the fourth week of the field study. Version 0.79 was the first release of the Answer Garden Substrate, described earlier in this thesis. (Version 0.76 lacked many of the capabilities necessary to building other applications.) However, to the end-user, there was little difference between the two. The only noticeable difference in the user's view was that version 0.79 permitted copies of the questions to be sent to the user, a feature requested by several of the early participants.

Neither of these versions included information retrieval engines (other than the standard, naive retrieval engine) or window stacks (to tidy up the user's screen).

The version 0.79 that was released to the field sites was essentially the same as the version distributed on the MIT X Consortium contributed software tape. This "contrib" version is readily available throughout the world. Because it was designed to be readily available, AGS version 0.79 is quite robust. Only a handful of minor bugs have been reported.[2]

---

[2]The only known important bug remaining in the field version was its inability to correctly adjust for the screen mechanics of decwm, Digital Equipment Corporation's window manager for its workstations. The decwm window manager pre-dated the conventions for window manger and client interactions, and its mechanics were idiosyncratic. (It is no longer used.) Unfortunately, this was the window manager used at one of the field sites. The effect was the inability to recall an opened node to the top of all other windows, and this resulted in users' multiply restarting some sessions. This was mentioned by users only to explain why there were multiple sessions close together in time; no further complaints were voiced. (This problem with the data was corrected in the usage logs before any analysis.)

There has recently been reported a similar bug with the Sun OpenWindows window managers (olwm). It was not reported during the field study (or the first six months of 1992); this may not have been a problem during the field study period.

One piece of software was not completed for the field study. I intended to include a back-end Question-Answer Tracking System (QA Tracker) to monitor the status of questions and their answers and to implement anonymity. Although a partial prototype was built using the Oval system (Lai, Malone, and Yu 1988), the QA Tracker was hand-simulated in the field study. This did not affect anything that the end-users saw during the field study.

7.1.4. Participants and research sites

Answer Garden was advertised through broadcasts on the Usenet bulletin boards and through personal contacts. Two field sites provided almost all of the data in the field study. These two sites were a research group at MIT and a class in the Harvard extension program. The additional field sites, which mostly served to provide confirmation for observations in the two major sites, were a software group at a federally-funded laboratory, a product team in a software company, an advanced development group in a large computer company, and a support group in another computer company.

7.1.4.A. Harvard software class

The Harvard site was a Harvard extension class in two-dimensional graphics programming using the X Window System. I decided to use the Harvard class, with its 44 students and 3 instructors, since its students were largely working software engineers. As such, they used Answer Garden in their workplaces, potentially substituting it for asking questions and seeking information from organizational colleagues.

All other reports of Answer Garden's use in the field study were that it was robust in its operation. No abortive sessions, core dumps, or other problems were reported or seen in the usage logs.

Training for class members was minimal. I presented Answer Garden in a ten-minute talk and gave class members a seven-page hand-out on starting and running Answer Garden. I also ran Answer Garden for a half-dozen class members after my presentation. The teachers of the class, additionally, emphasized during several early classes that Answer Garden would be a helpful tool to answer their questions during programming assignments.

7.1.4.B. MIT research group

The MIT research group, with 11 members, was engaged in sponsored research, and their work flow appeared to be similar to that in a software R&D company. Training was again minimal. I sent out an electronic mail message detailing how to start up and use Answer Garden. I also demonstrated Answer Garden to several key participants in the group. The leader of the group reported that he encouraged his staff members to use Answer Garden in one staff meeting and in several private meetings. The MIT group has existed for over five years, and included research staff, graduate students, and undergraduates. Two of the graduate students and three of the undergraduates were new to the group; the others had been in the group for a year or more.

7.1.4.C. Other sites

The additional four sites used Answer Garden sporadically, and therefore, provided minimal usage data. Two of the sites were local, and therefore, I was able to obtain interview data. The other two of the other sites were handled entirely remotely, via electronic mail and telephone.

7.1.4.D. Experts

There was, additionally, another set of participants in the field study. They were the experts that answered questions.

I wished to provide the study participants with authoritative information, so that they would not need to judge whether it was accurate or not. In order to do so, I tried to collect a group whose level of expertise would avoid any issues around authoritativeness of the information in the field study's information database. These experts were collected through advertisements on the Usenet bulletin boards and through personal contacts.[3] All of the experts had more than three years of experience with the X Window System, the Open Software Foundation's Motif (which is based on X), or Unix International's OpenLook (which is also based on X). Two colleagues and I served as the first-line of answerers. Other experts included four staff members of the MIT X Consortium, including those responsible for the design and maintenance of the Xlib, Xt, and Athena widget set layers of X. Three others were consultants on Motif and OpenLook. Other experts volunteered, but were not needed.

7.1.5. Data collection procedures

Six data collection procedures were used in the field study:

7.1.5.A. Initial questionnaires

First, all participants were asked to fill out an initial questionnaire that included questions about their current job, education, work experience, and X Window System experience. It also contained a question to determine whether there were others of whom they could ask X questions.

---

[3]For anyone considering a similar study, I should note that having a group of experts and a group of user sites made the arrangements for the field study especially difficult. It was necessary to get both groups in tandem before the study could be run. It took about six months to arrange that both the user sites and the experts would be available at the same time. This, of course, would have been easier in an intra-organizational study, but that would have required a substantially greater level of organizational commitment to the study than was received.

7.1.5.B. Usage logs

Second, usage data were collected at the "mouse-stroke" level for every session.[4] The usage data were written by the system into one statistics file per site. This file was invisible to users of the system This usage data was to be harvested once a week (or more) and sent to MIT for analysis. A utility program was provided to automatically mail out the old statistics file and start a new one. During the study period, the statistics files were harvested once a week at best. Once the statistics file was received at MIT, each session was tagged.

7.1.5.C. Critical incident interviews

The third method of data collection was critical incident interviews conducted by telephone with a sample of the Answer Garden users. Most critical-incident interviews ran between 15 and 20 minutes in duration, incluing time spent obtaining qualitative data concerning the interviewee's responses. For example, users often provided (and were encouraged to provide) additional information on why they rated the system as they did.[5] Because these critical incident interviews were not systematically randomized, only the qualitative data from the 49 critical incident interviews are being analyzed here.

---

[4]All participants in the study were notified that their usage would be monitored and that an interviewer would occasionally call to ask them questions about their use of the Answer Garden. All of the participants signed a form stating that they understood that their use of the application was voluntary and that they could terminate their use at any point in the study. Moreover, users were reminded that their usage was being monitored for research purposes when they started the application.

[5]Users were reassured in the interview that negative responses were as valued (or more so) than positive ones.

7.1.5.D. Final questionnaires

Final questionnaires, based on Davis' (1989) user satisfaction measure, were distributed to all potential participants. The questionnaires also asked for qualitative responses about user satisfaction.

The return rate for these questionnaires was less than 30%. Because of the low return rate, the user satisfaction measures were not analyzed. However, the qualitative responses were included in the analysis below.

7.1.5.E. Additional qualitative data

In addition to the qualitative data obtained during the critical incident interviews, there were qualitative data from electronic mail discussions, research notebooks, and interviews with participants. All questions and answers sent during the field study were also logged.

7.1.6. Data sets analyzed

The data collection resulted in three data sets.

7.1.6.A. Session data set

The first set was the session data set, and it included data for each Answer Garden session from the two sites.

This data set included 15 variables. The data set included the session number, date of the session, the user's site, and the user id. Two variables measured the length of the session, the number of nodes opened by the user and the elapsed time spent in the Answer Garden session. Three variables measured mail usage, the number of times the user opened a mail window during the session, the number of mail messages sent by the

user, and the time spent from opening the mail window until the mail was sent or the mail window was canceled. All of these were determined from the usage logs.[6]

Several additional variables were calculated in the session data set. It was easily possible to determine whether this was the user's first session. In much of the analysis, I used a dichotomous variable for mail usage to indicate whether the user had used the mail system in any of his sessions. An additional variable measured the time spent actually examining nodes; it was calculated as the elapsed time spent in the session less the time spent sending or composing mail. The time spent per node was calculated as the time spent looking at nodes divided by the number of nodes in the session.[7] Finally, several transformed variables were created to achieve normal distributions.[8]

---

[6]There were three changes I made in the usage logs in building this data set. First, some users had problems with their window managers and the number of Answer Garden windows, and so restarted the application. For the purposes of this study, multiple sessions that occurred in close temporal proximity were collapsed into one session. Second, some users, as will be discussed below, left Answer Garden running over the course of days. If an extended period had elapsed since the last session, and all Answer Garden nodes except the top level nodes had been closed previously, the usage log was broken into multiple sessions. Third, some users closed their Answer Garden nodes after extended periods of inactivity, inflating the session times. Their closings of the final nodes, when they occurred after an extended period, were dropped. Several sessions still occurred over many hours, without clear evidence of being multiple sessions, and these were treated as outliers.

[7]An attempt was made to transform this variable (PERNODE) into a normally distributed variable. While the transformation SQRT(LOG(PERNODE*10)^2)) looked as though it provided a suitable distribution when examining the normal distribution plot, the variable failed the Kolmogorov-Smirnov one-sample test. Therefore, non-parametric tests were used with PERNODE.

[8]The logarithmic transformation of the elapsed time (LOGELAP) was found to be normally distributed through examination of a normal probability plot and through a Kolmogorov-Smirnov one-sample test. The Lilliefors probability was .43, indicating that the distribution of LOGELAP did not differ statistically significantly from the Normal distribution.

7.1.6.B. User data set

The user data set included data from the initial questionnaires as well as summary statistics from the session data set. The summary variables included the total number of nodes opened, total time spent in Answer Garden, the total number of times a mail window was opened, the total number of times mail was sent, the total amount of time the user spent in the mail system, the number of days between the first and the last session, and the total number of sessions.

The questionnaire collected a number of demographic variables. The questionnaire measured the number of months the participant had been at their current company, the number of months the participant had been in their current job position, and the number of months the participant had been employed. These ranged from 0 months to 25 years. The questionnaire also measured the participant's educational achievement, and this ranged from high school through the Ph.D. The sum of the months of employment and educational achievement was used as a surrogate for the person's age. Another variable measured their self-reported vocational title, but this variable did not vary to any appreciable extent.

Several questions measured the participant's information network and needs. A dichotomous variable measured whether the participant felt he had people to ask about X

The logarithmic transformation of the time spent examining nodes (LOGLOOKN) was also found to be normally distributed through examination of a normal probability plot and through a Kolmogorov-Smirnov one-sample test. The Lilliefors probability was .66, indicating that the distribution of LOGLOOKN did not differ statistically significantly from the Normal distribution.

An attempt was made to similarly transform the number of nodes (NUMNODES) and failed. Therefore, only non-parametric tests were possible with NUMNODES.

at work. This was also measured by a categorical variable built from questions on the number of people that could be asked.[9] A variable measuring the participant's experience with X collapsed answers about the length of their general usage and programming in an X environment. There were also variables measuring whether the user self-reported using the Usenet bulletin boards or an electronic mail distribution list about X and measuring the participant's intention to work at his office.

7.1.6.C. Qualitative data

The qualitative data included, as mentioned above, data from the critical incident interviews, final interviews with the experts, and final user questionnaires, as well as observational data. The comments from the critical incident interviews were grouped in several different ways, depending on the needs of the analyses below, in a manner standard to qualitative studies (Filstead 1970; Kirk and Miller 1986; Schwartz and Jacobs 1979; Strauss 1987; Van Maanen, Dabbs, and Faulkner 1982). For example, they were grouped by whether the participant evaluated his session negatively or positively; subsequent analysis looked for patterns in these groupings of evaluations. The critical incident interviews were also categorized by whether the participants gave status indicators in their interviews, whether they spoke of anonymity, the order in which they used Answer Garden, and so forth. The interviews with the experts were categorized, as will be discussed below, by the problems and issues they encountered. Additionally, experts were asked follow-up questions specifically about several issues uncovered in the field study. The final user questionnaires were categorized according to the type of usage and by the problems mentioned. The observational data was used to supplement and

---

[9]The questionnaire asked the number of people that could be asked in person, asked in person within a short walk, asked in person within a long walk, and asked through electronic mail or telephone. Because of the small number of people answering the questionnaire, data for this variable were collapsed.

bolster the qualitative and quantitative data. In general, a concerted effort was made to use the qualitative and quantitative data together (Jick 1979).

7.2. Patterns of Use

> "In this section we wish to present some trends and observations of the uses, strengths, and weaknesses of this ... tool. ...In this section we have tried to be as candid as possible about the weaknesses and research problems.... We hope this candor does not create an overly negative impression about what we feel is a very positive research effort." (Conklin and Begeman 1988, p. 147)

The statistical analysis of the usage data showed that:

- Answer Garden was used by the study participants as an occasional tool. Nearly sixty percent of the potential participants in the study used Answer Garden; however, most users used it infrequently during the study period. This was as expected.

- Usage between the two major sites differed, perhaps because of differing natures of their work. The two sites showed different usage patterns by week and in their pattern of adoption, but no other differences in their use of the system were apparent.

- Users could be categorized into three groups, "tire-kickers," "intermittent users," and "heavy users." The heavy users showed a markedly different pattern of usage, with indications that they had adopted Answer Garden into their normal work patterns. The intermittent users used Answer Garden a small number of times, but they used it over periods longer than a month.

- Answer Garden sessions were short. Seventy percent of the sessions were less than 10 minutes long, and the average time spent on a node was only 1.2 minutes (s.d., 2.5 minutes). There

was no statistically significant difference between first sessions and subsequent sessions, perhaps indicating that users did not have a training overhead for the system.

Each of these findings are discussed in detail below.

7.2.1. Answer Garden usage

During the study, Answer Garden was used 194 times by 35 users in the two major sites (mean number of uses per user = 5.54, s.d.= 7.01). Since Answer Garden was designed to be a system that would be used intermittently, I expected usage would occur only when the user had a difficult question, perhaps twice a week or even once a month for consistent users. In fact, this is what was observed. The maximum that any user used Answer Garden was 28 times over three months.

This pattern of usage did not appear to be from lack of interest. Only five of the users appear to have been just looking from curiosity; the others self-reported serious usage. Moreover, these 35 users were 59% of the 59 potential users [sic]; additionally, several potential participants reported being unable to connect to the Harvard Answer Garden for technical reasons.

7.2.2. Differences in usage patterns between sites

Answer Garden's use was not uniform across the test period. As Figure 7.1 shows, the aggregate usage showed large usage initially as users experimented with the application. Usage dropped in week 11 since this was the Thanksgiving holiday. Additionally, usage dropped after week 14 when the MIT group ended their use with the end of MIT's semester. The Harvard class continued with Answer Garden because their semester continued until the middle of January, and the course instructors felt it unfair to remove it during the final weeks.

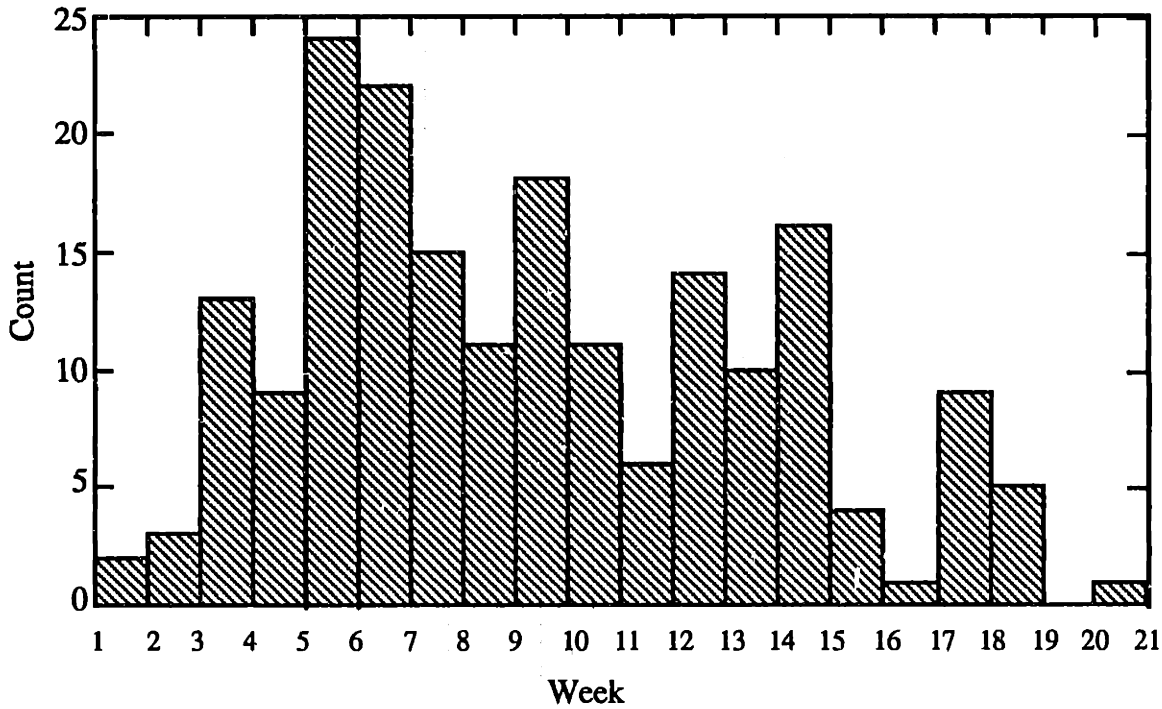

Figure 7.1: Sessions by week (all users)

This usage pattern was the result of both groups' usage aggregated, and these two groups, in fact, showed different usage patterns. The MIT group (see Figure 7.2) started two weeks earlier than the Harvard group and displayed gradual increase in the number of sessions per week through week 6. The MIT group then fell into a steady usage of two to four sessions per week, with use from nine of eleven potential users. The spike during week 8 suggests that usage can increase by a factor of 100% during "normal" periods.

Interviews with these MIT participants revealed no systematic pattern since usage was driven by a combination of local deadlines and steady-state work. This is the same as the demands on experts' time now; people do not ask questions of experts on a specific schedule. However, it increases the difficulty of scheduling experts for answering questions through Answer Garden, and suggests that a workflow component is required for similar systems.

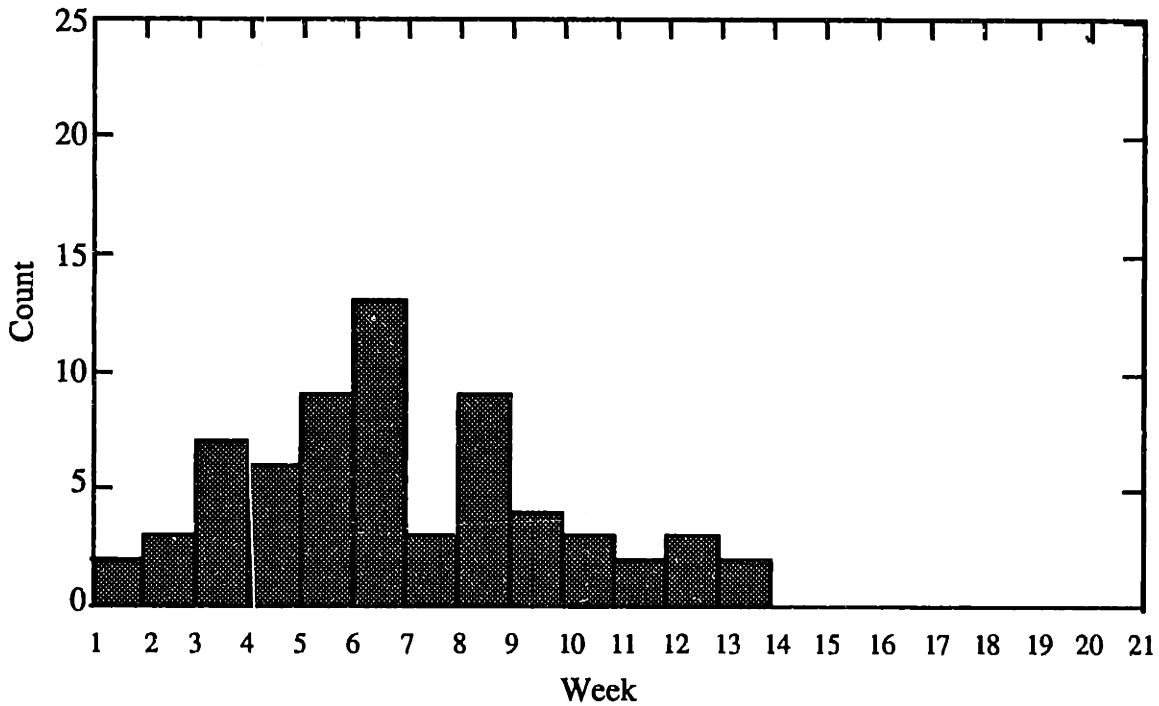

Figure 7.2: Sessions by week (MIT group)

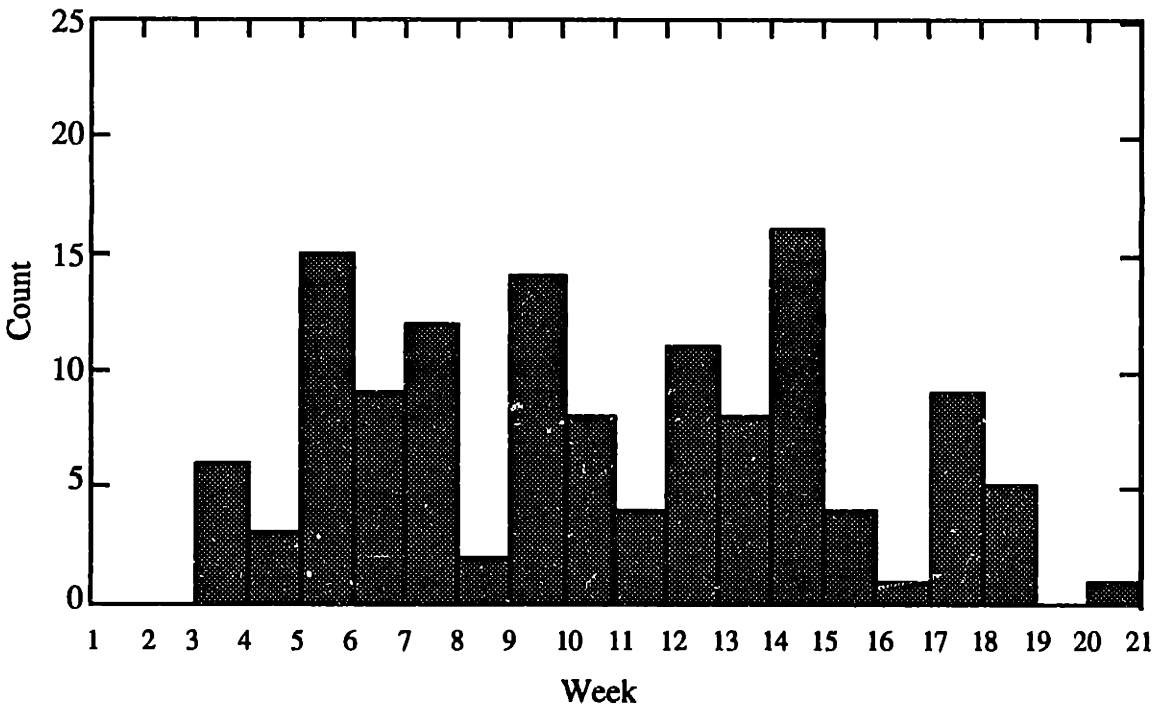

Figure 7.3: Sessions by week (Harvard group)

As might be expected, the usage from the class was driven by class deadlines, leading to a more irregular pattern. (One can even see the patterns of usage changing as late submissions of class assignments began to be allowed.) Usage was down during the Christmas holidays, and may have continued to trail after that period (i.e., after week 16) since only part of the Harvard class did programming assignments for their final project.

The two groups also differed in their pattern of introduction. In the MIT group, all but one participant had had their initial session by the eighth week of the field study. In the Harvard group, there were three distinct clusters of initial sessions, with most either in the first, occurring between weeks 3 and 8, or the second, occurring between weeks 9 and 15. This corresponded to the two periods for major assignments, and argues that enough people found it useful that new participants joined.

I believe that these differences in initial session patterns result from two sources. First, the tempo of the Harvard group was different and dependent on relatively short assignments; the MIT group was engaged in long-term projects. Second, the MIT group was substantially smaller and more socially cohesive. While the MIT group had a number of largely separate projects that used X on a varying basis, they worked together on a daily basis. Thereby, the introduction of the application to new participants could occur at a faster pace.

I expect that use of organizational memory systems such as Answer Garden will show similarities to the introduction and use in both locations. To the extent that an organizational memory system will be used by many organizational members or groups, perhaps separated in time and space, the introduction may reflect that of the Harvard group. As with the Harvard participants, there may be clusters of introductions, as the work groups find the system at differing times. Within each group, usage may appear more like the MIT group, where usage spreads quickly.

7.2.3. *User groups*

Even taking into account the user's location, usage varied considerably by individual. A simple diagram of the number of sessions by user (Figure 7.4) shows that six users were "heavy" users, accounting for 61% of the total sessions.

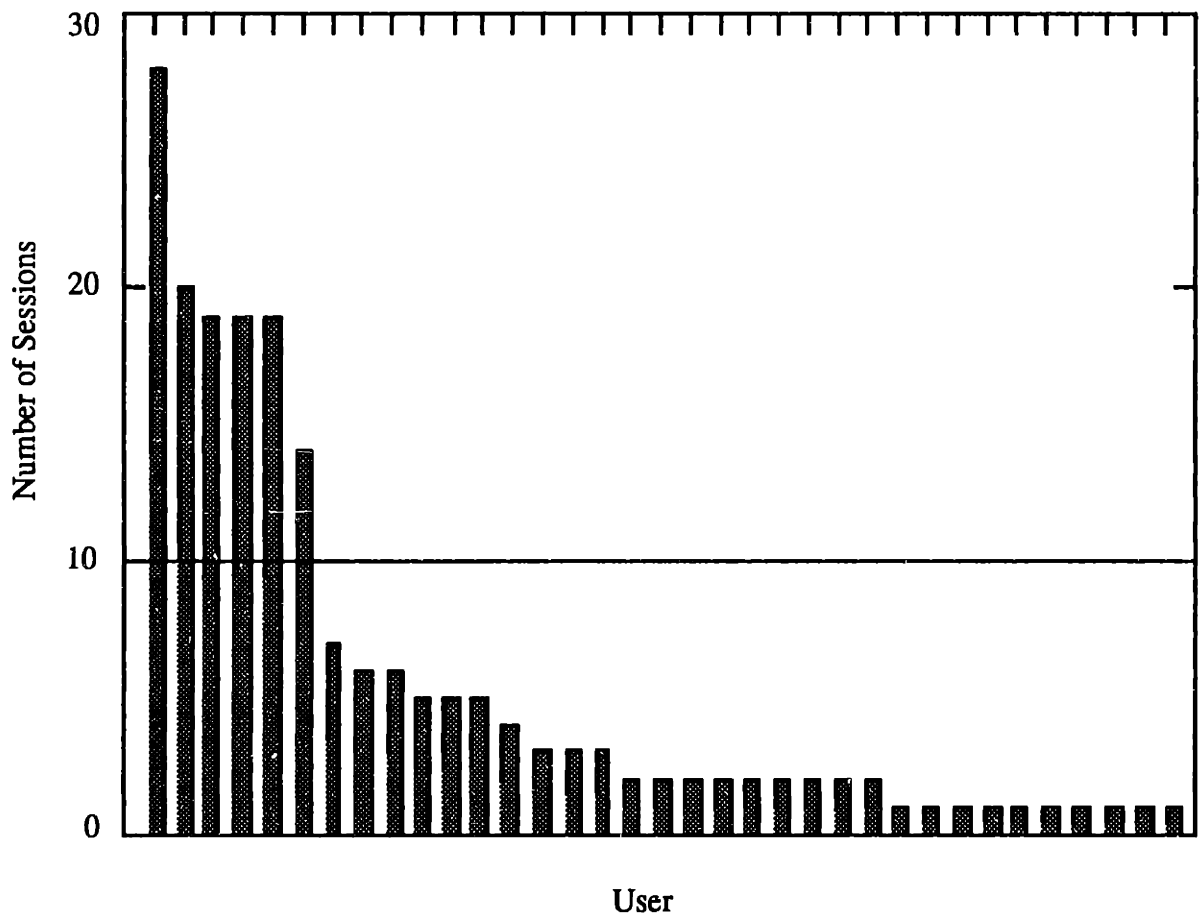

Figure 7.4: Number of sessions per user

The other participants could be further divided into two groups, a finding obscured in Figure 7.4. A further analysis of usage by person, broken down by the date of their session, shows that there were many intermittent users. These participants used Answer Garden more infrequently, but they used it over a long time period. Figure 7.5 shows all of the users. The sessions for the six heavy users are connected with a black line (users 13, 15, 16, 18, 27, and 32). The other, gray lines show users who did not use

the application as frequently, but who used it over an extended period. The vertical lines indicate the beginning of the Harvard study and the Christmas vacations.

There were an additional 12 participants who were classified as "intermittents." Some of these intermittents used Answer Garden for as long, but not as intensively, as the heavy users.

Participants were not included as intermittents unless they used Answer Garden in sessions stretching over at least a month period. The group of "tire-kickers," those who did not use the application for an extended period of time, are shown as solitary clusters of sessions. Seventeen people, or 49% of the participants in the study, were categorized as tire-kickers.

In addition, there were 27 participants who did not use Answer Garden at all. These "non-users" did not use the application at any point during the field study.

I attempted to find determining demographics for these user groupings. Research by Constant, Sproull, and Kiesler (1991, also suggested in Finholt and Sproull 1990 and Finholt 1990) showed that the use of mail archives is tied to geographical distance and, thereby, to information isolation. Other research has shown that education and age are correlated to use and adoption of information technologies (e.g., Chen 1982). Unfortunately, the statistical power for this sample size (n=35) was not sufficient to test for anything but large effects (Cohen 1977, Lipsey 1990) and, in fact, no statistically significant results were obtained.

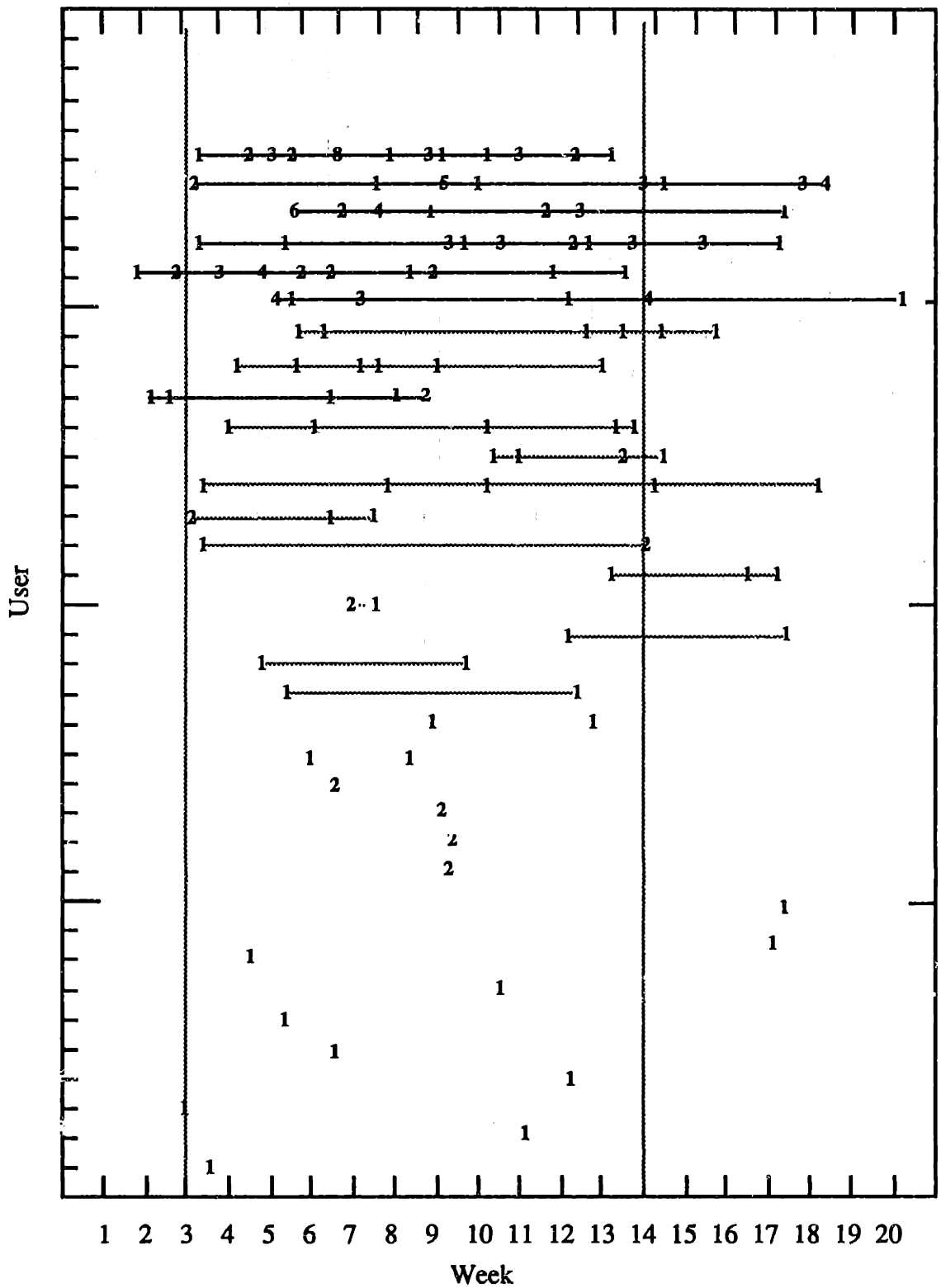

Figure 7.5: Usage by date and user, sorted by frequency of use

(numbers = the number of sessions for a user, aggregated over half-week periods)
(heavy users = dark line, intermittent users = dashed light line, tire-kickers = no line)

Using a series of contingency tables and Kruskal-Wallis tests, I attempted to find statistically significant differences among the demographic variables by the user groupings.[10] None of these tests were statistically significant, even when controlling for users' sites. Because of the small sample size, however, the lack of statistically significant results should not necessarily be interpreted to mean that no demographic variables affected usag .

Some indications of the relationship between personal characteristics and Answer Garden usage can be obtained from the observational data as well as the qualititative data in the final questionnaires. Although a user's decision to choose Answer Garden as an information channel cannot be completely separated from their evaluations of each session, users did provide some clues to their choice to use or not use Answer Garden in general. Overall, users appeared to choose Answer Garden because of a variety of subtle factors.

---

[10]I examined the heavy users (n=6) and intermittent users (n=12), the combination of heavy and intermittent users (n= 18) against tire-kickers (n=17), and the combination of heavy and intermittent (n=18) users against non-users (n=27). For all three of these user groupings, these contingency tables did not have statistically significant results. These included contingency tables for whether the user used the mail system, whether he sent mail, whether he had others at work of whom to ask questions, whether he had X experience, whether he used electronic bulletin boards, and whether he used Answer Garden at his work site.

The likelihood ratio chi-square test was used unless the Fisher exact test was appropriate. Because multiple chi-square tests were carried out, I planned on using a Bonferri reduction of the significance level from the .05 level. However, no chi-square test was even close to a probability level of .05.

Additionally, the Kruskal-Wallis tests of the variables measuring length of employment, time at the current company, and time at the current job against these three user groupings also had statistically insignificant Mann U statistics.

The heavy users appeared to incorporate Answer Garden into their normal work routine. They showed a noticeably different style in their use from the other participants. For example, four of the six heavy users kept Answer Garden running on their workstations as part of their normal suite of workstation programs. One heavy user, who often kept Answer Garden in an iconified state between uses, reported:

> You know that I find it valuable when I've always got it running. It's just one of my normal suite [of tools].

The heavy users' sessions were often elongated, with signs that they returned to use a running program after several hours or even days. The other two heavy users were unemployed, using Harvard workstations and thereby unable to have long-running sessions.

It is less obvious why the others divided into intermittents, tire-kickers, and non-users. Some of them had trouble connecting to Harvard remotely and using Answer Garden. Said one participant:

> All my work was done on BigComputer Research's network. Although on the internet, we block certain incoming protocols I believe.

Others echoed this:

> There was difficulty in logging into Harvard. I did not do my work here and it seemed to be an inconvenience [to use Answer Garden].

> I am not terribly familiar with the procedures for remote login and networking in general. Consequently I was unable to connect with Answer Garden and after a couple of tries gave up.

This was also true for both tire-kickers and non-users. Answer Garden, for them, was not a large enough lure to overcome the mechanical, start-up problems in using it. Others preferred more known methods:

> I never tried it. I was able to find my info in books that I had. I feel that Answer Garden is helpful for those who don't have the books on hand.

or could not make the time to learn the system:

> I was too intensely involved in debugging and coding and too far behind [in the course] to try to master another tool. As I found out later, there was nothing to mastering it.

These reasons are all standard problems in introducing most software systems. In addition, Answer Garden offered some new learning curves. For some, the problem was navigation problems in using the diagnostic questions and classification trees. Even a heavy user spoke of the learning curve for Answer Garden:

> There was a learning curve issue in finding out where the good information was, you know? I have a recollection of others having a hard time finding the good info. The people who persevered got good at it [Answer Garden].

However, users also spoke of more subtle issues. One important issue was an ambiguity in the users' minds about what questions were appropriate:

> I was often unclear as to what were legitimate Answer Garden questions. My problems seemed and often proved to be the result of "dumb" mistakes which I did not wish to impose on Answer Garden.

This was echoed in an MIT user's statement, which deserves to be quoted at length. This user was not representative of all users, but his thinking may be representative of those who did not use the system or who were tire-kickers. It should also be noted that this user was not present at any training sessions.

> I was actually a little late in discovering what AG was all about. At first I thought it was just an on-line hierarchical database of answers to common questions. In general, I find that these databases, like OLC on Project Athena, seldom contain answers interesting to me. ...Concerning AG, I never once expected to find an answer to a question of mine in the database, though I did check it before emailing a question to an expert.
>
> Once I knew AG would route a question directly to a human expert, I used it for every question I could not answer on my own. Typically, if I was confused about something, I would first casually ask another X programmer in my group (that is, verbally, if they were logged-in near me), then I would check a manual, and then, if I was still lost, I would decide to send e-mail to someone. I always went straight to AG at this point, instead of trying to guess

> someone who would likely know the answer, because I trusted that there was a mechanism in AG that would guarantee a satisfactory answer (I remember one time that a question of mine was forwarded to a second expert after the first had made a preliminary stab at an answer). A great deal of time is saved when the questioner does not have to play email-tag to find a person who knows the answer.
>
> The one inhibition I felt using AG, knowing that the experts were typically busy and working on projects more important than my little application programs, was a desire not to ask a stupid question. There was a certain air of formality about using the interface which often led me to double-check the documentation or to try to preempt any questions the expert might ask. I would read over the text of a question, realize that an obvious question would be "Have you tried Y?", then I would try Y and incorporate the result, if it did not clear up the problem, into the text of the question. This sort of anxiety is typical of what happens when I ask a question of someone I do not consider a peer. However, I do not feel it generalizes to computer-mediated Q & A services. For instance, I never hesitate to ask the most trivial, embarrassing questions on OLC because I have a certain contempt for the consultants and don't mind making them earn their pay. My inhibition in using AG would likely have been mediated if I had not personally known the people answering the questions. (Note that I never trust a computer that says that any action I take is "anonymous", which I believe AG claimed). Nevertheless, I did use AG to ask questions.

This response is interesting for several reasons. First, the issues involved appear to be complex and interwoven. On the side of using Answer Garden is the simple fact that the user will get an answer to his question, either from the database or from an expert. Against using Answer Garden was initial skepticism and misunderstanding about the nature of the application and a continued concern about losing status and not bothering the experts.

For this user, the twin issues of losing status and not bothering the experts were not clearly separated. Therefore it may have been difficult for him to believe that asking "stupid" questions was to be desired. Even if the status implications were removed, he might still have had problems with "bothering" the experts. However, the comments suggest that there was a group for whom the status implications were ameliorated but not removed. It may be that training, for these participants, was inadequate; it did not

properly set the correct expectations.[11] Alternatively, these participants may have had difficulty in supplanting these organizational norms for an isolated task since the norms are so prevalent in their work life.

His general concern appears to have resulted from two sources, knowing the experts and not feeling like a peer. Of interest was his not having a status interaction with help desk personnel, presumably because they are of equal or lower status. His concerns about experts, however, are likely to be true of anyone in organizational life, although no other user echoed his comments about being a peer.

As mentioned, this user was not representative. As will be discussed below, there were certainly users for whom being able to ask "stupid" questions was a relief. The session evaluations, moreover, show evidence that status interactions were minimized for many users. Nonetheless, these last two statements suggest that there was a group of users for whom the issues about information seeking were still complex and problematic. I do not think that simple demographics are likely to tell us much about why some people adopt Answer Garden in varying degrees; the issues are much too complex.

7.2.4. Length of sessions

Answer Garden sessions (n = 194) were short, measured either by the time spent or by the number of nodes accessed. While some users appeared to be engaged in browsing the information in the system, the short sessions were consistent with users' seeking information to solve specific problems.

---

[11]This argues that training not only must educate potential users in the mechanics of the system, it must also train them in the organizational expectations around that system (Orlikowski 1992).

In the study, the average session time was 14.0 minutes (s.d. of 23.8 minutes). However, there was a considerable range, from 0.1 minute (an abortive session) to 3.3 hours (a session marked by several long periods of inactivity). Most sessions were short, less than ten minutes as shown in Figure 7.6. (The six sessions above 75 minutes were removed from this graph as outliers.)

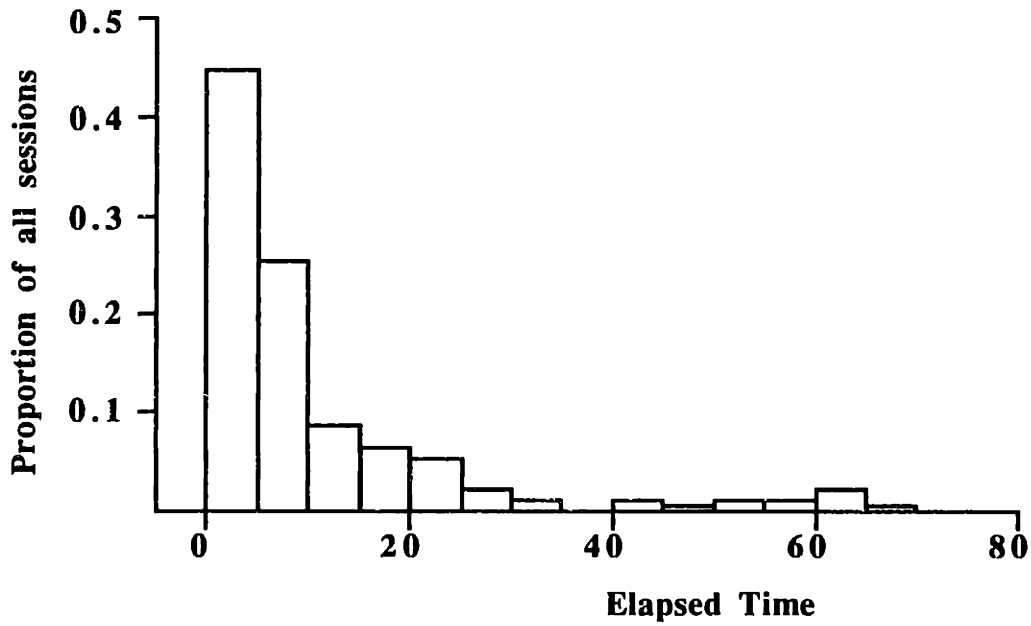

Figure 7.6: Times per session

The propensity toward short interactions can be seen as well in the number of nodes opened. Over 70% of the sessions opened less than ten Answer Garden nodes. Moreover, the time spent per node was quite short and did not vary a great deal. The average time spent on a node was only 1.2 minutes (s.d., 2.5 minutes). This distribution is extremely skewed, with 78% of the sessions having a per-node time of less than one minute (and slightly over 50% being less than 25 seconds). While this per-node time includes a mix of navigational nodes (such as the Graphers) and of information nodes,

users clearly did not spend considerable time reading the nodes in detail. It was, however, consistent with users browsing for information.[12]

Moreover, users spent relatively the same amount of time per node regardless of the type of session. The amount of time spent on a node was not related in a statistically significant manner to their usage level. Nor was it related to whether the session was the user's first session or whether the user used mail.

However, the number of nodes opened by a user was dependent on other factors. Users tended to open more nodes when first using Answer Garden. The Kruskal-Wallis non-parametric one-way analysis of variance test of the number of nodes by whether this was the user's first session was statistically significant at the .05 level ($n = 194$, Mann-Witney $U = 1714.5$, $p < .001$). This may have been from exploration, using the diagnostic questions instead of the graphical trees, or from navigational problems when learning the application. The number of nodes opened on this first session was not statistically significantly related to the user's later usage level.

For subsequent sessions, a Kruskal-Wallis one-way analysis of variance test showed that the relationship between usage level and the number of nodes opened by a user in a session was statistically significant at the .05 level ($n = 159$, Kruskal-Wallis $B = 17.60$, $p < .001$). Table 7.2 shows the means, standard deviations, and medians for the three user groups. Those who were tire-kickers did not differ in their later sessions in a statistically significant manner from their first session. However, those who were high users were characterized by much shorter sessions.

---

[12]A person can browse through three to four pages of text in 20 to 25 seconds (Gary Marchionini, private conversation).

Table 7.2: Number of nodes opened by usage level (non-first sessions)

n	USELEVEL	mean	s.d.	median
159	tire-kickers	10.08	9.38	5.00
33	intermittents	11.21	8.47	9.00
113	heavy users	5.95	5.38	4.00

An examination of the three groups' distributions of number of nodes opened shows differing distributions. The tire-kickers have both long and short sessions in nearly a uniform distribution, while the heavy users have an exponential distribution. The intermittents have a distribution that is in-between. This suggests that, as users continue to use the application, they tend to have more short sessions. There is, however, no obvious relationship where the number of nodes accessed necessarily decreases over time on the system. This tendency towards more short sessions may result from some users' already knowing the application, either being able to jump into the middle of an information database using the graphical trees as navigation nodes or having browsed the information nodes to know what is in the information database already.

Session time was dependent, as might be expected, on whether the user used the communication facilities within Answer Garden. The relationship between sending mail and the logarithmic transformation of the elapsed time was statistically significant ($R^2 = 0.172$, $p < .001$). This was as one might expect, although the R^2 is quite low. When the time for using the mail facilities was removed, there was no statistically significant difference in the session lengths between those who used mail and those who did not.

What users appear to be doing is to be using the application in nearly the same fashion over time. They do not change their time spent per node in later sessions or in

sessions where they do not find their answer. However, the users tend, over time, to have more short sessions, perhaps as the result of knowing the information database better.

7.2.5. Mail usage

Mail usage was heavy. In 53% of the 194 sessions, mail was sent for a total of 121 mail messages. (In 16 sessions, users sent multiple mail messages.) There were an additional 43 times that users opened mail windows, but did not send any message.

The reasons were very obvious in the critical incident interviews. A typical user response during an interview was: "I could have used more info [in the database]."

Largely this was the result of an incongruity between the coverage in the database and the size of the topic domain. The X information database was small (98 nodes at the start of the study with approximately 225 questions and answers) compared to the number of possible questions and answers about X (at least thousands of potential questions). This suggests two possibilities, that future Gardens should be restricted to smaller domains so users will be more likely to find answers to their questions already in the information database, or that the initial period of activity will be largely devoted to building the domain coverage.

7.3. Evaluations of Answer Garden

The qualitative evaluations of the application ranged considerably. One end of the continuum was the terse "I don't like systems like this" without additional elaboration. The other end was from a student on the class evaluation at mid-semester: "Now that we have Answer Garden to answer questions, the TFs [teaching fellows] need to find something else to do."

People's reasons for not using Answer Garden overall, based on their end questionaires as well as some additional interviews, were presented above. Below are

their evaluations about specific sessions, gathered during the critical incident interviews. Following the user evaluations are the evaluations by and of the experts in the field study.

7.3.1. User's' evaluations of Answer Garden sessions

As mentioned, there were 49 critical incident interviews, conducted with the participants about their use of the system on a session basis. Users offered 30 negative statements and 23 positive statements about the system. (Several participants gave mixed responses.) Overall, the majority of numeric scores in these same critical incident interviews were favorable.

Two findings stand out in the user evaluations. First, the users primarily wanted an accurate solution quickly. Second, the indications were that users felt comfortable with seeking answers through a combination of information retrieval and communication systems.

In general, the overwhelming emphasis in the performance evaluations for the sessions was that the users were satisfied as long as they got an accurate solution quickly. This was affected by several factors. Users rated the application more highly when the answer was already in the information database or when they got a quick response by electronic mail to their question. A typical evaluation was "It was very good. Whoever answered it was exactly right, and very quick too." Participants gave an unfavorable rating when they got no answer or a slow answer. In fact, almost all (83%) of the unfavorable responses were due to not being able to get the information quickly, either in the information database or by electronic mail.

In addition, users did not like it when they had to wade through a considerable amount of text to find that the answer was not present. If the answer was not in the information database, they wanted to know quickly so they could ask their question via electronic mail. In fact, being able to quickly determine that the information was not in

the database was often seen as a positive attribute. A majority (60%) of the favorable responses in the interviews mentioned being quickly able to determine whether the information was in the database, getting a quick response, or finding the information in the database.

When asked about the information itself, users had a mixed reaction to the information gathered through Answer Garden. In nearly 20% of the interviews, participants pointed to Answer Garden as providing information they could not have received elsewhere. However, about a third of the users reported problems with the specificity of the material (either too high or too low) and the level of the explanation provided by the expert (either too high or too low).

As mentioned, users appeared to also be comfortable with information seeking through a combination of standard information retrieval system and the social network. This conclusion is based on indirect evidence, primarily the differences in comments between the first weeks of the study and later. During the first weeks of the field study, users had to be prompted to accept the use of the social network as part of the information retrieval process. Participants consistently reported not wanting to bother the experts or expressed surprise at the idea that they could ask the question of a human if the answer was not present. (They had been told how the application worked repeatedly in introductory talks and handouts, implying that they were not used to the idea of Answer Garden at first.) Several steps were taken to reinforce their understanding, including a sentence on the initial node reminding them to ask questions. After the first month, the performance evaluations suggested that their continued usage of the application was natural. Users stopped evaluating the information in the database separately from the electronic mail response; the system evaluation was based on both together. Users did not answer that they did not ask a question through the system because it seemed strange or novel even though they were prompted for such responses. By the middle of the field

study, tying the social network into the information retrieval system appeared to be normal to the users.

In addition, Answer Garden seemed to ameliorate the status implications of information seeking for many users. No one mentioned negative status implications in the critical incident interviews. This was true for both the people who sent a question and for those who did not.

In fact, session evaluations were on the whole appreciative of reducing the status interactions in information seeking. When it was mentioned, the users liked being able to ask their questions anonymously. Said one interviewee: "[It's] a vehicle where you're not intimidated; you're asking anonymously and through text." In 25% of the interviews, participants said that the application helped them ask for information that anyone would be expected to know. Another 56% of the interviewees said that their questions were intermediate in difficulty.

These session evaluations contradict some users' summary evaluations of Answer Garden. It may be these concerns were the distinguishing factor between tire-kickers and people who used the system more. All of the summary evaluations that mentioned status implications negatively came from tire-kickers or non-users. It may be also that some users felt comfortable in individual sessions, but kept a background concern when evaluating whether to use Answer Garden in general.[13] Nonetheless, some users felt a reduction in the status interactions; this may not have been true for all users.

---

[13]I also considered whether the session evaluations were worded badly and did not adequately elicit feelings of status interactions. It was difficult to directly ask users about status implications without prompting that consideration in their minds during later sessions. Therefore, there were two indirect questions in the session evaluation. The questions were "Why did you use [information channel] in that order?" and "Why did you rate using the system in that manner?" The former question allowed the user to

7.3.2. Experts and Answer Garden

As mentioned, there were seven external experts in the field study, all of whom had more than two years of experience. As mentioned, many of the experts answering questions were directly responsible for parts of the X Window System or its derivative vendor releases. The design of the field study tried to keep the workload on any of these external expert to a minimum. The most questions given to an external expert was two per week.

Five of the experts expressed initial fears about being overwhelmed by the task of answering questions. Accordingly, the experts demanded and were given the right to refuse to handle questions based on their workload. This rule was invoked by some experts for almost 50% of the field study, suggesting that redundancy in expertise will be required for similar systems. The external experts also requested the right to not answer any question that required excessive time or was too specialized; however, this rule was invoked only once during the field study.

---

respond that he did not wish to use Answer Garden as an information channel to avoid status implications; the latter allowed the user to include status implications (or other considerations) in the evaluation of the session. Careful attention was paid for hints of status interactions in the responses, and when there were such responses, follow-up questions were asked.

It should be pointed out that if these questions had been worded badly, then users should not have mentioned status interactions at all in the session evaluations, either positively or negatively. However, some users did answer that there was a reduction in status implications.

The differences between the summary evaluations and the session evaluations need to be explored in further work.

No expert complained about the time required for answering questions. However, the style of answering was an issue in the study. Four of the experts showed a marked formality in their responses. Their answers were longer, containing substantially more detail and more cases than would a quick electronic mail response. When interviewed, these experts reported wanting to provide a more complete answer for two reasons. First, they tried to immediately generalize from the specific situation to a general answer. Moreover, they wished this general answer to be suitable for a large range of questions and special cases instead of growing the answer iteratively over time. Second, several reported that their response served as a public "badge" for them within the organization or community. As such, they wanted to make sure that their answer was complete and accurate - not doing so might reduce their status with people who did not know them well.

This push towards formality of response goes against the users' desires to have short, easily readable answers and the system goal of capturing informal flows of information. Not publicly identifying the expert might reduce the formality and further reduce the status implications in information seeking, but users use this identification to judge the potential accuracy of the information. This dilemma may argue for the necessity of an information "editor" or "moderator" to ensure authoritativeness and consistency in the information database.

Nonetheless, three of the external experts did provide informal responses. This suggests the possibility of keeping the responses informal. Again, an editor or moderator might be valuable in some situations; he could work to keep the tone informal.

7.4. Authoring observations

Authoring issues form an important nexus between the social and technical aspects of Answer Garden. They are, in large part, social with their emphasis on shared

meaning and rhetorical structure. However, authoring issues cannot be clearly separated from the technical context since the system features and constraints play a large role in the actual authoring process.

No design assertions were made about the authoring process, so the findings below are quite tentative. Additionally, I did much of the authoring mechanics in the field study, also restricting the generality of the findings. Below are observations gathered from research journal and mail log entries. Most of them have been confirmed in further authoring studies (not reported in this thesis).

- The authoring task can be roughly divided into three subtasks: writing the actual information, building the diagnostic or classification network, and placing the material in Answer Garden. The authoring of the actual information takes as long as any writing takes. The Answer Garden emphasis offers two advantages. One is that the writing can be iterative, so the author can begin by writing only part of the intended corpus. The other is that one can use electronic mail messages and other pre-existing materials to begin the writing. Additionally, writing can be easily shared.

  The second subtask is to develop a classification or indexing structure. If diagnostic questions are desired, additional time must be spent on developing adequate diagnostics. The final subtask is that of putting the information and classification structures into the form required by AGS. In the field version, this took time for the pre-existing materials since that material needed to be cut and pasted into the correct node.

Automatic tools are absolutely required, but they will not substantially relieve the burden of writing or of classifying. Without automatic classification tools, they will relieve only a third of the authoring task, that of physically placing the information in the format and location of the organizational memory system.

- An additional time requirement is to turn the questions received from the users into more generalized questions. Note that this is not the same as providing a complete answer; it is one of making the question itself recognizable and useful to other users. Often, questions come with the specifics of the situation. Research by Aaronson and Carroll (1987a, 1987b, 1987c), among others, has shown that many naive users cannot separate their question from the specifics of the situation. While this may not be a problem for the user examining previous questions, the author may need to rephrase the question and answer in order to be more general.

- Several authoring considerations were forced by the nature of the system. Of particular interest was that Answer Garden, in order to succeed, had to provide the appearance of social activity. For example, a node called "New Questions and Answers" was added to the top level of the diagnostic tree so users could see that other people were using the system. Attempts were also made at social inclusion by using the second person in answers and requesting follow-up questions if the answer was unsatisfactory, incomplete, or incorrect.

- Users wanted to quickly skim a node to determine whether it contained the solution to their problem. One rhetorical device that appeared to alleviate this issue was to provide a summary at the top of the node.

- As mentioned above, the inclination by some experts to provide complete answers needs to be weighed against the users' desire to have short, concise explanations. Users repeatedly expressed a dislike for large amounts of information to scan in order to determine whether an answer to their question was present. It may be fruitful in later versions of Answer Garden or in similar systems to have several levels of explanation. In other words, it might be better to have "a short answer" and "the real explanation." This would, however, increase the writing effort for experts.

- It was hoped that the authoring would not be done entirely at one time, but would be incremental and iterative. There was some evidence of this. I changed the classification network three times during the field study as the result of new types of information and comments from users. One such comment was a request for tutorial materials, and the resulting nodes required a reorganization of the classification network. Other reorganizations involved the inclusion of the "New Questions and Answers" node and the rearrangement of information about user interface objects.

  However, no user requested additional information about an existing answer (i.e., one already in the information database) that he felt was incomplete or incorrect. This may have been the result

of the users' being satisfied with the information, or it may require additional emphasis during implementation.

- The size of the information database appears to have an important psychological effect. Initially, it was thought that an empty database could be installed, and people would build an Answer Garden entirely by their use. The experience in an early alpha site belied this premise when the participants expressed a lack of interest in using an empty Answer Garden. One participant reported that he did not feel as though anyone would answer his question when the database was empty.

  However, additional mechanisms, such as visual meters indicating how much the database has been built up, might ameliorate this psychological effect. These mechanisms would be additional indicators of social activity, similar to those required in the field study.

  Furthermore, the size of the X information database may have led to some of the effects observed. For example, a sparse database may lead people to feel as though they should ask "stumpers" instead of easy questions. Stocking an information database with a few hundred "stupid" questions might prompt users to ask similar questions.

I believe that the above observations should be true for any application that uses questions and answers, and not just Answer Garden. Some of these observations, such as the rhetorical need to demonstrate social inclusion and activity, should be true for all

organizational memory systems (if not all CSCW applications). However, further research using alternative applications will be required.

7.5. Technical findings

Another goal of the field study was to determine what technical changes would be required to place Answer Garden and the AGS in the field. There were four major technical findings as a result of the field study.[14]

The first was that the expertise engine lacked the complexity required in use. The design of AGS required that each expert be associated with a specific Answer Garden node. (An expert might be associated with many such nodes, however.) Actual use showed two additional situations. One was that an expert would be associated with a subnetwork of nodes. The other was that a group of experts would be associated with some node, with their exchanging effort depending on their workflow. Changing an expert in the system involved editing a large number of nodes rather than a single data item. Further, there was no easy way to override an expert dynamically, for example, depending on the expert's workflow or vacation schedule. A rule-based expertise engine with inferencing would have made this easier. Furthermore, separating the expertise engine from the information database would make system maintenance and updating easier.

The second finding was that the design decision to include only one-way (forward) hypertext linkages in AGS was unfortunate. The lack of two-way links made it

---

[14]It should be noted that Answer Garden and the AGS had gone through three major and innumerable minor design and implementation iterations by the time of the field study. Additionally, many of the technical requirements for field deployment had been determined from the alpha sites. However, the field study was the first deployment to use Answer Garden for an extended period of time.

difficult to create an inferencing expertise engine. An inferencing expertise engine would need to know all of the parent nodes (and not just the user's history) in order to infer the correct experts from a rule system based on subtrees. Additionally, the lack of two-way nodes made it difficult to have additional types of "go back" buttons in the system and to do certain types of adaptive information retrieval engines. Such two-way linkages could be added to the Answer Garden Substrate as it currently exists if a secondary file with the links were added.

The third finding was that the system did not scale as required. User interface problems resulting from large numbers of nodes were resolved by including additional information retrieval engines. However, the problems of social scaling were more difficult. Early in the development of Answer Garden, it became necessary to allow local experts, where these local experts would occupy a position between the user and the general experts. Examples of such local experts might include help desk personnel or local "gurus." That neither of these two parties wished to be eliminated was pointedly conveyed to me long before Answer Garden even went into the alpha sites, and I felt that implementation success required not ignoring the existing social mechanisms.

The Answer Garden Substrate does have a number of mechanisms involved in allowing local experts. In AGS, a site manager can provide a set of local experts for any nodes that have been changed locally. Additionally, the site manager can specify that all questions go to the local experts as in the case of a help desk. (If the local expert needs to forward the question, a Forward-To field in the mail message will contain the names of the global or node experts.) However, these mechanisms did not scale adequately for all organizational situations. For example, there is only one set of local experts; they cannot be specified for subtrees. Moreover, the current mechanisms cannot handle tiered help desks.

The final finding was that the set of states assumed for the Question-Answer Tracking Subsystem (QA-Tracker) was not complete. The QA-Tracker was to track the state of each question and answer, and in doing so, it needed to correctly model the process of asking and answering questions. While it was prototyped in Oval, with no assumption of completeness, the QA-Tracker was hand-simulated in the field study. A technical goal was to further determine the correct model for the question-asking process and the states necessary for support in a QA-Tracker.

The QA-Tracker states included in the prototype were:

Question asked (Q)	User asks a question.
Question forwarded (QF)	QA-Tracker sends mail message to expert.
Forwarded question accepted (FA)	Expert agrees to answer question.
Answer returned from forwarder (QA)	Expert answers question.
Question refused (QR)	Expert refuses the question (for any reason). QA Tracker then goes to next expert if there is one.
Cancel question (CQ)	User cancels the question.
Cancel request for answer (CA)	QA-Tracker cancels the request for an answer from the expert.

This list of message types expanded during the field study as I observed the users' and experts' behavior. Many users, as well as myself, sent "thanks" messages acknowledging that the answer or effort was helpful. Additionally, experts needed messages nagging them to answer questions. (The QA-Tracker must also cancel questions when the expert has taken too long and must go onto the next expert.) Furthermore, when users ask follow-up questions to the experts, those questions should be labeled as such both for the experts' benefit and for authoring tools.

One further type of activity required in a QA-Tracker is the result of users wanting to know the status of their question. The QA-Tracker must send out status messages at regular periods so the users (and experts) do not lose faith that the application is working. This was hand-simulated in the field study.

Additionally, the experts asked for the following types of messages.

Question refused for time period	Expert refuses the question (for any reason) for a period of time. This might be used if the expert wanted to answer the question only if it were still outstanding after the specified time period.
All questions refused for time period	Expert takes himself off all expert lists for a specified time period. This allows the expert to adjust his workflow.
Question refused but answer requested	Expert wishes to know the answer when one is available.
Question forwarded by expert.	Expert has forwarded the question to another person.

The last two are particularly interesting. Often experts may not know the answer. They may wish to know the answer when it is provided by another expert. In addition, they may know of whom to ask and wish to forward it themselves..

All of these findings argue for the improvement of the "back-end" of Answer Garden and AGS. They will translate into changes in the user interface for Answer Garden, but largely they suggest that the social use of Answer Garden requires additional technical support.

7.6. Summary

Answer Garden was used. However, its usage differed from many systems in that it was used as an occasional tool and sessions were short. Users showed evidence of browsing for solutions to their specific problem rather than performing a detailed reading of the information database.

Answer Garden users could be categorized into three groups. There were a set of heavy users who showed evidence of incorporating Answer Garden into their normal work life. There were also a set of intermittent users, who used Answer Garden less but used it over an extended period. Additionally, there were a set of tire-kickers who used Answer Garden a small number of times. No distinguishing demographics were found to explain the differences between the groups, although it was suggested that the issues involved choice of information channels are quite complex, subjective, and interwoven.

Evaluations of Answer Garden were, in the majority, favorable. Evaluation of sessions by the users appeared to be clearly tied to speed of information access. Users appeared to be happy if the information was in the database. They also rated the system favorably if they could tell the information was not in the database quickly and if the answer to their electronic mail question came back quickly. Their evaluations and comments suggested that by the end of the study period, they were comfortable with information seeking through a combination of information retrieval system and social network.

The goals of reducing status implications in information seeking and reducing the need for reciprocity were partially satisfied. There were many users who showed evidence of information seeking without status implications. However, there were a group of users for whom Answer Garden did not reduce or only ameliorated the status

implications and need for reciprocity. Further training or changes in system clues may be required to reach more potential users.

Chapter 8: Conclusions

This thesis started with the statement that socially influenced software design had the social world as its central consideration. Answer Garden, and the other AGS applications, attempt to augment information seeking and memory in organizations. To do so, they necessarily hypothesize about the social world through their design assumptions, and they offer at least the possibility of examining that social world through their design assumptions and operation.

8.1. General conclusions

How successful were the design assumptions behind Answer Garden? The field study resulted in the following.

- I asserted that one type of organizational memory system could combine retrieval from information repositories with access to organizational members. I found through the field study that such organizational memory could be built, although the final coverage of the information domain was not extensive. Users employed the system to find information by accessing the questions and answers provided or by asking new questions through electronic mail. The answers to these new questions were inserted into the information database, thus growing the corpus of questions and answers. The combination of information retrieval and communications system was key to the successful operation of Answer Garden. Moreover, users appeared to find the combination comfortable.

- I also asserted that such systems should provide suitable incentives for use. The incentives appeared to work in the field study. Users found the system effective when they received correct and timely answers to their problems, and they were willing to ask questions through the system. However, the incentives for the experts were not tested since the system was not available for an extended period.

These findings demonstrated that Answer Garden could work in principle. However, these first two findings could have been the result of any field trial of a system.

The relationship of socially-oriented design to the social assumptions can be best seen in the considerations about information seeking. I asserted that such an organizational memory system as Answer Garden should result in the reduction in status implications and need for reciprocity. The field study resulted in both a mixed achievement and some new insights about this design assertion:

- I found that Answer Garden could reduce the status implications for many information seekers. The ability to ask questions anonymously and to ask questions of a correct expert were found to be beneficial by a number of users.

  However, some users still had issues concerning the status implications in information seeking through the system. Whether these users would find their concerns ameliorated over an extended period of use is unknown. It may also be possible to ameliorate the status implications by providing access to lower-status help desk personnel.

- For the experts, the field study uncovered the need for experts to maintain their organizational "face." I had not expected, before the field study, that experts would also have status implications in their information providing role. Further work will need to concentrate on finding mechanisms to reduce any status implications for the experts.

- The need for reciprocity appeared to have diminished, if not disappeared, in the two field sites. Users asked questions through the application even though they were not providing information back to the experts.

- A number of users continued to fret over their "bothering the experts." This could indicate continued concern over the status implications in information seeking. However, from the users' responses, it also indicates that software R&D engineers may have an additional social consideration when seeking information that had been previously masked by the concern over status implications.

  One possibility for these comments might be a respect for the experts' position. The words the participants in the field study used could indicate a respect for the tasks and duties of people above them in a technically-based hierarchy. Because software R&D engineers recognize technical expertise as a meritorious achievement and because their authority is based on technical

expertise, they may not wish to "bother" the experts.[1] This situation would then show characteristics of Blau's example of a medical bureaucracy with its respect for professional authority (Blau and Meyer 1987). It is possible that the status implications were masking this additional concern for the experts' role and position.

If such a concern does exist, Answer Garden does little to ameliorate it. Indeed, Answer Garden may exacerbate it. At least in the field study version, Answer Garden identifies some organizational members as experts, placing them automatically above the user. This would imply that the clear-cut separation of experts and users is not only artificial, it may lead to operational difficulties.[2]

An alternative possibility is that the concern resulted from the experts' being volunteers, rather than being organizationally responsible for these answers. In other words, users felt that answering questions required labor that was not organizationally rewarded. Shifting incentives might change the behavior and comments of the users. While I gave users knowledge of the incentives for the experts, it may not have had as much meaning to the users as clear organizationally based rewards.

---

[1]This is not the same as a status interaction (or concern over one). A status interaction involves a two-way exchange or the potential of one; this is a deference to authority.

[2]I will return to this point below.

Further research is required to differentiate whether either respect for professional author or concern over organizational rewards causes such comments.

- An interesting, and provocative, finding from the field study was that a large proportion of the users did not get answers that were at the right level or length of explanation. In reflection, the assumption that users should always have their questions answered by experts may have been false.

  The design of Answer Garden in the field study assumed that the status implications and need for reciprocity were only negative and should be removed. It may be, however, that they also provide positive organizational benefits. For example, the status implications in information seeking and the need for reciprocity may serve to channel information seekers towards others at their expertise level; thus, providing the seekers with answers at the right level and length of explanation.

  If this is correct, then it is not the status implications and need for reciprocity that are problematic *per se*. The organizational problem arises when the information seeking can be considered dysfunctional (Blau 1955) or even pathological (Scott 1992). For example, an organizational dysfunctionality occurs when the people at the same expertise level cannot answer a question and there is no person with greater expertise available.

In general, these latter findings showed that the social issues drove the specifics of use. The key findings from the field study were clearly social.

8.2. Caveats: the dark side of Answer Garden

Relatively little can be inferred from the field study about the long term effects of such an application. Nonetheless, Answer Garden is a tool that has potential negative consequences as well as positive ones. I would be remiss if I did not raise some general concerns.

First, Answer Garden could lead to a bifurcation between globally-known "experts" and the masses of "users." This is particularly a problem with the assumptions used in the field study. For example, the field study version of Answer Garden did not allow a range of expertise for people answering the questions, nor the ability for fellow users to answer. Such a separation between expert and user tends to reinforce the assumption that there is, in fact, a clear separation of expert and novice. There is a range of evidence (e.g., Draper 1985) that there is no such dichotomy; instead, there is a range of expertise.

This assumption is likely to be exacerbated with a tool such as Answer Garden. Since the declaration of being an expert is made public, global, and organizationally sanctioned, such an expert will be likely to held in high regard by organizational members. Such publicity might lead to the reduction of organizational recognition for local expertise. Furthermore, the separation between expert and user may condition organizational members to think in terms of a dichotomy between "knowledgeable" experts and "unskilled" users over time. People who gain organizational recognition, rather than that of their peers, will have authority and will be believed; those without such recognition may be considered to be without expertise. Such effects could have a large number of detrimental secondary effects such as the reduction of opportunities for organizational members to learn new skills, the de-skilling of certain occupational groups, and the reduction of expertise redundancy in the organization.

Second, expertise is provided by organizational members in many forms. Since the number of questions and answers can be easily counted in a system such as Answer Garden, organizations might be tempted to base their reward structure upon that count. Furthermore, the management of organizational expertise could also be focused on such a count. Such a focus would be misguided, leading to the better management of only a portion of organizational expertise.

Finally, systems such as Answer Garden continue two tensions: Who owns knowledge and memory? And, how much should knowledge be distilled and separated from the people within a firm? The model of Answer Garden that was used in the field study argued that the organization or community owned the knowledge instead of the organizational members.[3] There is clearly a large range of possibilities for the question of ownership. It should be noted that it is often a managerial interest to argue organizational ownership. It is also a managerial interest to foster the abstraction and the removal of knowledge from organizational members. This is not necessarily wrong-headed. Clearly there is some knowledge, and its concomitant memory, that is based in proprietary achievements and resources. However, there is other knowledge that is professional and personal. More importantly, some important organizational knowledge and memory cannot be usefully separated from people.[4] As organizations seek to capture

---

[3]It should be noted, however, that there is nothing in the Answer Garden Substrate technology requiring this to be the case. For example, experts could be paid when they provide the information or according to the number of people accessing their knowledge. They might also be paid to maintain the information over time. Additionally, an Answer Garden information database could include only routing information for finding experts, so that the "knowledge" remained hidden.

[4]Indeed, one could argue that only people can distill the memory for best use and can provide sufficient contextual richness for efficient retrieval. This does not argue against organizational memory systems, just against unconsidered and indiscriminate belief in the utility of such systems.

more and more of the informal communications and working knowledge of the firm, while removing the organizational members' vocational security, this will result in additional tensions around information-providing.

8.3. Implications of the study and of the system

8.3.1. Implications for software engineering

Answer Garden was found to provide software engineers with two types of help. It provided information about technical details that are often obscure or difficult. It also provided help to software engineers learning a new technological system and skill. Providing information databases of informal information and working knowledge, based around sets of commonly asked questions, may lead to the easier adoption and learning of new technologies. The software engineering world changes rapidly; tools such as Answer Garden may provide some amelioration of the effects.

8.3.2. Implications for organizational memory systems

Answer Garden argues for a particular way of constructing and using organizational memory. While there are a number of possible designs for organizational memory systems, I tried to present strong theoretical arguments for combining information retrieval and communications. The field study showed that such a combination is possible and useful.

Answer Garden also argued for capturing informal memory. It attempted to provide incentives for both the information seekers and information providers. To the extent that any other systems attempt to capture informal memory, they will find themselves concerned with the social issues of status implications, reciprocity, and "face," as was found in this study. The field study's results hint that the incentives behind Answer Garden may carry to other systems. Providing individual-level incentives, where

each person follows his own best interest, can lead to collective behavior with promising results.

In addition, the field study argued that the contextual, social issues of organizational memory are critical to the adoption and use of systems to augment that organizational memory. I believe that organizational memory systems will ignore such social and organizational issues only at the risk of being unusable, non-useful, or dangerous.

8.4. Limitations of the study

Any study is necessarily limited. This section attempts to discuss several of the more important limitations in this study.

First, the generalizability of the study based on the particular study population is open to question. Software engineers are not representative of the population as a whole. Indeed, I argued earlier in this thesis that software engineering, especially user interface engineering, may not even be similar to other types of engineering. I would not expect necessarily that these findings would apply to groups with career anchors other than technical anchors. Nonetheless, to the extent that groups find it difficult to seek information because of status implications and the need for reciprocity, similar systems may have similar effects. One might expect other engineering populations to be roughly similar; one would not expect this of sharing professions such as teachers or nurses.

Furthermore, the two study groups were engaged in research or in education. Research organizations have different information-seeking and information-sharing characteristics from engineering groups (Allen 1977). However, one of the sites, the Harvard class, did have working engineers using the system within their organizations. Furthermore, informal trials within a few computer companies resulted in similar results. One might expect similar results about information seeking within other organizational

settings. However, use within a commercial organization might result in very different information sharing characteristics. For example, Orlikowski (1992) found in her study of a consulting firm that information sharing was not organizational rewarded and, therefore, did not occur readily. The generalizability of the study results will not be clear until there are additional studies within other organizations.

Second, there is limited generalizability, at best, from the knowledge domain used in the field study to a wider variety of knowledge domains. Some knowledge domains would not fit easily within Answer Garden. Domains with unclear analytical boundaries, where the database would be difficult to navigate, would not fit well. Additionally, domains that are ill-defined, poorly or ambiguously categorized, sociologically reflexive, open to clearly subjective definition, or dominated by "why" knowledge might not fit well into an Answer Garden. The field study employed a knowledge domain that was technically oriented, and as such provided relatively clear definitions, distinct categories, and an emphasis on "how-to" and "what" knowledge. To the extent that a knowledge domain is technically oriented, the findings here may be generalizable.

Third, the generalizability from this study is also limited by methodological issues. There was a lack of randomized sampling in the critical incident interviews. I have used only the qualitative responses from those interviews, and weighed their evidence carefully against other conversations and interviews with the participants and against observations of other sites. Another methodological issue was that I both interviewed participants as well as built the system. I tried to avoid bias both in myself and with the participants by arguing that I would learn more from failure in this research than success. Furthermore, I had others conduct some interviews to confirm the data received from my interviewing. Nonetheless, because of all of these factors, the results could be biased. Again, more research with Answer Garden, or similar systems, will be required.

The last limitation concerns the interplay between social and technical findings. Seemingly small nuances in system design can lead to radically different impressions of use and of user satisfaction. These design differences may lead to different findings. While I believe I included important technical features for an organizational memory system, the required technical commonalties for such systems will not be clearly known until many systems and applications are built and field tested.

8.5. Future work

This thesis has suggested many avenues for future investigation. In this section, I consider only three. First, further studies are needed to replicate and extend the findings in this thesis. Future studies could be carried out with different user populations to determine whether the same findings extend to groups with other career anchors or in situations with differing kinds of information databases. A particularly important question is to determine whether users will continue with their concern with "bothering" the experts, and if so, to determine the causes.

Second, additional work could extend the Answer Garden Substrate's technology in several directions. Clearly, the AG suite of tools for authoring and publishing could be supplemented and extended. This will be necessary to get others to use the tool independently.

As well, a new set of Internet applications has recently gained popularity. The Wide Area Information Server (WAIS), Wide World Web (WWW), and Gopher are now de-facto standards. NCSA Mosaic, which incorporates all three standards as well as adding hypertext annotations and other features, is also widely used. If I wish to obtain sites without my offering substantial help (and much begging), I will need to either incorporate these standards into Answer Garden or, more likely, Answer Garden into them.

Moreover, several technical findings in this thesis suggested that the Answer Garden architecture could be more flexible. Another direction for future work is to offer a technical prototype for a flexible system to incorporate both information retrieval and communication facilities. Such a system is currently being designed.

Finally, to return to the Introduction of this thesis, I alluded to some additional goals for this project:

- To determine whether social issues must be considered in at least some types of technical design.
- To provide better mechanisms for helping software engineers stay current and learn new skills.
- To gain a greater understanding of an under explored area of organizational life, namely organizational memory.

While some progress was made, more work is needed for each of these goals. For example, while I believe there exists the basis for understanding the basic structures and functions of organizational memory and organizational memory systems, there are many more relationships to be considered. For example, the relationship of organizational memory to control structures has not been mapped out. One might expect that memory in hierarchically based organizations to be different in networked organizations. Similarly, the effect of organizational or national culture on organizational memory remains to be determined. Studies of Answer Garden, studies of similar organizational memory applications using other technical systems, and studies of organizational memory in general will be required.

8.6. One last word...

This thesis had as a premise, the iterative cycle among social understanding, design of technical artifacts, and field evaluation. Such evaluations should help further social understanding. I hope the results of this thesis suggest that socially oriented technical design, with the concomitant studies, can serve as a valid method of examining the relationship between new CSCW artifacts and potential work conditions.

References

Aaronson, Amy, and John M. Carroll. 1987a. The answer is in the question: a protocol study of intelligent help. *Behaviour and Information Technology*, 6 (4) : 393-402.

Aaronson, Amy, and John M. Carroll. 1987b. Intelligent Help in a One-Shot Dialog: A Protocol Study. *Proceedings of ACM Human Factors in Computing Systems and Graphics Interface (CHI + GI) '87* : 163-168.

Aaronson, Amy, and John M. Carroll. 1987c. Understanding Intelligent Help: A Protocol Analysis of Advisory Interactions. User Interface Institute, IBM T. J. Watson Research Center. RC 13059.

Acevedo, Raul. 1991. Emacs OLCR User's Manual Version 3.3. MIT Information Services manual.

Ackerman, Mark S. 1992. The Answer Garden Substrate: A Platform for Building Organizational Memory Applications. *Proceedings of Workshop on Information Technologies & Systems (WITS 92)* : 104-113.

Ackerman, Mark S. 1992. Answer Garden and the Organization of Expertise. Massachusetts Institute of Technology, Center for Coordination Studies, Technical Report 127.

Ackerman, Mark S., and Thomas W. Malone. 1990. Answer Garden: A Tool for Growing Organizational Memory. *Proceedings of ACM Conference on Office Information Systems* : 31-39.

Adelson, Beth. 1984. When Novices Surpass Experts: The Difficulty of a Task May Increase With Expertise. *Journal of Experimental Psychology*, 10 (3) : 483-495.

Adelson, Beth, and Elliot Soloway. 1985. The Role of Domain Experience in Software Design. *IEEE Transactions on Software Engineering*, SE-11 (11) : 1351-1360.

Advertel Communication Systems, Inc. 1988. The Beginner's Guide to Confer II. System manual.

Allen, Thomas. 1977. *Managing the Flow of Technology*. Cambridge, MA: MIT Press.

Allen, Thomas J. 1969. Information Needs and Uses. In *Annual Review of Information Science and Technology*. Edited by C. A. Cuadra and A. W. Luke. 3-30. Chicago: Encyclopaedia Britannica, Inc.

Anderson, Elijah. 1978. *A Place on the Corner*. Chicago: University of Chicago Press.

Anick, Peter G., Rex A. Flynn, and David R. Hanssen. 1991. Addressing the Requirements of a Dynamic Corporate Textual Information Base. *Proceedings of ACM/SIGIR Conference on Research and Development in Information Retrieval* : 163-172.

Bartschi, Martin. 1985. An Overview of Information Retrieval Subjects. *IEEE Computer*, May 1985 : 67-84.

Bearman, David. 1987. Collecting Software: A New Challenge for Archives & Museums. Archival Informatics Newsletter & Technical Report. Volume 1, #2.

Becker, Howard. 1986. *Doing Things Together: Selected Papers*. Evanston, IL: Northwestern University Press.

Berlin, Lucy M., Robin Jeffries, Vicki L. O'Day, Andreas Paepke, and Cathleen Wharton. 1993. Where Did You Put It? Issues in the Design and Use of a Group Memory. Hewlett-Packard Software Technology Laboratory. Hewlett-Packard Software Technology Laboratory report #HPL-93-11.

Blair, David C., and M. E. Maron. 1990. Full-Text Information Retrieval: Further Analysis and Clarification. *Information Processing & Management*, 26 (3) : 437-447.

Blau, Peter M. 1955. *The Dynamics of Bureaucracy: A Study of Interpersonal Relations in Two Government Agencies*. Chicago: University of Chicago Press.

Blau, Peter M., and Marshall W. Meyer. 1987. *Bureaucracy in Modern Society*. New York: Random House.

Borenstein, Nathaniel. 1985. The Design and Evaluation of On-Line Help Systems. Ph.D. dissertation, Carnegie-Mellon.

Borgman, Christine L., Andrea L. Gallagher, David Kreiger, and James Bower. 1990. Children's Use of an Interactive Catalog of Science Materials. *Proceedings of Proceedings of the 53rd American Society for Information Science Annual Meeting* : 55-68.

Bostrom, Robert, and J. Stephen Heinen. 1977. MIS Problems and Failures: A Socio-Technical Perspective. *MIS Quarterly*, 1 (3) : 17-32.

Bourdieu, Pierre. 1990. *The Logic of Practice*. Translated by Richard Nice. Stanford, CA: Stanford University Press.

Braudel, Fernand. 1980. *On History*. Chicago: University of Chicago Press.

Brissenden, Roger J. V. 1992. AXAF User Interfaces at SAO. *Proceedings of Workshop on User Interfaces for Astrophysical Software* , manuscript version.

Burke, Peter. 1991. *New Perspectives on Historical Writing*. University Park, PA: The Pennsylvania State University Press.

Callon, Michel. 1987. Society in the Making: The Study of Technology as a Tool for Sociological Analysis. In *The Social Construction of Technological Systems*. Edited by T. P. H. Wiebe E. Bijker and Trevor Pinch. 83-106. Cambridge, MA: MIT Press.

Cammarata, Stephanie, David McArthur, and Randall Steeb. 1988. Strategies of Cooperation in Distributed Problem Solving. In *Readings in Distributed Artificial*

Intelligence. Edited by A. H. Bond and L. Gasser. 102-105. Palo Alto, CA: Morgan Kaufmann Publishers, Inc.

Campagnoni, F. R., and Kate Ehrlich. 1989. Information Retrieval Using a Hypertext-Based Help System. *Proceedings of SIGIR '89* : 212-220.

Carr, Edward H. 1961. *What Is History?* New York: Vintage Books.

Carroll, John M., and Jean McKendree. 1987. Interface Design Issues for Advice-Giving Expert Systems. *Communications of the ACM*, 30 (1) : 14-31.

Chalmers, A. M. 1989. Case Study: Records Management in a Pharmaceutical Environment. *Records Management Journal*, 1 (1) : 19-28.

Chapanis, Alphonse. 1975. Interactive Human Communication. *Scientific American*, 232 : 36-42.

Chen, Ching-chih. 1982. *Information seeking : assessing and anticipating user needs.* New York, NY: Neal-Schuman Publishers.

Cicourel, Aaron. 1990. The Integration of Distributed Knowledge in Collaborative Medical Diagnosis. In *Intellectual Teamwork: Social and Technological Foundaitons of Cooperative Work.* Edited by J. Galegher and R. Kraut. 221-242. Hillsdale, NJ: Lawrence Erlbaum.

Cohen, Jacob. 1977. *Statistical Power Analysis for the Behavioral Sciences.* New York: Academic Press.

Cohen, Michael D., James G. March, and Johan P. Olsen. 1972. A Garbage Can Model of Organizational Choice. Administrative Science Quarterly, 17 (1) : 1-25.

Collingwood, R. G. 1946. *The Idea of History.* New York: Oxford University Press.

Conklin, Jeff. 1992. Corporate Memory. *Proceedings of Groupware '92*: 133-137.

Conklin, Jeff, and Michael L. Begeman. 1988. gIBIS: A Hypertext Tool for Exploratory Policy Discussion. *Proceedings of CSCW '88* : 140-152.

Constant, David, Lee Sproull, and Sara Kiesler. 1991. Discretionary Information Sharing in a Networked Organization. Carnegie Mellon University, Department of Social and Decision Sciences. Manuscript.

Coppeto, Thomas J., Beth L. Anderson, and Daniel E. Geer Jr. 1989. OLC: An On-Line Consulting System for UNIX. *Proceedings of USENIX Winter 1989*, manuscript version.

Crane, Donald P. 1979. *Personnel: The Management of Human Resources.* Belmont, California: Wadsworth Publishing Company.

Crawford, Susan. 1978. Information Needs and Uses. In *Annual Review of Information Science and Technology.* Edited by M. E. Williams. 61-82. White Plains, NY: American Society for Information Science.

Croft, W. B., R. Krovetz, and H. Turtle. 1990. Interactive Retrieval of Complex Documents. *Information Processing & Management*, 26 (5) : 593-613.

Cruth, Donald L., Robert M. Noe III, and R. Wayne Mondy. 1988. *Staffing the Contemporary Organization: A Guide to Planning, Recruiting, and Selecting for Human Resource Professionals*. New York: Quorum Books.

Curtis, Bill, Marc I. Kellner, and Jim Over. 1992. Process Modeling. *Communications of the ACM*, 35 (9) : 75-90.

Davis, Fred D. 1989. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly*, 13 (3) : 319-342.

DellaFera, C. Anthony, Mark W. Eichin, Robert S. French, David C. Jedlinsky, John T. Kohl, and William E. Sommerfeld. 1988. The Zephyr Notification Service. *Proceedings of Winter 1988 Usenix Technical Conference* : 213-220.

Dervin, Brenda, and Michael Nilan. 1986. Information Needs and Uses. In *Annual Review of Information Science and Technology*. Edited by M. E. Williams. 3-34. White Plains, NY: American Society for Information Science.

Dessler, Gary. 1984. *Personnel Management: Modern Concepts & Techniques*. Reston, Virginia: Reston Publishing Company.

Doane, Stephane M., James Pellegrino, and Roberta L. Klatzky. 1990. Expertise in a Computer Operating System: Conceptualization and Performance. *Human-Computer Interaction*, 5 : 267-304.

Dourish, Paul, Victoria Bellotti, Wendy Mackay, and Chao-Ying Ma. 1993. Information and Context: Lessons from a Study of Two Shared Information Systems. Rank Xerox EuroPARC. Rank Xerox EuroPARC Technical Report EPC-92-124.

Draper, Stephen W. 1985. The Nature of Expertise in Unix. *Proceedings of Human-Computer Interaction (INTERACT '84)* : 465-471.

Dumais, Susan T., George W. Furnas, Thomas K. Landauer, Scott Deerwester, and Richard Harshman. 1988. Using Latent Semantic Analysis to Improve Access to Textual Information. *Proceedings of ACM CHI'88 Human Factors in Computing Systems* : 281-285.

Duncan, Robert, and Andrew Weiss. 1979. Organizational Learning: Implications for Organizational Design. In *Research in Organizational Behavior*. 75-123. JAI Press.

Edmonds, Mark Lansdale and Ernest. 1992. Using memory for events in the design of personal filing systems. *International Journal of Man-Machine Studies*, 36 : 97-126.

Egan, Dennis E., Joel R. Remde, Thomas K. Landauer, Carol C. Lochbaum, and Louis M. Gomez. 1989. Behavioral Evaluation and Analysis of a Hypertext Browser. *Proceedings of ACM CHI'89 Human Factors in Computing Systems* : 205-210.

Essick, Raymond B., Bob Kolstad, Rick L. Spickelmier, and Lou Salkind. 1983. Notesfile Reference Manual. University of Illinois. University of Illinois technical report UIUCDCS-R-82-1081.

Feldman, Martha S. 1987. Electronic Mail and Weak Ties in Organizations. *Office, Technology, and People*, 3 : 83-101.

Filstead, William J. 1970. *Qualitative Methodology: Firsthand Involvement with the Social World.* Chicago: Markham Publishing Company.

Finholt, Tom, and Lee S. Sproull. 1990. Electronic Groups at Work. *Organization Science*, 1 (1) : 41-64.

Finholt, Thomas A. 1990. Accessing Organizational Expertise through Computer-Mediated Communication. Carnegie-Mellon University. Thesis proposal.

Fischer, Gerhard, and Andreas C. Lemke. 1988. Construction Kits and Design Environments: Steps Toward Human Problem-Domain Communication. *Human-Computer Interaction*, 3 (3) : 179-222.

Foss, C. 1988. Effective Browsing in Hypertext Systems. *Proceedings of RIAO '88* : 16-23.

Galbraith, Jay. 1973. *Designing Complex Organizations.* Reading, MA: Addison-Wesley.

Gerstberger, Peter G., and Thomas J. Allen. 1968. Criteria Used by Research and Development Engineers in the Selection of an Information Source. *Journal of Applied Psychology*, 52 (4) : 272-279.

Goffman, Erving. 1959. *The Presentation of Self in Everyday Life.* New York: Doubleday Anchor.

Gorry, G. Anthony, Andrew M. Burger, R. Jesse Chaney, Kevin B. Long, and Christina M. Tausk. 1988. Computer Support for Biomedical Work Groups. *Proceedings of ACM Conference on Computer Supported Cooperative Work (CSCW '88)* : 39-51.

Graham, M. B. W. 1992. Notes on Organizational Memory: Practice and Theory. Xerox Palo Alto Research Center. Manuscript.

Grudin, Jonathan. 1989. Why groupware applications fail: problems in design and evaluation. *Office,Tehnology ,and People*, 4 (3) : 245-264.

Guidon, Raymonde. 1990. Knowledge Exploited by Experts During Software System Design. *International Journal of Man-Machine Studies*, 33 : 279-304.

Gwei, Godwin W., and Eric Foxley. 1990 Towards a Consultative On-Line Help System. *International Journal of Man-Machine Studies*, 32 : 363-383.

Hammond, Nick, and Lesley Allinson. 1988. Travels around a Learning Support Environment: Rambling, Orienteering, or Touring? *Proceedings of ACM Conference on Human Factors in Computing Systems (HI '88)*: 269-273.

Hauptman, Oscar. 1986. Managing Software Development: Communication as a Success Factor. Ph.D. Thesis, Massachusetts Institute of Technology, Sloan School of Management.

Hewins, Elizabeth T. 1990. Information Need and Use Studies. In *Annual Review of Information Science and Technology*. Edited by M. E. Williams. 145-172. New York: American Society of Information Science.

Hiltz, Starr Roxanne, and Murray Turoff. 1981. The Evolution of User Behavior in a Computerized Conferencing System. *Communications of the ACM*, 24 (11) : 739-762.

Horton, M., and R. Adams. 1987. Standard for Interchange of USENET Messages. Network Working Group RFC-1036.

Huber, George P. 1990. A Theory of the Effects of Advanced Information Technologies on Organizational Design, Intelligence, and Decision Making. *Academy of Management Review*, 15 (1) : 47-71.

Huber, George P. 1991. Organizational Learning: The Contributing Processes and the Literatures. *Organizational Science*, 2 (1) : 88-115.

Jick, Todd D. 1979. Mixing Qualitative and Quantitative Methods: Trigulation in Action. *Administrative Science Quarterly*, : 602-611.

Jones, William P. 1986. The Memory Extender Personal Filing System. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '86)* : 298-305.

Kass, Robert, and Irene Stadnyk. 1991. Intelligent Assistance for the Communication of Information in Large Organizations. Center for Advanced Research. Center for Advanced Research CMI-91-011.

Kirk, Jerome, and Marc L. Miller. 1986. *Reliability and Validity in Qualitative Research.* Qualitative Research Methods Series. Beverly Hills, CA: Sage.

Korfhage, Robert R. 1984. Query Enhancement by User Profiles. *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR '84)* : 111-121.

Korfhage, Robert R. 1991. To See, or Not to See: Is that the Query? *Proceedings of Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval (SIGIR '91)* : 134-141.

Kraft, Donald H. 1985. Advances in Information Retrieval: Where Is That /#*&@ Record? In *Advances in Computers*. Edited by M. Yovits. 277-318. New York: Academic Press.

Kraut, Robert E., and Lynn A. Streeter. 1990. Satisfying the Need to Know: Interpersonal Information Access. *Proceedings of Conference on Human-Computer Interaction (INTERACT '90)* : 909-915.

Kwasnik, Barbara H. 1989. How a Personal Document's Intended Use or Purpose Affects Its Classification in an Office. *Proceedings of Twelfth Annual International ACM*

SIGIR Conference on Research and Development in Information Retrieval : 207-210.

Lai, Kum-Yew, Thomas Malone, and Keh-Chiang Yu. 1988. Object Lens: A "Spreadsheet" for Cooperative Work. *ACM Transactions on Information Systems*, 6 (4) : 332-353.

Lamming, Michael G., and William M. Newman. 1992. Activity-based Information Retrieval: Technology in Support of Personal Memory. EuroPARC Report EPC 92-002.

Lancaster, F. W., and E. G. Fayen. 1973. *Information Retrieval On-Line*. Los Angeles: Melville Publishing Company.

Lansdale, M. 1988. The psychology of personal information management. *Applied Ergonomics*, 19 (1) : 55-66.

Lee, Jintae. 1990. SIBYL: A Qualitative Decision Management System. In *Artificial Intelligence at MIT: Expanding Frontiers*. Edited by P. Winston and S. Shellard. Chapter 4. Cambridge: MIT Press.

Lesk, Michael. 1991. SIGIR 91: The More Things Change, the More They Stay the Same. *SIGIR Forum*, 25 (2) : 4-7.

Lesser, Victor R., and Lee D. Erman. 1988. Distributed Interpretation: A Model and Experiment. In *Readings in Distributed Artificial Intelligence*. Edited by A. H. Bond and L. Gasser. 120-139. Palo Alto, CA: Morgan Kaufmann.

Levitt, Barbara, and James G. March. 1988. Organizational Learning. *Annual Review of Sociology 1988*, 14 : 319-340.

Lipsey, Mark W. 1990. *Design Sensitivity: Statistical Power for Experimental Research*. Newbury Park, CA: Sage.

Lotus Development Corporation. 1991. Lotus Notes. Manuals, release 2.0. Various.

Malone, Thomas W. 1985. Designing Organizational Interfaces. *Proceedings of ACM CHI'85 Conference on Human Factors in Computing Systems*, 66-71.

Malone, Thomas W., and Kevin Crowston. 1992. Toward an Interdisciplinary Theory of Coordination. MIT Center for Coordination Science technical report #120.

Malone, Thomas W., Kevin Crowston, Jintae Lee, and Brian Pentland. 1992. Tools for inventing organizations: Toward a handbook of organizational processes. MIT Center for Coordination Science, manuscript.

Malone, Thomas W., Kenneth R. Grant, Kum-Yew Lai, Ramana Rao, and David Rosenblitt. 1988. Semi-structured Messages are Surprisingly Useful for Computer-Supported Coordination. In *Computer-Supported Cooperative Work: A Book of Readings*. Edited by I. Greif. 311-331. San Mateo, CA: Morgan Kaufmann.

Malone, Thomas W., Ken R. Grant, Franklyn A. Turbak, Stephen A. Brobst, and Michael D. Cohen. 1987. Intelligent Information Sharing Systems. *Communications of the ACM*, 30 (5, May 1987) : 390-402.

Malone, Thomas W., Kum-Yew Lai, and Christopher Fry. 1992. Experiments with Oval: A Radically Tailorable Tool for Cooperative Work. Massachusetts Institute of Technology, Center for Coordination Science, technical report #132.

Mandel, Eric, John Roll, Stephen S. Murray, and Mark S. Ackerman. 1992. AXAF User Interfaces for Heterogeneous Analysis Environments. *Proceedings of the Conference on Astronomy from Large Databases - II* .

March, James. 1990. How Decisions Happen in Organizations. Conference on Computer Supported Cooperative Work '90. Manuscript.

Marchionini, Gary, and Ben Shneiderman. 1988. Finding Facts vs. Browsing Knowledge in Hypertext Systems. *International Journal of Man-Machine Studies*, 21 (1) : 70-79.

Marcus, Richard S. 1988. Expert Retrieval Assistance Development and Experimentation. *Proceedings of the American Society of Information Science Annual Meeting (ASIS '88)* : 115-119.

Marshall, Catherine C., and Peggy M. Irish. 1989. Guided Tours and On-Line Presentations: How Authors Make Existing Hypertext Intelligible for Readers. *Proceedings of Hypertext '89* : 15-26.

Miller, Vernon D., and Fredric M. Jablin. 1991. Information Seeking during Organizational Entry: Influences, Tactics, and a Model of the Process. *The Academy of Management Review*, 16 (1) : 92-120.

MMB Development Corporation. 1991. TEAMate User Guide. Manual, release 2.1.

Morgan, Howard L., and David J. Root. 1979. A Concept of Organizational Memory. *Proceedings of the Office Automation Conference*: 31-36. Also University of Pennsylvania Wharton School Working Paper 79-04-03.

Morrison, Joline. 1991. Development and Evaluation of a Model for Team and Organizational Memory. University of Arizona. Unpublished dissertation proposal.

Mumford, Enid, and Don Henshall. 1979. *A participative approach to computer systems design: a case study of the introduction of a new computer system.* London: Associated Business Press.

Nielsen, Jakob. 1992. Finding Usability Problems through Heuristic Evaluation. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '92)* : 373-380.

Noll, John, and Walt Scacchi. 1991. Integrating Diverse Information Repositories: A Distributed Hypertext Approach. *IEEE Computer*, 24 (12) : 38-45.

Orlikowski, Wanda J. 1992. Learning from Notes: Organizational Issues in Groupware Implementation. *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW) '92* : 362-369

Orr, Julian. 1986. Narratives at Work: Storytelling as Cooperative Diagnostic Activity. *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW 86)*: 62-72.

Pentland, Brian T. 1991. Making The Right Moves: Toward a Social Grammar of Software Support Hot Lines. Ph.D. Thesis, Massachusetts Institute of Technology.

Pentland, Brian T. 1992. Organizing Moves in Software Support Hot Lines. *Administrative Science Quarterly*, 37 : 527-548.

Pickering, Jeanne M., and John Leslie King. 1992. Hardwiring Weak Ties: Individual and Institutional Issues in Computer Mediated Communication. *Proceedings of ACM 1992 Conference on Computer-Supported Cooperative Work (CSCW 92)*: 356-361.

Pinelli, Thomas E. 1991. The Information-Seeking Habits and Practices of Engineers. *Science & Technology Libraries*, 11 (3) : 5-25.

Pinelli, Thomas E., John M. Kennedy, and Rebecca O. Barclay. 1990. The Role of the Information Intemediary in the Diffusion of Aerospace Knowledge. *Science & Technology Libraries*, 11 (2) : 59-76.

Pollack, Martha E. 1985. Information Sought and Information Provided: An Empirical Study of User/Expert Dialogues. *Proceedings of the Conference on Human Factors in Computing Systems (ACM CHI'85)* : 155-159.

Raeburn, Ken, Jon Rochlis, William Sommerfeld, and Stan Zanarotti. 1989. Discuss: An Electronic Conferencing System for a Distributed Computing Environment. *Proceedings of USENIX Winter 1989* : 331-342.

Ram, Sudha, Stephen Hayne, David Carlson, and Jr. Jay F. Nanamaker. 1989. Integration of Information Systems Technologies to Support Consultation in an Information Center. *Proceedings of International Conference on Information Systems (ICIS 89)* : 67-77.

Ram, Sudha, Stephen Hayne, and David Carlson. 1992. Integrating Information Systems Technologies to Support Consultation in an Information Center. *Information & Management*, 23 : 331-343.

Remde, Joel R., Louis M. Gomez, and Thomas K. Landauer. 1987. SuperBook: An Automatic Tool for Information Exploration-- Hypertext? *Proceedings of Hypertext '87* : 175-187.

Robek, Mary F., Gerald F. Brown, and Wilmer O. Maedke. 1987. *Information and Record Management*. Encino, CA: Glencoe Publishing.

Rockart, John F., and Jim E. Short. 1989. IT and the Networked Organization: Towards More Effective Management of Interdependence. Center for Information Systems

Research, Massachusetts Institute of Technology. MIT Center for Information Systems Research Working Paper no. 200.

Rosenberg, Victor. 1967. Factors Affecting the Preferences of Industrial Personnel for Information Gathering Methods. *Information Storage and Retrieval*, 3 : 119-127.

Rosenbloom, Richard S., and Francis W. Wolek. 1970. *Technology and Information Transfer: A Survey of Practice in Industrial Organizations*. Boston: Division of Research, Graduate School of Business, Harvard University.

Sackmann, Sonja A. 1992. Culture and Subcultures: An Analysis of Organizational Knowledge. *American Sociological Quarterly*, 37 (1) : 140-161.

Salton, Gerard. 1989. *Automatic Text Processing*. Reading, MA: Addison-Wesley.

Salton, Gerald, and M. H. McGill. 1983. *Introduction to Modern Information Retrieval*. New York: McGraw-Hill.

Saltzer, Jerome. 1991a. Fault-Tolerance in Very Large Archive Systems. *Operating Systems Review*, 25 (1) : 81-82.

Saltzer, Jerome. 1991b. File System Indexing, and Backup. In *Operating Systems for the 90's and Beyond, Lecture Notes in Computer Science 563*. Edited by A. Karshmer and J. Nehmer. 13-19. New York: Springer-Verlag.

Saracevic, Tefko. 1970. *Introduction to Information Science*. New York: R. R. Bowker Company.

Scheifler, Robert W., and Jim Gettys. 1987. The X Window System. *ACM Transactions on Graphics*, 5 (2) : 79-109.

Schein, Edgar H. 1978. *Career Dynamics*. Reading, MA: Addison-Wesley.

Schwartz, Howard, and Jerry Jacobs. 1979. *Qualitative Sociology: A Method to the Madness*. New York: Free Press.

Scott, W. Richard. 1992. *Organizations: Rational, Natural, and Open Systems*. Englewood Cliffs, NJ: Prentice Hall.

Sebrechts, Marc M., and Merryanna L. Swartz. 1991. Question Asking as a Tool for Novice Computer Skill Acquisition. *Proceedings of ACM Human Factors in Computing Systems Conference (CHI'91)* : 293-299.

Shneiderman, Ben. 1987. User Interface Design for the Hyperties Electronic Encyclopedia. *Proceedings of Hypertext '87* : 189-194.

Sproull, Lee, and Sara Kiesler. 1991. *Connections: New Ways of Working in the Networked Organization*. Cambridge, MA: MIT Press.

Stewart, Thomas A. 1991. BrainPower. *Fortune*, June 3, 1991, 44-60.

Strauss, Anselm L. 1987. *Qualitative Analysis for Social Scientists*. New York: Cambridge University Press.

Streeter, Lynn A., and Karen E. Lochbaum. 1988. Who Knows: A System Based on Automatic Representation of Semantic Structure. *Proceedings of RIAO 88* : 380-388.

Suchman, Lucy, and Eleanor Wynn. 1984. Procedures and Problems in the Office. *Office: Technology and People*, 2 : 133-154.

Swick, Ralph R., and Mark S. Ackerman. 1988. The X Toolkit: More Bricks for Building User-Interfaces. *Proceedings of Winter 1988 Usenix* : 5-17.

Trigg, Randall H. 1988. Tools for Communicating in a Hypertext Environment. *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW 88)*: 216-226.

Utting, Kenneth, and Nicole Yankelovich. 1989. Context and Orientation in Hypermedia Networks. *ACM Transactions on Information Systems*, 7 (1) : 58-84.

Van Maanen, John, James M. Dabbs Jr., and Robert R. Faulkner. 1982. *Varieties of Qualitative Research*. Studying Organizations: Innovations in Methodology #5. Beverly Hills, CA: Sage.

Walch, Victoria Irons. 1990. The Role of Standards in the Archival Management of Electronic Records. *American Archivist*, Winter 1990 : 30-43.

Walker, Janet H. 1987. Document Examiner: Delivery Interface for Hypertext Documents. *Proceedings of Hypertext '87* : 307-324.

Walsh, James P., and Gerardo Rivera Ungson. 1991. Organizational Memory. *The Academy of Management Review*, 16 (1) : 57-91.

Webster's. 1986. *Webster's Ninth New Collegiate Dictionary*. Springfield, MA: Merriam-Webster.

Weick, Karl E. 1979. *The Social Psychology of Organizing*. New York: Random House.

Wyle, M F, and H P Frei. 1989. Retrieving Highly Dynamic, Widely Distributed Information. *Proceedings of Twelfth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* : 108-115.

Yates, JoAnne. 1988. Creating Organizational Memory: Systematic Management and Internal Communication in Manufacturing Firms, 1880-1920. Sloan School of Management, Massachusetts Institute of Technology, Working Paper #2006-88.

Yates, JoAnne. 1989. *Control through Communication: The Rise of System in American Management*. Baltimore: John Hopkins Press.

Yates, JoAnne. 1990. For the Record: The Embodiment of Organizational Memory, 1850-1920. *Business and Economic History*, 19 : 172-182.

Yates, Joanne, and Wanda J. Orlikowski. 1992. Genres of Organizational Communication: A Structurational Approach to Studying Communication and Media. *Academy of Management Review*, 17 (2) : 299-326.

Zellweger, Polle T. 1988. Active Paths Through Multimedia Documents. In *Document Manipulation and Typography*. Edited by J. C. v. Vliet. Cambridge University Press.

Appendix A: Single User IR Model

Below is a brief explication of each box in Figure 3.2 (replicated in Figure A.1 on the next page). The references provided are representative of the categories, and are not intended to be exhaustive. If not noted, reference is to the surveys in Salton and McGill 1983, Salton 1989, Kraft 1985, and Bartschi 1985.

1. Query formulation. An important issue in IR research is the expression of a user's information need. The query process, either directly or indirectly, is critical to why users retrieve information. While many authors include other processing when speaking of the "query" process, I use it in the diagram to denote only the user's formulation of his query statement.

   There are two largely separate research streams concerned with the query step. Many cognitive studies concern themselves with information seeking, or why users come to systems with questions. Technically, efforts have been focused on making it easier on the user to ask his query. This research includes

   - User interface work for creating more intuitive interfaces for casual users. This ranges from interfaces for children (Borgman et al. 1990) through questions of information presentation (Remde, Gomez, and Landauer 1987, Walker 1987, Campagnoni and Ehrlich 1989) to issues of navigation through the information space (Egan et al. 1989, Marshall and Irish 1989, Trigg 1988, Foss 1988, Zellweger 1988, Shneiderman 1987).

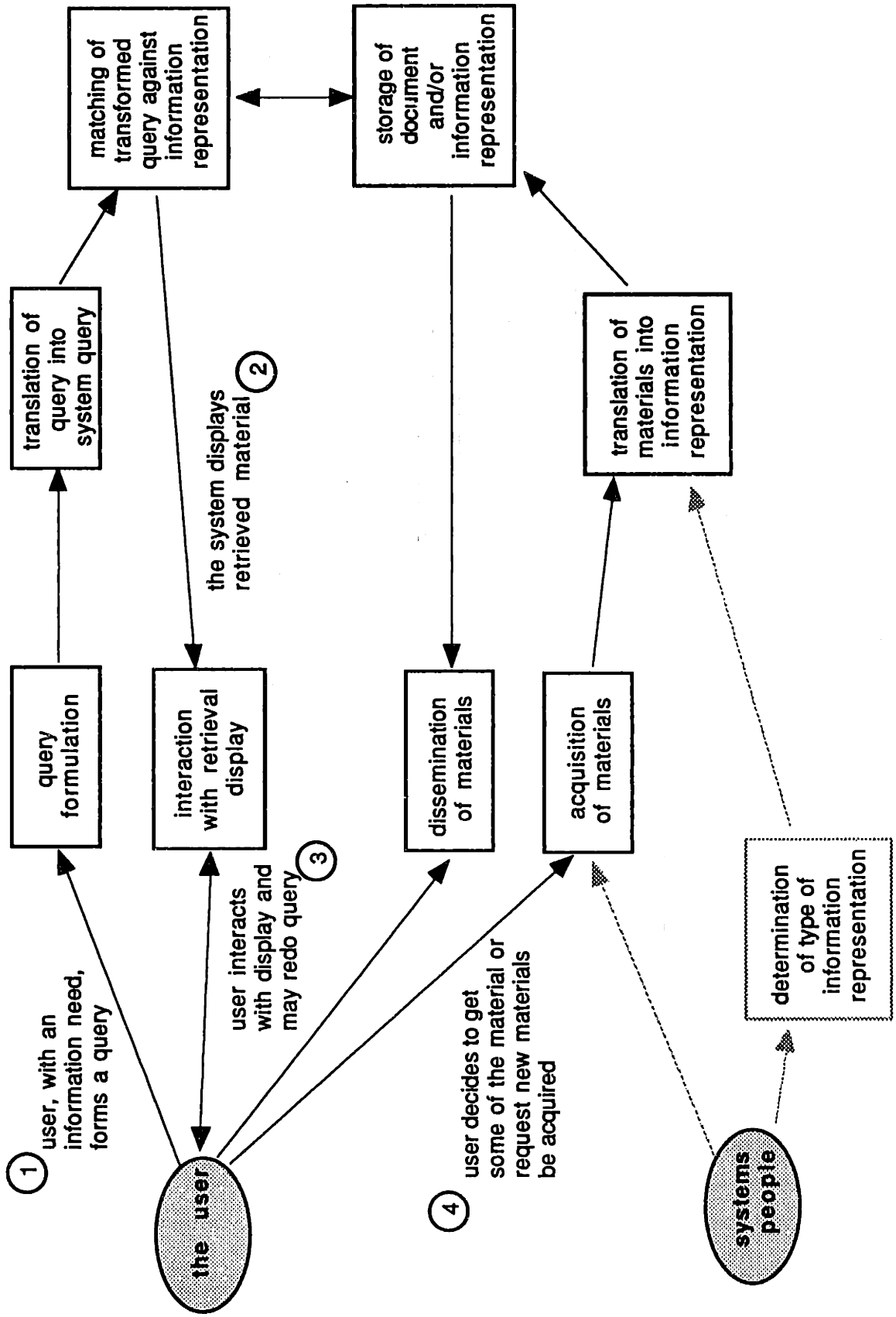

Figure A.1: A user-centered model of information retrieval (figure 3.2)

A recent British stream of research has been open queries to additional criteria including personal events (Lamming and Newman 1992, Edmonds 1992).

- Natural language processing for allowing the user to specify queries in his own language.

There is, in addition, another stream of research that combines the display of the information space (or retrieved information space) with query formulation. I will discuss visualization techniques, and their older counterpart, relevance feedback, below. In this stream of research, the display and the query steps are combined.

2. Query translation and mediation. The user query, once it has been formulated in whatever user interface form is allowed, must then be translated into a format that the system can match against the database representation. Research has been conducted on intelligent intermediaries for this step. These include:

   - Intelligent agents and filters to query and sift future streams of information (Malone et al. 1987).

   - Expert systems that embed search strategies (Marcus 1988).

   - Systems that handle much of the extraneous detail of specific information sources.

   - Systems that combine data from several sources to provide interconnected information.

3. Information representation. In figure 2.2, this block is in the lower right hand corner of the diagram since it is not a user-centered activity. However, it is logically necessary to consider information representation before discussing how the user query is matched.

This topic has garnered over 30 years of research. At the risk of simplifying, one can say that there are two fundamental problems that have been addressed. The first is one of physical storage. Until recently, it has been too expensive to place the actual information corpus on-line, and as a result, surrogates (abstracts, keyword lists, or the like) have been used. The second is more fundamental: it assumed that the set of words in a document map to some conceptual structure, but necessarily imperfectly. Therefore, it is difficult to determine whether any given document is on a particular topic, and it is even more difficult to determine whether it satisfies a user's query. Therefore, considerable effort has been devoted to

- Constructing representation schema that can be used for evaluating queries against the set of documents as represented. This research stream will be discussed below in the section on query evaluation.

- Determining methods for automatic classifying documents (to avoid hand indexing). A critical assumption is that "one can determine subject content by analysis of the words in the text. (Kraft, p. 289)" This research stream is sometimes called content analysis, although most research is concerned with creating statistical methods for analyzing word frequencies. This research

stream has come under sharp attack in Blair and Maron (Blair and Maron 1990).

- Formulating conceptual structures to tie together sets of documents. This allows users to search using concepts or topics, allowing the user to ignore the precise mapping between representation schemes and documents. Classification theory research examines the use of such techniques as indexing hierarchies, thesauri, and semantic networks.

4. Matching queries with the information representation. Considerable effort has been expended to better match a user query with the information representation of the document database. In effect, it is assumed that the user query is an inexact expression of the user's information need and that the information representation is an inexact expression of the concepts in the information database. The problem, then, is to match the inexact expression of the query with the inexact expression of the database in such a way that the user is satisfied with the result.

   There are, roughly speaking, three types of matching: term matching, structural retrieval, and semantic analysis. Any of these may be tied to adaptive retrieval based on the context of the user.

   - Semantic analysis includes natural language understanding and, beyond that, discourse analysis. This is the holy grail of information retrieval.

   - Structural retrieval allows users to retrieve information on the basis of additional structure in the information object including

additional fields (Malone et al. 1987, Malone et al. 1988) or document structures.(Croft, Krovetz, and Turtle 1990)

- Term matching includes a wide variety of technologies with vigorous debate among the proponents of each. This approach attempts to match a set of query terms with some set of terms in the collection. This approach has evolved over many years, and different approaches have been developed and combined. The proposed solutions include:

  a. The ever-popular Boolean retrieval. This may be keyword or full-text retrieval.

  b. Vector. If the document contains an index term, the vector representing the document has the corresponding element "turned on." Vectors representing documents and query can then be measured for similarity. Additionally, vectors of documents can be compared to discover clusters of documents. There are a variety of methods in the research literature for measuring similarity.

  c. Term weighting. Both Boolean and vector retrieval can include weights on the terms to indicate both frequency and distribution of occurrence (in the information representation) and user interest (in the query). The user may be asked to indicate which terms in the query are the most relevant, or these weights may be derived from query histories. Relevance feedback uses term weighting along with term substitution. A

variety of treatments about how to use the weights and then measure the result have been proposed.

d. Statistical manipulation. A large number of techniques have been suggested. Most prominent include probabilistic retrieval and latent semantic analysis (Dumais et al. 1988, Streeter and Lochbaum 1988).

e. Cheap natural language. Some commercial products use a cheap version of natural language, accepting a natural language query and then doing more standard retrieval on terms plucked from the query. This may or may not include grammatical rules.

- Adaptive retrieval. In adaptive retrieval techniques, the system uses information about the user's past states to better rank potential retrieval items. Studies include Jones 1986 and Korfhage 1984.

5. Information storage. Considerable effort has been devoted to ascertaining storage methods for quick retrieval of the actual documents. This was an early avenue of research for full-text retrieval, and it has gained new momentum with the storage costs of multimedia information (such as images or video) (Saltzer 1991a, Saltzer 1991b, Noll and Scacchi 1991).

6. Display of the retrieval. Originally, the retrieval results were merely listed. Emphasis is now on techniques that allow the user to interact with the retrieval display. One such technique is relevance feedback: The user picks documents from the display that closely match his interests and thus

forms new queries to narrow in on his needs. Relevance feedback is now generally available in commercial software.

Some techniques currently under research allow the user to pick documents from a visualization display, usually formed on the basis of cluster analysis methods such as latent semantic (Korfhage 1991). The visualization techniques (as well as the older relevance feedback) could be considered a hybrid, including elements of query formulation and retrieval display.

Hypertext and hypermedia, to a large extent, can be considered as one technique for the display and direct manipulation of information objects. In this sense, hypertext and hypermedia systems allow the user to browse based on either hand or automatic indexing of the materials (Marchionini and Shneiderman 1988).

7. Dissemination of materials. Once the user has determined a solution to his information needs, he may wish to obtain the actual materials. As an example, consider the situation of identifying a video segment from the retrieval of an abstract. One probably wants to view the actual video segment. The dissemination step includes legal issues related to copyright and intellectual property rights; economic issues related to pricing structures and ownership; and, technical issues related to locating the materials, archiving the materials, and the translation of materials into user-required formats.

8. Acquisition of new materials. Most IR systems include relatively static collections, and are not well connected with information production systems. Some new research focuses on

- Producing agents that can seek out new information on a network. This is conceptually similar to some of the issues in (2) above.

- Tying the retrieval system into the communications network to more easily solicit new information.

Appendix B: Answer Garden Substrate (AGS) Overview

B.1. Introduction

The Answer Garden Substrate (AGS) is the underlying system beneath the Answer Garden, ASSIST, LiveDoc, and other applications.[1] It provides a set of functionality to allow these, and similar, applications to be created.

This appendix provides an in-depth overview to AGS. The following appendix, Appendix C, provides an in-depth discussion of the extensibility of AGS. These appendices include some of the material in Chapter 5; this is especially true for this appendix. However, these appendices assume a greater level of technical sophistication than does Chapter 5.

Why build an underlying system for organizational memory systems? Quite simply, it was to allow the easy construction of additional applications, some of which might deviate from the Answer Garden application slightly and some by substantial amounts. AGS is an attempt to generalize the Answer Garden functionality.

Historically, AGS was required when software engineers at the Harvard-Smithsonian Astrophysical Observatory wished to build the ASSIST application

---

[1] As with any system, care must be taken to distinguish among instances of an application (for example, the same application running at different sites), different applications built using the same underlying system, and the system itself. Additionally, there is one intermediate -- the same application using different information databases. One would expect usage to be dissimilar among applications; one might also expect usage to be dissimilar with the same application but with different databases or at different sites.

(described above). They wanted to add a substantial amount of functionality and flexibility to what was then a system that could handle only the Answer Garden requirements. I have generalized almost all of the functionality in the original Answer Garden system so as to allow many different types of organizational memory applications. This generalization into a system-level framework constitutes the AGS system.

The goals, then, of AGS were to provide the functionality to:

- ❑ Create the Answer Garden application.
- ❑ Easily enhance or tailor existing applications (especially the Answer Garden application) by hooking in additional user interface objects, information retrieval objects, communication channels, message types, and file access methods.
- ❑ Create new organizational memory applications.

New applications are created by application programmers. While a substantial amount of tailoring can be accomplished by end users and site managers, I have assumed that programmers would be required to build additional services and user interface objects.

The rest of this section contains a functional description of AGS. Broadly speaking, AGS can be divided functionally into four different levels:

1. A set of object classes that manipulate and process information and/or display that information.

   In AGS, these object classes are called *node types*, and the object instances are *nodes*. Nodes are generally visible and separately

moveable on the user's screen, although this is not necessarily the case.

When building or modifying an AGS information database, one is primarily concerned with the set of existing object classes. These existing node types and their physical representations are described below.

2. General support services for AGS applications. These include editing and common screen management among nodes. In addition, the appearance and functionality of existing applications can be tailored.

3. Facilities for adding new object classes (or node types) to the system. Applications can add new node types if they wish to extend AGS's repertoire of information objects or display capabilities. Adding new node types requires minimal understanding of AGS internals.

4. AGS also provides facilities for adding or modifying the general support services for AGS applications. These include information retrieval services and file management services, as well as other, unforeseen services such as database or multimedia services.

The description of AGS is divided into two parts. This chapter will provide an overview of AGS, providing more information about the existing node types and about customization. In the next chapter, I will describe the internals of AGS.

B.2. General comments

AGS is an X Window System toolkit application (Scheifler and Gettys 1987, Swick and Ackerman 1988). The Xt toolkit imposes a certain software architecture on its applications. Specifically, all user interactions are handled through the Xt toolkit and its objects. These Xt objects, called *widgets*, form a separate object system from that in AGS (although some of AGS functionality is provided by new widgets).[2] Communication from the Xt widget objects to the AGS objects is done through routines called *callbacks*. Callbacks are functions that are triggered by specific user interactions. These callbacks essentially form most of the user-interaction state machine for an AGS application.

B.3. Pre-existing node types: authoring an information database

The node types, which in AGS are largely synonymous with both the user interface objects and the information objects, are the most critical components of AGS. This approach is different from other systems that separate display functionality from the information contents (e.g., Lai, Malone, and Yu 1988). The combination of user interface and information resulted from the need to have functional components (such as menu items) as well as the need to have display characteristics required by the information author (such as boldface) in the display.[3]

---

[2]AGS uses the Athena (Xaw) widget set currently. I will use Xt synonomously with Xaw since these statements will also be true when (and if) AGS is ported to another Xt widget set such as Motif or OpenLook.

[3]The easiest way to do this was to allow markup languages in the initial system conception. It was my feeling that until the design requirements for such systems were understood, one should not commit the considerable effort to create WYSIWYG editors. There is nothing in AGS preventing binary formats (that is, using direct manipulation authoring and editing), and there is nothing in AGS preventing multiple

The following sections describe the existing node types as well as the information fields required of all AGS nodes.[4] In general, people wishing only to author information databases, using some existing application, need to know only about the various node types available.

The node types that currently come with AGS are the SBrowser, Grapher, QA-Node, Discussion, Code, and Ascii node-types. These were the base node types required to implement the Answer Garden application.[5]

B.3.1. SBrowsers (structured browsers)

The SBrowsers look like menus of buttons. They can also include normal, bold, and italic text, and they can serve as menus, answers, tutorials, and so on. Figure B.1 shows a SBrowser that has a menu of buttons. Figure B.2 shows one with text and a visual button; this visual button starts an external program. Note that both have both bold and normal text.

---

viewers on the same data. In fact, the Outline and the Grapher node types display different views of the same data.

[4]Readers wishing to know the specifics of authoring an information database should see the Author's Guide to Answer Garden, currently the AGS node "Info_Authoring_AG". Parts of this document overlap.

[5]The SBrowser, Grapher, and Discussion node types have existed from the beginning. QA-Nodes were added during alpha tests of Answer Garden, and Code was implemented during the field study of Answer Garden. The Ascii node type was added at the request of a research site, but was used extensively in the field study.

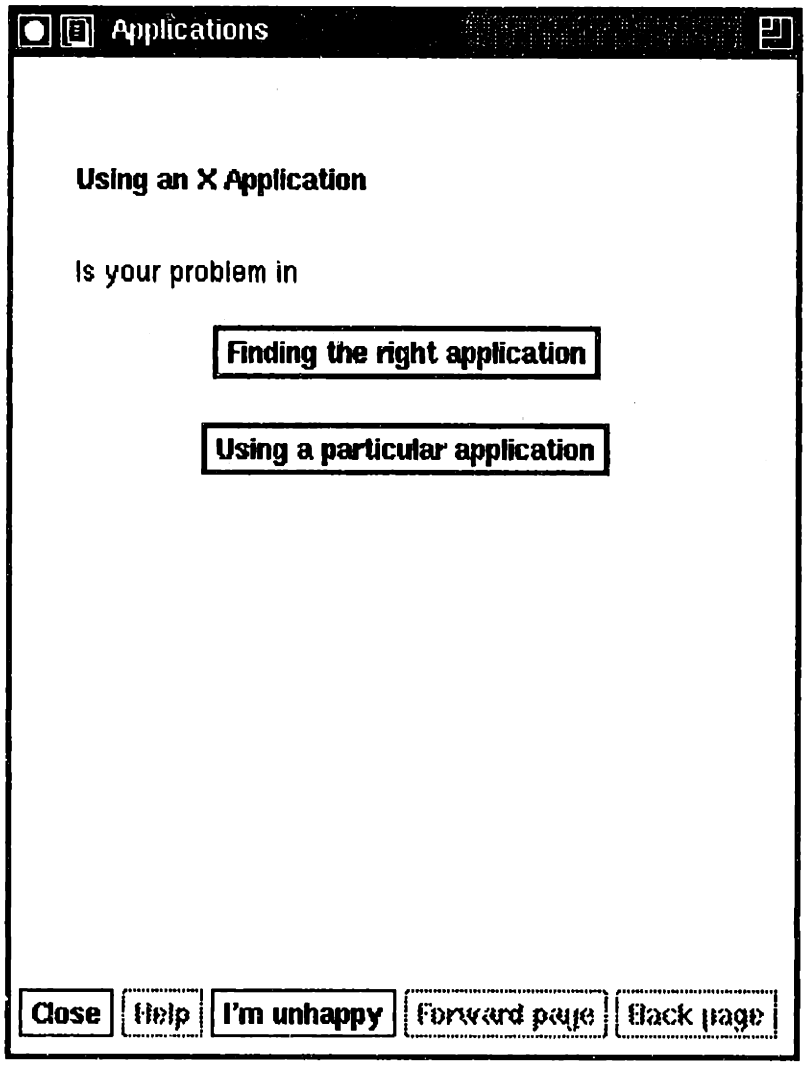

Figure B.1: SBrowser node with a menu of visual buttons

SBrowsers use a language quite similar to Scribe, a text mark-up language used at MIT. There is a very limited number of commands implemented at this time; the capabilities have been growing slowly as needed.[6] The commands include formatting

---

[6]The SBrowser node type relies on a second-generation version of my scribePSBrowserWidgetClass widget used in several hypermedia systems. The ScribePSBrowser could display Scribe's Postscript output, and this had the advantage of being able to utilize a full-featured commercial product. However, because Postscript is essentially a compiled language, the widget could not dynamically place or re-place images, video, or sound. The Knedit widget, on which the SBrowser is based, was an attempt to interpret a superset of Scribe. It uses this markup language because the requirements were not well enough known in advance

commands, such as boldfacing a string, italicizing the string, or displaying a bitmap. In addition, the writer of the node can include visual buttons; these visual buttons can serve either as links to other nodes or to trigger AGS or application commands. These AGS commands will be described later, and may also be included in the node's text. The application commands are written by the application writer.

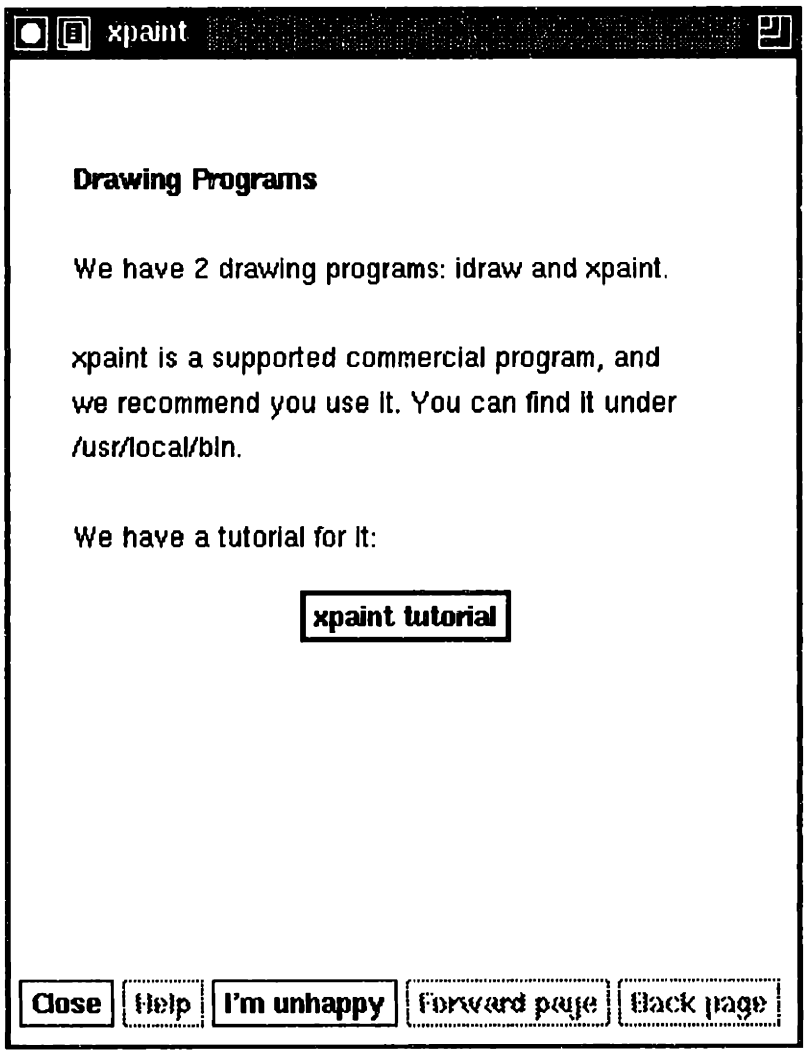

Figure B.2: SBrowser with text and visual button

---

to justify spending the effort on a WYSIWYG editor; markup languages are easier to construct in the initial research stages.

SBrowsers can point to any type of node. Thus one can activate a QA-Node, Code, Ascii, or Discussion node from an SBrowser as well as another SBrowser. Note that while the user views a tree, the links actually form a full network. In a typical AGS application, the network will be a non-cyclical directed graph since the user is trying to narrow down a question or to find a specific item. However, there is nothing in AGS to prevent cycles. (An authoring tool is provided to check for cyclicality.)

A variety of customizations at the node or node type level are provided. In fact, the beginning node for many Answer Garden databases (as shown in Figure B.3) is a SBrowser node. These are detailed in the customization section below.

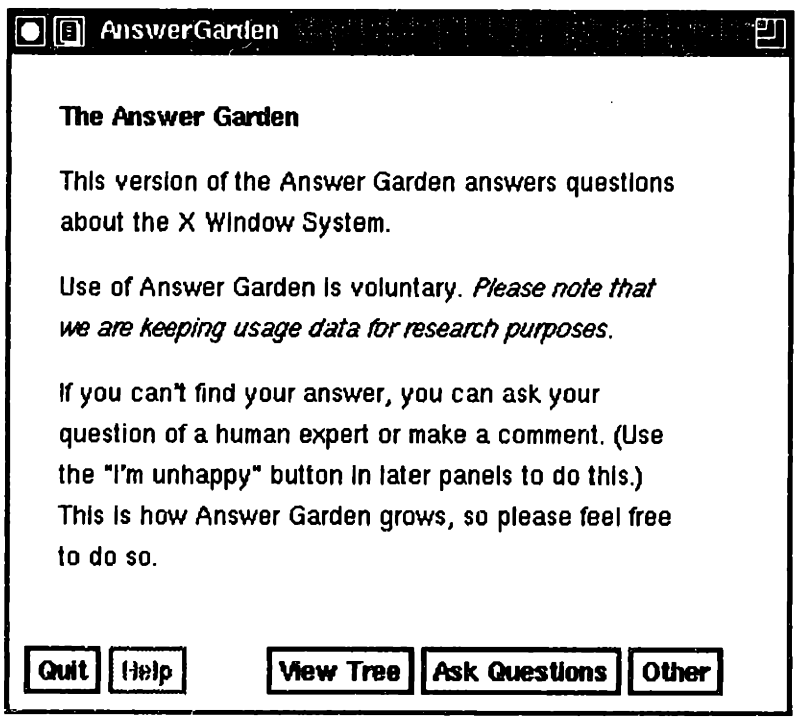

Figure B.3: SBrowser customized as the Control node (first node opened in Answer Garden)

B.3.2. Graphers

The Graphers display a tree of nodes (Figure B.4). When the user selects the label in the tree, the Grapher will activate the node given by the node name. (Again, each label

actually specifies a command string that is executed when the user selects the label. However, this is most often just a node name, and this node is activated as a result.)

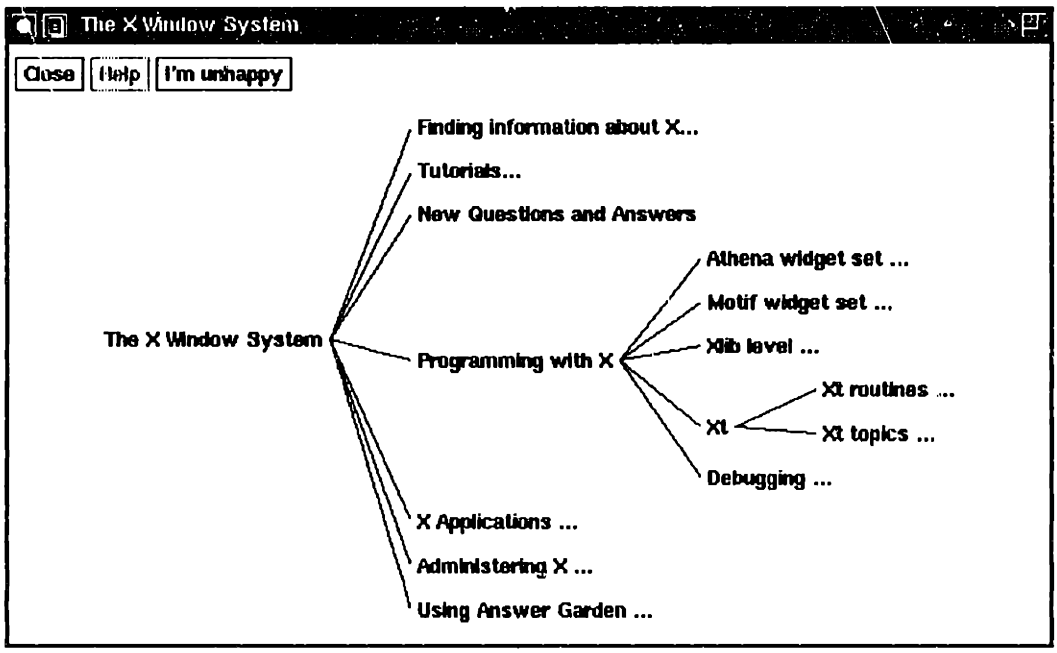

Figure B.4: Grapher for the X Window System database in Answer Garden

In the Answer Garden application, the Graphers form an alternative view to the tree specified through the SBrowsers. In fact, Graphers were developed to provide an overview picture in lieu of many SBrowsers. However, AGS applications need not conform to this model. The ASSIST application uses the Graphers for a reference view and the SBrowsers for a tutorial and problem solving view.

Graphers may appear at any point in an AGS application. They may be activated from an SBrowser, another Grapher, or a new node type.[7]

---

[7] I have found it useful to employ a "...." convention in AGS applications to denote a sub-grapher in Graphers.

B.3.3. QA nodes

QAs (sometimes called QA-Nodes) are used for situations where many pieces of information are placed in the same node. The canonical use is a series of questions and answers (Figures B.5 and B.6).

QA nodes use a markup language to indicate the annotations that go in a left margin. This provides the reader with the ability to quickly scan the information in a node.

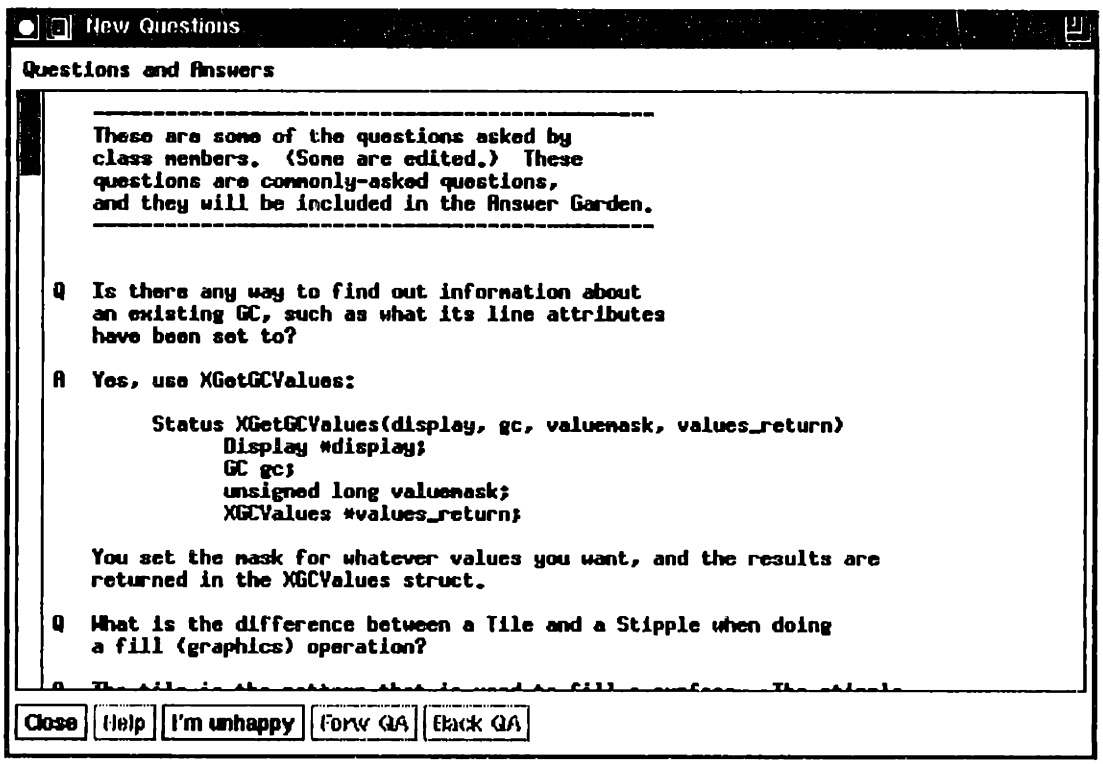

Figure B.5: QA node with new questions and answers asked by class members during the field study

Actions

Questions and Answers

Q We would like to pass our structure into a Translation Table Function in addition to the widget name and event.

A The problem of accessing program data from an action routine is an annoying one, to which there is no "clean" solution. I tend to use the X Context manager to work around the problem by assigining data to the window of a widget, and then retreiving this data from the action routine, since I have the widget (and window) ID in the action routine I can associate unique data with each widget in my application.

The Context manager is described in Scheifler, Gettys and Newman in section 10.12.

Q I would like to use the params argument of an Xt action procedure to pass a pointer to a structure. Is this possible?

A Well, the argument is declared to be a list of Strings, and Strings are expected to be character pointers, but in principle you could cast any kind of pointer you wanted and include it in the list.

However, the translation table parser only understands how to generate character strings params so the only way you could invoke an action procedure with a different type of pointer is to call it directly in your own code.

When translations were initially designed, some thought was given

Close | Help | I'm unhappy | Forw QA | Back QA

Figure B.6: QA node with a series of related questions and answers

What is a Shell widget?

Questions and Answers

What is a Shell widget?

Outline of this node
1. When you get a shell
2. Why shells exist
3. Special "features" of shells

1 When you get a shell

When you do an XtInitialize or XtAppInitialize, the routine returns a widget id. This widget id is the id of the shell widget for your application. You can create additional shell widgets (ie, shells that can be moved independently by the user through the window manager) by doing an XtAppCreateShell call.

2 Why shells exist

The window for the shell widget is a child of the RootWindow for the screen. Thus, the shell widget is the only widget that needs to contain code for dealing with the interaction with the window manager.

Thus you do not need to set any WMProperties in your application; the shell's

Close | Help | I'm unhappy | Forw QA | Back QA

Figure B.6: Tutorial using QA node type

The QA-Node was created to allow experts to throw many questions and answers together in one node without having to structure them. In the Answer Garden application, it was hoped that the diagnostic questions specified in the SBrowser would lead to atomic answers provided in other SBrowsers. However, a mechanism for allowing questions and answers to be quickly placed in an information database was required, and this requirement led to the QA node type. In Answer Garden, QA nodes are also used for some tutorials and documentation (Figure B.6). In other AGS applications than Answer Garden, QA-Nodes have been used for lists of bug reports and enhancement requests.

Future enhancements of this node type would be useful.[8]

B.3.4. Discussion nodes

Discussion nodes are simply compilations of Ascii data, such as a series of electronic mail messages. This node type was provided to allow the visual discrimination of nodes that contain a series of opinions. In the Answer Garden application, as well as others, users would want to be able to quickly distinguish situations where the expert could provide a definitive answer (or at least an approximation of one) from situations where there were many opinions about an answer (Figure B.7). Examples of questions leading to such a node might be "What is the best restaurant in Boston?" or "Which is better for coding an application, Lisp or C?"

Discussion nodes can also be used for tutorials and documentation.

---

[8] An important extension would be the ability to select portions of the QA based on a set of additional tags. For example, users should be able to view only the text pertaining to "release 5."

popup prompt design

Topic discussion

```
Organization: MIT LCS, Cambridge, MA
Subject: Basic questions on X Toolkit Intrinsics


[Summary:  I would like to call XtMainLoop recursively.]

What are the best ways for a Toolkit client to query the user and wait
for a response?  I understand that this goes against the spirit of the
toolkit, which is to have the program respond to the user rather than
vice versa, but there are situations in which it is necessary:

For example, an exceptional condition is discovered while processing a
dialog box.

From: (Donna Converse)
Subject: Re: help (with a popup prompt design)
Organization: X Consortium, MIT Laboratory for Computer Science


> I want to have a generic warning routine, call it display_warning(),
> that when invoked will popup and display a warning message
> (under the Xt instrinics) and require the user to select a proceed or
> cancel button.  The problem
> is that I want display_warning() to return a value depending on the
> user's selection. How do I get display_warning() to wait until a selection
> has been made before returning?.

I think the programming model for control flow needs to be changed from a
```

Close | Help | I'm unhappy | Forward msg | Back msg

Figure B.7: Discussion node containing electronic mail messages about the design of popup dialog boxes

B.3.5. Code nodes

Code nodes are essentially the same as Discussion nodes. The label at the top is slightly different to allow users to quickly determine that a node contains program code segments (as shown in Figure B.8). It was used in the Answer Garden field study. Users can cut and paste example code into their work from the node type.

B.3.6. Ascii nodes

Ascii nodes provide ready access to external, Ascii files. They were provided to allow the quick inclusion of pre-existing files (or files that are to be edited on a frequent basis as part of normal operating procedures). However, finding the set of experts associated with an Ascii node is more complicated and less flexible than for other node types since the Ascii nodes do not have AGS headers.

Buttons in the Motif widgets

Code segment

```
int i;
char str[10];

Arg arglist[5];
XmString my_string;


/* Set up the shell widget. */
shell = XtInitialize(argv[0],"XToes",NULL,0,&argc,argv);


/* Set up the composite widget, row column manager */

rowcol = XmCreateRowColumn(shell, "foot", NULL, 0);

/* Set up the first 8 push buttons.  Label them with
   "button1" through "button8".  Set their callbacks to
   everyone_else(). */

for (i=0;i<NChildren;i++)
    {
        sprintf(str,"buttonXd",i+1);
        my_string = XmStringCreateLtoR(str,XmSTRING_DEFAULT_CHARSET);
        XtSetArg(arglist[0],XmNlabelString,(XtArgVal)my_string);
        button[i] = XmCreatePushButton(rowcol, str, arglist, 1);

        XtAddCallback(button[i],XmNactivateCallback,everyone_else,NULL);
        XmStringFree(my_string);
```

Close | Help | I'm unhappy | Forward msg | Back msg

Figure B.8: A code segment from the X Window System database in Answer Garden

B.3.7. Other node types

Other node types can be quickly designed and implemented. Other existing node types include a database access node type, a journal entry node type, a parameter editor node type, and a help node type. The database access node type allows an AGS application to access remote databases.

B.3.8. The information in a node

As will be described below, the actual physical storage mechanism is not specified by AGS, and so no specific file format is required. Each physical storage mechanism may have its own format. In addition, each node type may have its own fields and formatting requirements.

Moreover, nodes need not even have an associated physical file. It is possible for a node to be *virtual*, and thus, built dynamically. An example of such a node might be a manual page built specifically upon request or a node built from a relational database query.

Virtual nodes not withstanding, AGS prefers to have a set of information for each node. Some of this information appears in header fields in the physical file.[9] Below is the minimal set I found required for the various AGS applications. The information for a node is:

data	**description**	**comments**
`body text`	data for the node display	The exact format is dependent on the node type.
`node expert`	the address or set of addresses for outgoing communications.	Used by AGS to determine the outgoing address. (AGS may override the address specified in the node in certain set circumstances. See the Communications Services section below.)

[9]This is a weakness in the current implementation of AGS. If there are no header fields, outgoing communications default to an address specified in the customization field. Therefore, an application consisting primarily of virtual nodes or of Ascii nodes will allow little routing.

Problems with having the expert specified within the node's file occurred even in the Answer Garden application, where each node had a separate physical file. This was a technical finding in the field study.

author	the author of the node	Set by the original author during node creation in the editor, or inferred by AGS.
show_author	level of identification for the author	Users identify the information's quality in part by the author of the information. However, some authors do not wish to be identified by name for fear that they will be inundated with queries.[10] On the other hand, some authors, such as consultants, want to be identified. AGS provides the capability for authors to identify whether they want to be known by name, known by their organization, or not known at all. Although the value is kept by the system, no node types currently use this value; the current behavior of all node types is not to identify the author at all.

[10]Users often wish to have the expert identified so they can evaluate the quality of the information. On the other hand, the first time Answer Garden was ever shown to X experts (who were to participate in the Answer Garden field study), all quickly requested that their name be removed from the node. On further investigation, I found that this could be resolved by removing the electronic mail addresses of the experts since most users would be sufficiently geographically distant.

author's organization	the organization of the author or last modifier.	Used by AGS to determine the outgoing address for communications. It is used to determine whether the node has been modified at a local site, which is required information for both updating the information database and determining the correct expert for the node.
expiration date	the date after which it is presumed that the data needs to be refreshed.	One of the problems in an information system is that the data becomes obsolete. In other words, information has a shelf-life. When authoring, the author (or modifier) can specify a date or string (such as next_revision or rev5.5). No specific activity occurs on or after this date; however, an authoring tool is provided to mail the specified author an alert concerning the node on that date. The author can then choose to ignore the message, modify the date, or modify the data.
last modifier	the last modifier of the node	Set by the editing subsystem. It is used by AGS to determine the outgoing address for communications.
last mod date	date of last modification	Set by the editing subsystem for version control.

mod num	a version number	Set by the editing subsystem for version control.

The last three fields are maintained automatically by the editing system in AGS.

Each node file, then, is a semi-structured object since it is a combination of structured and unstructured data. Node types (and even node instances) may have their own, additional header fields. For example, a node type might include keyword or "see also" fields. This provides the node type author with the capability of having additional structured information within a node. Additionally, an application writer can include structured data specific to the application as a whole through this mechanism.

B.4. Customizing an AGS application

An existing AGS application may be tailored to a large extent through a set of application and Xt widget resources. There is no firm boundary between minor modifications and extending AGS, since only a small amount of code can substantially change an AGS application. However, this section concerns only modifications that require no code writing.

B.4.1. Customization

An AGS application can be customized to a large extent by site managers or the application writer by merely changing the X resource file for the application.[11] Changing

---

[11]Resource files in X are Ascii files that contain a list of property and value pairs. These properties may be specified using wildcards, and by doing so may apply to a specific widget, a set of widgets within a layout or node, or all widgets of a specific type, or all widgets. In addition, resources may be changed on a site-wide basis (i.e., for all users) as well as on a per-user basis. As well, AGS allows resource specifications above those of X.

the resource file for an AGS application allows the customizer to quickly change fonts, colors, sizes, positions, and general appearance quickly and without compiling. This is, for example, the way to change the label of the "I'm unhappy" button in the Answer Garden application. One can change it to whatever he wants in the resource file (or even on the command line when starting the application).

On top of general resources, AGS provides a set of resources for nodes in general and the application overall. For example, application writers, or even users, can select how many window stacks exist, whether the user is provided with a copy of outgoing messages, whether users can edit nodes, and what the startup nodes are.

Appendix C: Answer Garden Substrate Internals

This appendix describes the internals of the Answer Garden Substrate system (AGS). The appendix covers the same material as Chapter 5 but in substantially greater depth. Readers may also wish to see Appendix B.

C.1. AGS Services

As mentioned, the node types are the heart of AGS. But, in order to explain how to modify existing applications or add to the AGS system, it is necessary to describe the rest of the AGS system. AGS is really the confederation of nine services. The functionalities described below were required to complete a set of prototype organizational memory applications.

The services are:

1. A set of UI presentation objects. These are the node types described above. Note that the presentation objects may consist of specialized widget code as well as standard Xt application code.

2. A Communication Service. This is responsible for handling outgoing electronic mail or other communication mechanisms.

3. An Information Retrieval Service. There is considerable controversy in the information retrieval community as to the best information retrieval engine. AGS is engine "promiscuous," having an application-program interface (API) suitable for a wide

variety of engines. Several information retrieval engines are provided.

4. A Node Service. This provides I/O access to the nodes, as well as some basic information about the nodes, to the other services. The Node Service provides access at the logical level to the other Services; all another Service needs is the node name.

5. A set of File Services. These provide access to physical storage for the nodes. Each File Service handles one type of physical access, such as flat files, an SQL database, or a distributed infobase server.

6. Authoring Service. AGS provides a base level of authoring tools, including a built-in editor for node level information as well as the actual information content. It also provides a small set of independent authoring tools. End users may or may not have access to the Authoring Service depending on the application and the wishes of the site manager.

In addition, there is a set of minor services available anywhere within AGS:

7. Window management. AGS can put up many windows, too many for most users. These windows can be managed through the standard X mechanisms, such as the user's window manager, or they may be handled within AGS through the abstraction of "stacks".

8. Function Service. All through the system, particularly in the node types, standard functionalities are provided. These include system calls and assignment statements.

9. Blackboard Service. AGS provides a two-level blackboard architecture that can be accessed through the system-wide functions in the Function Service as well as special routines. This Service allows internal memory between AGS components as well as external programs and agents.

Diagrammatically, these Services appear as Figure C.1:

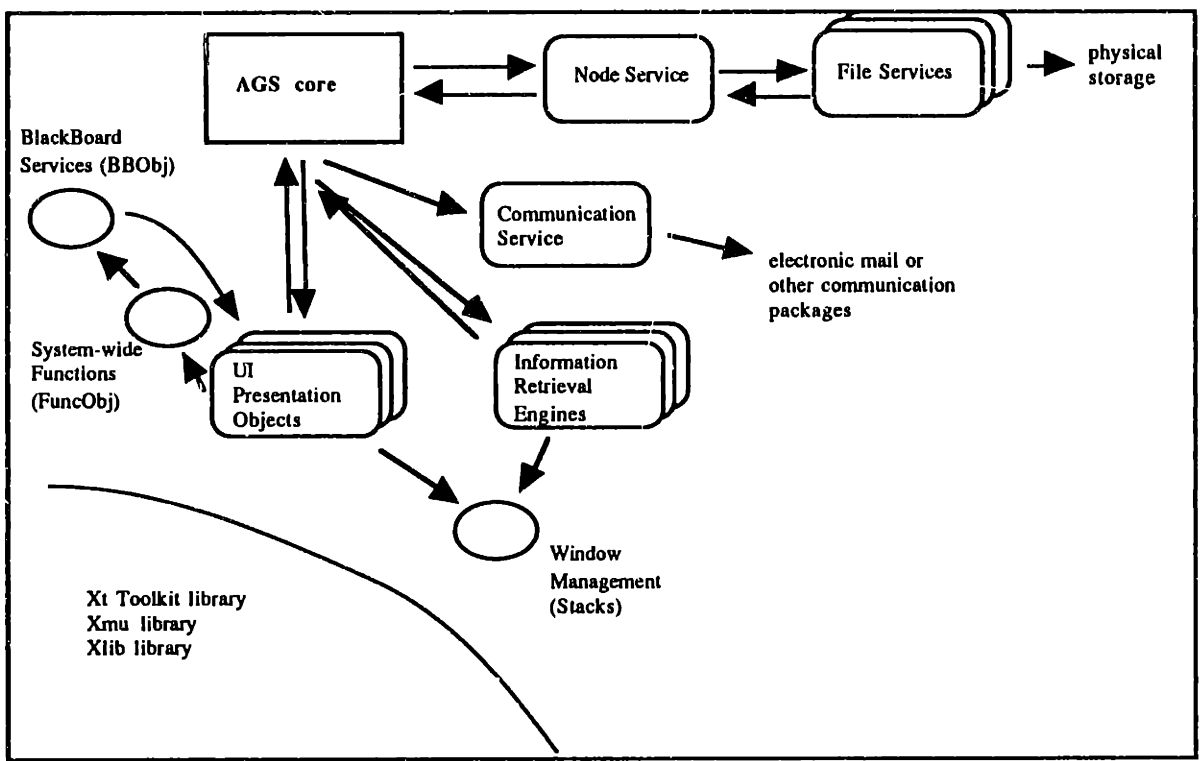

Figure C.1: The architecture of the AGS Services

Essentially, there are a relatively small number of "core" procedures that initialize and drive the various Services. These Services would be objects in an object-oriented system. (In fact, some of the services, particularly those that can be enhanced, not replaced, are Xt Objects.) The decomposition of AGS functionality into Services:

- Facilitates system maintenance.

- Localizes changes that are required to enhance the system by a system author.

- Localizes the changes that are required to enhance the system by an application writer, and thus makes it easier to enhance AGS.

Like any Xt application, AGS is event driven. Program action waits upon user input, timers, or system interrupts (usually for data availability).

I will discuss each of the Services in turn. Note that the User Interface presentation objects were described in Appendix B.

C.1.1. Authoring Service

AGS provides a rudimentary authoring sub-system. This includes an editor and series of authoring tools for editing and adding nodes to the system. Nodes may be edited either within AGS or through any other editor capable of dealing with the node's format. (For example, nodes could be edited using Emacs for a text node or a special graphics editor for an image node.) The internal AGS editor assumes that all nodes are in Ascii (including necessary mark-up).

It is possible from within the editor to add, delete, or modify the node information including any header data[1]. To do this, the editor is divided into two components, the NodeInfo and NodeEdit panes. There is one of each per node.

The NodeInfo pane (Figure C.2) provides a mechanism for the author to handle header fields. The exact mechanisms of this pane are not important here, but the default values when adding a node are settable by the site manager or the individual author in an external file.

---

[1] See the Authoring Guide in the X information database for details on the use of the editing facilities.

NodeInfo: Info_Authoring_AG

This is the editor for node information.

You can edit only the label, expiration date
and experts currently.

Node Type

SBrowser Grapher Discussion QA-Node Code Ascii-Node

Node name

Info_Authoring_AG Auto Manual

Label for Node

Using the authoring system Auto Manual

Storage Location

Info_Authoring_AG Auto Manual

Expiration date for node

06/25/94 Auto Manual

Experts for this node

ackerman@guilder.mit.edu Auto Manual

Okay Cancel

Figure C.2: NodeInfo pane for editing a node's header data

The NodeEdit pane uses an Xt text editor widget and allows an author to modify the node's text. Examples of nodes being edited are shown in Figures C.3 and C.4. Note that the text is merely Ascii. It can also be edited within any standard text editor, such as Emacs.

NodeFile: Info_Authoring_AG

Info_Authoring_AG

```
WRITING OR EDITING ANSWER GARDEN NODES
        Mark Ackerman
        ackerman@athena.mit.edu


Authoring
---------

To add or edit an Answer Garden node, you must select the appropriate
action from the edit menu.  You get the edit menu by pressing the
control key and the mouse button2 at the same time.  Hold down the
mouse button and drag the cursor to the appropriate action.  If this
doesn't work, see your AppDefaults resource file for the correct
bindings.

I'm going to give the easiest path, in my conception, through the
authoring process.  You may find another way easier - your choice.
After I go through the "official" way, I will tell you how to brute
force any change you want.  (Hey, this *is* Unix.)


Editing node of an existing node-type
-------------------------------------

The easiest way to author is to modify an existing node.  After all,
you are going to generally modify the information network.

Let's start by editing the Root node.  On the Control node (the
first node you see, the one that says this database is about the
X Window System), go into the button that says "Ask Questions".
```

Abort | Close | Save File | Edit File | Edit NodeInfo | Add Node | Show

Figure C.3: NodeEdit pane for editing an AGS node's text body

Figure C.4 shows the markup language used to create visual buttons in an SBrowser node. Visual buttons can be created similarly in Grapher nodes.

When new nodes are added in a Grapher node, the insertion into the corresponding SBrowser node can occur automatically. This allows the SBrowsers and Graphers to remain coordinated, if that is desired. This is under the author's control, settable from a menu item in the Control node (the first node on the screen). This facilitated the construction of Answer Garden databases.

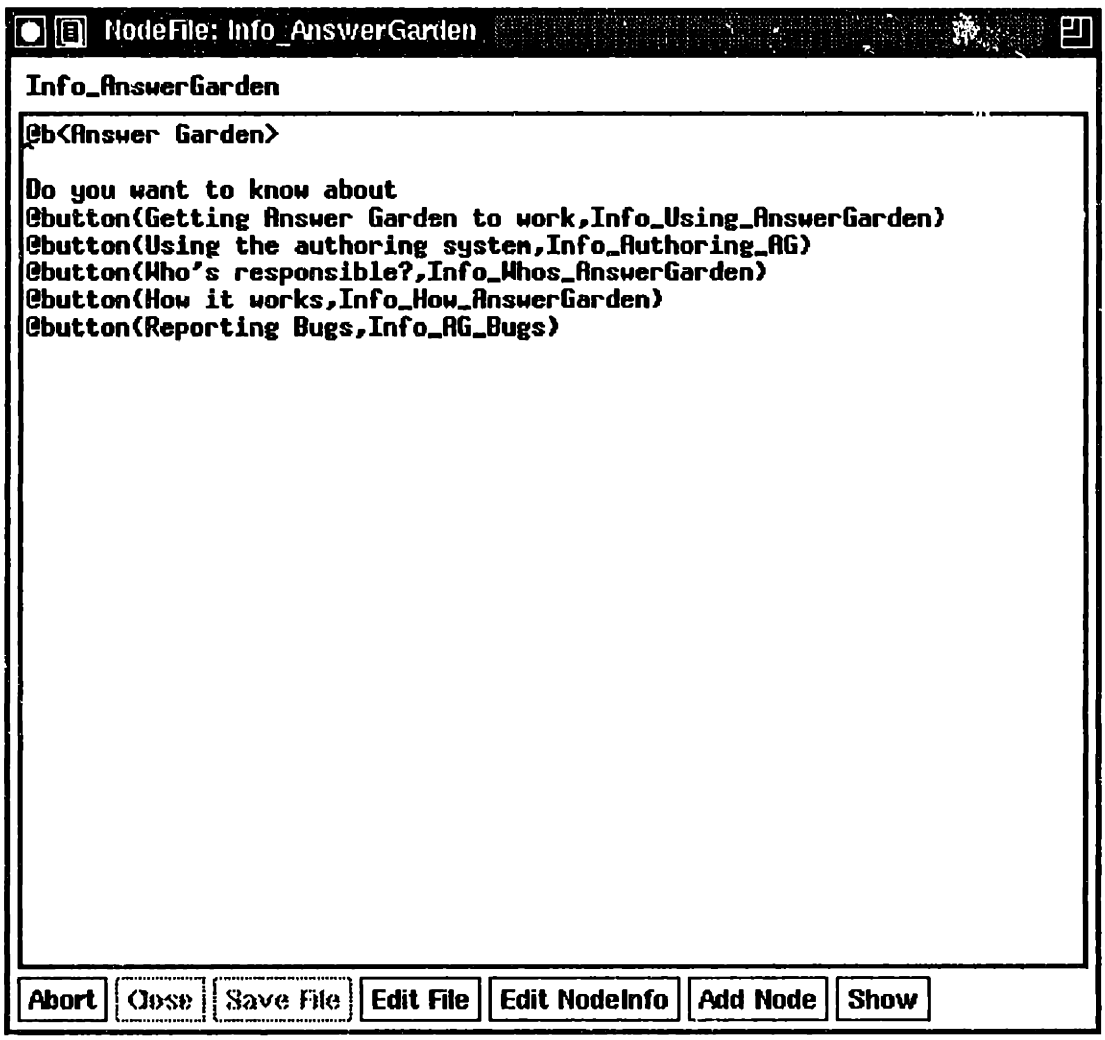

Figure C.4: NodeEdit pane for editing a SBrowser's text body

Besides the node editor, there is a suite of authoring tools, most of which can be run as background agents. These tools include processes:

- to watch for expired nodes,
- to search for cyclicality in the information database,
- to watch for orphan nodes,
- to check for links without references, and
- to automatically make Graphers from the SBrowser nodes.

Future enhancements include agents to watch for ill-formed information databases, databases with straggly shapes or with nodes that contain too much information.[2] Another type of agent requested is one that watches for nodes that are heavily used and send a request to the expert to move the node up in the network.

Nodes may be created or edited by end-users if the system administrator allows it. Nodes, if they are contained within Unix files, may be locked or hidden for individuals or groups using the standard Unix file security mechanisms; the current implementation of AGS follows these.

C.1.2. Communications Service

Users in an AGS application can communicate with experts or others directly through the application. To do this, AGS-level support is provided for asynchronous communications as well as synchronous communications.

AGS currently provides support for three Unix standard mail systems -- /bin/mail (usually called "binmail"), /usr/mail, and mh (the Rand mail handler). Any mail engine using the sendmail mail transport can be easily supported; others can be added.

AGS can also provide support for synchronous communications; the only current implementation uses the Zephyr messaging system (DellaFera et al. 1988). When used, communication is first attempted with Zephyr. If the person is unavailable, then an electronic mail message is sent.

---

[2]One possible application is a variant of Answer Garden that places incoming messages from a bulletin board or distribution lists in nodes according to the topic. After the conversation or thread dies down, an expert should be notified so that he can prune away the erroneous and non-essential information.

Mail (or Zephyr) messages are sent out in AGS as semi-structured mail messages. In the Answer Garden application, they are mailed with header fields that include the user's name, information about the user (usually containing the author's address and telephone number), the user's organization, some node information including node name and version number, the system release number, and a history list for the last 15 minutes. It is possible for the user to be anonymous, in which case the user information is stripped away by an intermediate layer. The additional fields should provide the recipient, such as an Answer Garden expert, with the ability to more efficiently and effectively handle queries. For example, it is possible to see from the history list whether the user was looking through nodes in a circular or random manner, indicating he was lost.

Application writers can tailor communication services to select a particular mail system; to provide a copy of the outgoing messages, with or without the semi-structured header fields, to the user; to write particular messages to the user; and to disable communications entirely. In addition, application writers can write procedures to change the header fields for outgoing electronic mail or to add new mail or messaging systems. For example, it is easy to add the facilities for asking questions through voice-mail with an electronic-mail trailer.

Additional details on the Communication Service can be found in *Appendix D.1: Communication Service components.*

C.1.3. Node and File Services

Information in AGS can be kept on secondary storage until required. This allows for arbitrarily large information databases with robust access.

To facilitate this, access to information in physical storage is divided into two services, the Node Service and at least one File Service. The Node Service handles

information as logical nodes in the system; the File Service translates the logical nodes into physical files or accesses.

The Node Service consists of a specific file, the AGNodeFile, and a set of routines. The AGNodeFile contains five fields for each node in the system: the symbolic name of the information node, the label for the information node, the type for the information node, file service to be used, and an arbitrary data slot. For the default File Service, this last field is the location for the information. The Node Service also consists of a set of routines for accessing fields in the AGNodeFile and for accessing the information and header fields in the nodes themselves.

In addition, some applications may wish to create virtual nodes, nodes created dynamically from information in other databases or through calculation. The AGS routines allow such nodes to be added to the system as needed.

Any node, moreover, may be either temporary or permanent. Temporary nodes are added to the Node Service only for the lifetime of the application instance; permanent nodes are added to the Node Service and to the AGNodeFile as permanent storage. Temporary virtual nodes are used heavily in the ASSIST application; for example, many help nodes are built from an existing database.

AGNodeFile files may include other AGNodeFiles by reference. (The files are Ascii with a specified format.) This was required by the Answer Garden application. A site wished to have multiple databases for various topic domains for the authors and editors, as well as the ability to combine them in a single database for the end-users. Additionally, if the UseLocalNodeFile resource is set to True, the system looks for a AGNodeFile or .AGNodeFile in the user's home directory, allowing the users to have their own, private AGS nodes.

The Node Service uses the appropriate File Service to access physical storage. The separation of routines handling physical storage into a File Service layer allows storage in flat files, in SQL databases, or on information servers. To implement a new storage mechanism, one merely writes the specified routines for a new File Service. AGS currently comes with only a flat file implementation.

AGS usage, as it was designed, should be that only the Node Service routines call File Service routines. If this practice is followed, then an application writer can add a File Service without breaking any of the user interface objects. However, an application writer can, if he wishes, directly access the important File Service routines.

See *Appendix D.3: Node Service* for additional information on the Node Service.

C.1.4. Information Retrieval Services

AGS can include a number of information retrieval engines. Instead of relying on a single information retrieval method as its method of retrieving text and other kinds of information, AGS provides mechanisms to include any and all engines. Thus, the application writer can use his preferred engine or engines. For example, an application writer might include a vector or probability based engine. Additionally, the application writer might include both, allowing he user to decide which to use.

Currently implemented engines include a simple full-text retrieval engine, a semi-structured engine, and an adaptive engine. The semi-structured engine allows the user to search on values in the header fields as well as the text in the node. For example, using a

semi-structured engine, the user could retrieve questions and answers in all nodes modified after 12/31/92 that contain the terms "Motif" or "Xm".[3]

C.1.5. Blackboard Services

AGS provides a two-tiered blackboard. These blackboards allow other system components or external agents to publish and to retrieve state information from one another.

The first tier of the blackboard is internal to an AGS application. This blackboard, or dictionary, allows data storage and retrieval. This could be used, for example, to have the user answer questions about the machine type he uses early in the application, and then to rely on those answers later when selecting paths or grapher diagrams.

The second-tier can be read and written among applications; this is the "global" blackboard. One possibility for its use is to have the editor publish the name of the node being edited to the blackboard, allowing design agents to examine either the particular node or the state of the information database. This blackboard is built upon the property mechanism in X11, and therefore requires an X server connection to be established.[4]

---

[3]If the header fields include a keyword or index field, and this field is used by authors, the semi-structured engine includes the capabilities of a keyword-based retrieval engine.

[4]Therefore, any design agents must also have a display connection to the same X11 server This is not a problem in authoring, where the design agents will generally have access to the same machine as the editor. This will also be true for processes that serve as query or navigation critics for the individual user. However, using an X11 server-based blackboard for group activities assumes that all members of the group have access to a common display server. Note that this display server is not used for anything but data access; no window operations need be performed. It should also be noted that the property mechanism, because of the round-trip overhead to and from the server, is not best for large-scale data transfer.

The Blackboard Service is easily accessed from anywhere within an AGS application.

C.1.6. Window Management Service

An AGS application can spawn dozens of windows. Through a combination of window manager functions and AGS functions, applications can control the spread and number of visible windows. The basic abstraction for this in AGS is the stack.

A stack appears visually as a deck-of-cards. The default is to spread towards the right-edge bottom of the screen with each node slightly displaced from the previous, but this can be easily changed. Since stacks can have a maximum depth, the application can even appear as a single pane. Other screen management functionality includes the ability to lock nodes in place (forcing them to remain on the screen) and the ability to move back to previously opened nodes.

A node may be placed in a particular stack by the node name, by the node type, or by default. The stack maintains the layout on the screen of its nodes. The addition and deletion of windows from the stacks are invisible to the AGS application if the stack mechanism is used; the AGS core functions handle them. Generally, application writers or site managers need only specify how the stacking is to work using the external resource file.

C.1.7. Function Service

Several useful commands are available throughout AGS. These include system calls to start any Unix process (so AGS can start external programs), the ability to trigger any callback from any node (so a node may trigger arbitrary functionality from its data or its visual buttons), the ability to open and close nodes, and general assignment statements to blackboard variables. These are the @system, @dynamic, @create, @close, and

@assign statements respectively. Not only may node type code trigger these, but any node's data may include these commands.[5]

In addition, the Function Service contains a generalized parser for '@' commands so that a node type may have its own commands. For example, a node type capable of audio might have a new command

```
@sound(sound_file, duration, start_position)
```

which could be parsed for the node type by the Function Service.

C.2. Extending AGS

As mentioned above, there is no solid line among customizing AGS (covered in the previous chapter), modifying an AGS application, and extending AGS. I concentrate on modifying and extending AGS here. There are three types of modifiability and extensibility possible in AGS:

1. Modifying the functionality of an AGS application by changing or adding an Xt callback. One might wish to do this in order to provide the user with an additional popup message, add additional help features, or even to add a print capability.

2. Adding a new AGS node type. One might want to do this, for example, to create an application that included triggering object modules or SQL database queries. Additionally, new node types

---

[5]The node type code must pass these, or any unrecognized '@' commands, to the Function Object for handling. There is only one Function Object in the system, which is an Xt Object type for easy inclusion in a widget class' code.

could present the same information differently, such as in an outline or visual format.

3. Adding or changing an AGS service. One might do this to add a new file access mechanism, a new communications mechanism, or a new information retrieval engine.

C.2.1. Extending AGS: modifying AGS applications through dynamic callbacks

Because of the general architecture imposed by the Xt toolkit, much of the user-driven functionality in AGS exists in functions triggered by user activities such as clicking on a visual button with the mouse. The Xt objects, called *widgets*, trigger functions called *callbacks* whenever certain user events occur. (AGS provides default interactions, but they can be overridden by the application writer or even by the user.[6])

Callbacks in AGS may be modified, first, by substituting code or callbacks for those that exist, or, second, by adding dynamic callbacks. The first method is the more normal Xt manner, where the application writer creates a new callback, either with the same name or by changing the `XtAddCallback` call within node type code. As a first level of modifiability, it has two important problems: It unfortunately ruins the object-oriented nature of AGS code, and it requires an intimate understanding of AGS internals.

To facilitate the easy substitution of functionality in an object-oriented manner, AGS provides an extension to the callback mechanism called dynamic callbacks. These callbacks are bound during run-time, on a node by node basis. They can be bound, for

---

[6]Xt provides a mechanism, called translations, that allow the user or site manager to respecify the user events triggering the various callbacks for a widget. They can respecify the translations in the external resource file.

example, from an external, Ascii resource file either by node name or by node type. (Note that the node type is not synonymous with any widget class; the lookup mechanism is supplemental to Xt.)

Using dynamic callbacks, the application writer creates a new callback, and merely substitutes it, by its name in the external, Ascii resource file, for any set of nodes desired. For example, an application writer might wish to provide a new close callback to close the connection with a database server. The writer merely overrides the system-wide callback with his new dynamic close callback by merely specifying the name of the new callback in the resource file:

```
*closeCallback:  myNewCloseCallback
```

for all nodes, or

```
*thisNodeByName.closeCallback:  myNewCloseCallback
```

for only one node.[7]

A further advantage of the dynamic callbacks is that they can be used in any node as an AGS command. Thus, clicking on a visual button in a Grapher or SBrowser node can trigger a dynamic callback for any functionality that the application writer would like.

In addition, the application writer can add new visual buttons to all nodes within an AGS application. For example, even though the standard AGS node types have three visual buttons by default ("Close", "Help", and "I'm unhappy"), an application writer can require that all nodes have five buttons, two of which are set to his own callbacks. These can be bound to specific callback functions in the application code (such as locking the window in place or editing the parameters to a subroutine function) in the external, Ascii resource file.

---

[7]This is possible because the system-wide close callback was built as a dynamic callback.

C.2.2. Extending AGS: adding new AGS node types

In order to use new information or presentation objects, an application writer may wish to extend the AGS system. The place where AGS shines is in its extensibility. The extensibility of AGS permits substantially different applications within the same system paradigm.

As mentioned, AGS has an object system built upon the underlying C code. Classes, or node types, consist of a class data structure as well as code. Instances, or nodes, may or may not have additional data structures. Adding a node type is merely a matter of setting up the class data structure and adding some additional code.

The data structure, called a `NodeType`, is initialized at compile time. It essentially includes function pointers to the relevant procedures. (Details may be found in *Appendix D.2: Adding New Node Types.)*

The attempt was to provide a minimal set of functions that needed to be written to create a new node type. Usually, only the one function, the `create_function` to create the necessary user interface, needs to be written by the node type writer..

The creator of the node type has a substantially free hand since there are few requirements imposed by the system. No restrictions on the type of information in a node are imposed, and few restrictions are made on the internal layout of the node's interface.

In detail, the minimal requirements imposed upon a node writer's efforts are:

1. The node writer must create an Xt top-level shell.[8] If a geometry specification is provided to the node code (through the parameter list for the create function), the node writer should create the shell using that geometry. This allows AGS to manage screen real estate properly.

   With the exception of the incoming geometry specification, this is standard Xt application code.

2. The node writer creates the standard Xt hierarchy of widgets he wishes to appear in the node.

   This is standard Xt application code.

3. At the minimum, the node writer must create visual buttons for the close, help, and question functions. A convenience routine is provided to both create the necessary number of buttons and bind them to the default dynamic callbacks. Writers are encouraged to use this convenience routine, since it will also allow an application writer to add visual buttons at the application level without changing existing node type code. (See Appendix D for additional details.)

---

[8]The shell widget is the root widget in the widget hierarchy. Thus, each separately moveable entity (by the user) generally has a separate shell widget.

4. At the very end of the `create_function`, the node writer should perform an `XtRealizeWidget` on the shell, an `XtPopup` on the shell, and return the shell ID to the calling AGS procedure.[9]

   With the exception of returning the shell ID, a minor change, this is standard Xt application code.

Thus, only four steps -- three of which are not unusual Xt programming practice -- are required. The slightly additional burden allows AGS to manage and layout the new nodes and to provide general AGS functionality.

In general, since the node uses Xt widgets for display, there is no separate display code for the node type. (However, a node type could be specified that did additional processing when the node had to be redisplayed.[10])

For more information, see *Appendix D.2: Adding New Node Types*.

C.2.3. Extending AGS: support functions for new node types

The above four steps are the only ones required for a new node type. However, a large number of support services are provided to the node type writer. These support services follow the AGS Services:

- Node and file system services and components. A number of routines exist to read and write files, to obtain and modify node

---

[9] Widgets are known to an application by their ID.

[10] In Xt terms, there is nothing prohibiting a Node Type from setting a callback or event handler on expose events. On the other hand, a custom widget's Expose procedure might be a better way of achieving this end.

fields (including standard and non-standard header fields), as well as to add, modify, and delete nodes from the system. See *Details: Node Service* for more information.

- User interface services and components. AGS comes with a number of specialized Xt widgets. They include the Layout widget (which serves as a bulletin board and is used in the Grapher node type), the Grapher widget (which lays out a tree of buttons and is used in the Grapher node type), the Knedit widget (which implements the functionality of the SBrowser node type), the Outline widget (which places a list in an outline format and is used in the NodeOutline node type), the NodeOut widget (which specializes the Outline widget for the NodeOutline node type), and the GrapherButton (which implements a number of additional features over and above the normal visual button including knowing its name and having multiple callbacks).[11] See *Details: AGS Widgets* for more information.

- Miscellaneous services and components.

  Node type code may open arbitrary nodes using the `AG_Create_Node` call.

  Node type code may retrieve arbitrary Xt resources based on node name or node type using `AGS_Get_Resource`.

---

[11]The Tree widget in the Athena widget set is a derivative of the original Grapher widget.

The Layout widget was originally done by Russ Sasnett. I have performed only minor fixes on it.

There are also a number of utility calls to handle string manipulations, the history list, and similar, common activities.

See *Details: Miscellaneous Utility Calls* for more information

C.2.4. *Extending AGS: adding Service components*

As I have mentioned many times, the File Service, Communications Service, and Information Retrieval Service were designed to be extensible. (The Node Service was not designed to be extensible, and while the minor Services are extensible, they will not be described here.) In addition, new Services can be added to an AGS application; this will be described below.

The "push" for extending the three services differed to some extent, and this is reflected in the nature of what can be extended. I believed that the File Service would need to be extensible to allow different organizations to access physical storage in different ways. Before the Answer Garden application was field tested, it became clear that organizations would want a flat file storage, a distributed file server (without NFS), and storage in a relational database. In order to provide such capability, I broke the Node Service away from the File Service, and designed the File Service as a confederated set of routines.

On the other hand, the communication mechanisms required by different groups might differ in two ways. First, there might be differences in transport mechanism. As mentioned, AGS supports standard Unix electronic mail systems as well as the Zephyr messaging system. Zephyr was selected as a test since it uses a substantially different transport mechanism. Second, organizations might differ in the types of headers and mail

structures desired. I believed that application writers needed to be supported when they wished to change communications functionality, and not just transport mechanism.

Finally, the information retrieval community had various favorite information retrieval engines, and one can find support in the literature for all of them. I did not wish to conduct information retrieval engine research; rather, I wanted to create a better environment for information access. Therefore, many different engines needed to be supported, all of which would have substantially different internals. The Information Retrieval Service allows many different engines to be used by the end-user; all can co-exist. (In addition, I added Blackboard services not only to allow design and user agents, but also to permit external information retrieval processes.)

Adding a new File Service or Information Retrieval Service component merely requires extending an initialized data structure that keeps track of the components in each of these Services. For example, to add a new File Service component to use SQL databases, one would write the required set of functions and bind them into AGS through a File Service data structure.

Currently, the Communication Service must be augmented in place, primarily through adding a new callback.

C.2.5. Extending AGS: adding a new Service

Occasionally, application writers may wish to add to the repertoire of AGS Services. Writers may wish to add database access functionality or other similar features.

Adding a new Service requires little additional work beyond writing it. To trigger the new functionality, the writer can add items to the menu on the startup node, add new visual buttons to all nodes (regardless of their node type or previous existence), and

override system callbacks. Additionally, routines from the new Service can be used in any node type code.

Appendix D: Answer Garden Substrate (AGS) Details

This appendix provides additional details for the Answer Garden Substrate (AGS). This appendix should be read in conjunction with appendices B and C.

D.1. Communication Service components

The Communication Service assumes that the act of sending out a message can be decomposed into a number of steps. These steps form the lowest level of Communication Service components. Above these are some higher level callbacks that use the low-level building blocks to compose and send the message.

There are two reasons for the modularity, that is, decomposing the functionality. First, it was anticipated that AGS application writers would want to enhance existing routines so as to include additional information, change the location of temporary mail files, or utilize additional mailers. An example of this would be to allow the use of mailers that do not depend on Unix's /bin/mail or to use other transport mechanisms such as a synchronous communications system.

Second, completely different transport mechanisms such as video or voice-mail will require quite different routines. However, in these cases, one might still want to send book-keeping information (such as the user's history list) in a parallel electronic mail message. These routines may be reused for that.

Following are the components for the default Communication Service. How to extend the Communication Service to include other messaging capabilities is explained in a later section.

The following routines are the elementary building blocks for the Communication Service. In the default configuration, they are usually called from within the routines described below.

`Comm_Initialize_Mailer`	Initialization routine. Allocates and initializes the data structure for the message instance.
`Comm_Get_Expert`	Determines the outgoing message address. This is the expertise engine. See comment #1
`Comm_Write_Outgoing_Header`	Writes a complete header onto a temporary file. This routine should be changed to include or exclude mail header fields. See comment #2.
`Comm_Write_Copy_Header`	Writes a minimal header onto the temporary file. Used for the copy of the mail message to the user.
`Comm_Write_Outgoing_Body`	Writes the body of the message on a temporary file. Retrieves the text from the appropriate text widget in the mailer popup.
`Comm_Send_Mail`	Handles transport mechanism for sending copy to user. Checks the resource for preferred mail transport. Only this routine needs to be changed to support a new message transport engine if the message format stays the same.

Comm_Send_Copy	Handles transport mechanism for sending copy to user.
Comm_Destructor	Frees up the data structure for the message instance.

The following routine is used to begin the communication process by popping up a dialog box for the user's message. This routine is usually set up automatically by the routines described below.

Comm_Setup_Question_Node	Routine to create the mailer popup. This routine would change if the application writer wished to radically change the visual appearance of the mailer popup (for example, to have new buttons in a radiobox to signify priority).

The following Xt callbacks are called by clicking on a visual button in the mailer popup. These routines are usually set up automatically by the routines in the next group.

Mailer_Send_Callback	Callback to send out a text message with full header fields. Invoked from the "okay" button on the mailer popup. This is the usual callback. See comment #3 for more information.

`Mailer_Send_With_Copy_Callback`	This is a callback to send a text message. Invoked from the "okay" button on the mailer popup. It is the same as `Mailer_Send_Callback` except that it also sends out a copy of the message (with a reduced header) to the user. It could call `Mailer_Send_Callback`, except for some statistics keeping.
`Mailer_Send_Straight_Callback`	Callback to send message without any header fields. Invoked from the "okay" button on the mailer popup. Useful only for sending a form straight out using the MH mail handler. See comment #4 for more information.
`Mailer_Cancel_Callback`	Callback to pop down the mailer popup without any further action. Invoked by the "cancel" button on the mailer popup.

The following routines use all the above building blocks. They are the callbacks that can be bound to a visual button in all of the nodes. This visual button is usually referred to here as the question button, and is often given the label "I'm unhappy."

General_Question_Callback	Callback to popup a mailer dialog box. The routine sets up two callbacks, one for cancelling the mail popup, and one for sending the message. Also calls Comm_Setup_Question_Node to create the mailer popup and put it on the screen. This callback can be specified (from a resource file, e.g.) using the string GeneralQuestionCallback. See comment #4 for more information.
General_Question_With_Copy_Callback	Callback to popup mailer. This routine sets up two callbacks, one for canceling the mail popup, and one for sending the message with a copy to the user. Also creates the mailer popup. This callback can be specified (from a resource file, e.g.) using the string GeneralQuestionCallbackWithCopy. See comment #4 for more information.
Comm_Excuse_Me	Callback to pop up a message box telling the user that no questions are allowed. The site administrator (or someone) has turned off mail for some reason (such as not having electronic mail or for security reasons). AG is only a static information system. (How boring.)

D.1.1. Comment #1: the expertise engine

The expertise engine is the section of code to find the expert for an outgoing electronic mail message. It is a critical component in an AGS application, and a major issue in the expertise engine is deciding where the message should be sent.

In general, the outgoing message is sent to either the node expert specified in the node, a set of local experts specified in a resource file, or a global expert list also specified in a resource file. The local experts might include an organizational help desk or a local "guru." The global expert list is a fallback list, in case no expert has been specified anywhere for this node.

A node will often have an owner or set of owners; this is the node expert. The node expert may, in fact, be a set of experts or even a distribution list. (This is true for the local and global experts as well.) The author (or modifier) of a node can set a node expert while editing if he wishes.

useLocalExpert	local node?	node expert	use
True	yes	set	node expert
True	yes	not set	local expert list
True	no	set	local expert list; forward
True	no	not set	local expert list
False	yes	set	node expert
False	yes	not set	local expert list; forward
False	no	set	node expert
False	no	not set	global expert list

AGS currently uses the heuristic in the table above for finding the right person. The heuristic uses three parameters, a Boolean value called `useLocalExpert`, a

determination of whether the node was written locally, and whether a node expert was specified specifically for the node. The Boolean `useLocalExpert` is a resource indicating whether there is a local expert that should take precedence over global experts. As mentioned, this would be set to `True` if there were a local help desk or some other intermediate expert.

Local nodes include information specific to the organization or site. They might include nodes describing specific organizational policies or even the location of binaries. The determination of whether a node is local (that is, it was written and is owned in the user's organization) is also made heuristically. If the organization in the node header matches the user's organization, then this node is assumed to be a local node. An alternative would be to check the node header's organization against a global organization. However, there may be a node contributed by a different organization from the global organization (MIT in the case of the X database). If the user's organization is not the same as the organization for the contributed node, it is assumed that the contributing organization has provided a node is globally available, and the message is sent to the global expert. If this node that is global but contributed by this organization, this algorithm still works since if a node expert is specified for the node, then the question or comment will still go to that node expert.

For some of the messages sent to local experts, the node expert or global expert lists are also sent in the Forward-To field.

The general heuristic tends to work in practice. However, the second heuristic, of determining whether a node is local, cannot distinguish between a node published to the world and one published for local consumption. An additional header field for each node would solve this problem.

D.1.2. Comment #2: the header fields for AGS messages.

Outgoing messages experts include a set of header fields that include the following. Copies to the users do not currently include the history list or other user information fields. (Note that sites that wish to provide users with exact copies of outgoing messages need to re-specify the `Mailer_Send_With_Copy_Callback`.)

`To:`	The outgoing mail address. This should be the appropriate set of experts and is normally set by the expertise engine's heuristic. (See Comment #1.)
`Forward-To:`	Optional additional address. This is normally set by the expertise engine if present. (See Comment #1.)
`Cc:`	Notification list. This can be any set of names and is specified in the code header file. Designed to provide a copy to the researcher.
`Subject:`	Fully qualified subject line (see below).
`Reply-To:`	Useful for some experts' mailers. See comment #5.
`Organization:`	The user's organization (specified in the application resource file)
`Version:`	AG version number (specified in the code header file).
`MessageType:`	Hardwired to `XQuestion` (for Information Lens type systems that use message type as a sorting field).

`HistoryList:`	History list of accessed nodes for the last 15 minutes or the beginning of the session (whichever is appropriate).
`NodeLocation:`	Name of the node.
`NodeLabel:`	Label of the node.
`NodeFile:`	Physical location of the node.
`UserInfo`	User information available from the system passwd file (using the system call `getpwuid`). This field should be removed when the user is supposed to be anonymous.
`From:`	User name (provided by mailer). This field should be removed when the fuser is supposed to be anonymous.

This is a semi-structured header, in the sense that additional information is provided in extra header fields. Systems similar to Information Lens or Object Lens (Malone et al. 1988, Lai, Malone, and Yu 1988) can operate on this additional information.

The "Subject:" line was designed to be used by a Question/Answer Tracking Server (ATS for short). The subject line provides the topic (for humans), the node name (for appending answers to QA nodes and the like), the node type (to determine whether automatic replacement is feasible), and the tracking stage (for example, Q for the user asking a question). In later uses, the Subject: line will also be the location for the assigned user-message-id to track a conversation between the person asking the question and the person answering the question.

D.1.3. Comment #3: sending a message

This is a callback to send a text question or message specified in the mailer popup (set up in `GeneralLostCallback`) and send it through electronic mail. It uses a temporary mail file (usually set to /tmp/agfile or ~/.agtmp). This callback uses the various Comm components to open the temporary file, write the header fields, write out the message body, send the message through the desired transport engine, and then close the temporary file.

This callback can specified externally (for example, from a resource file) using the value `GeneralQuestionCallback`. That is, the resource

```
AnswerGarden*questionCallback:  GeneralQuestionCallback
```

will result in this callback being invoked from the "okay" button of the mailer popup. The resource specification

```
AnswerGarden*questionCallback:  GeneralQuestionCallbackWithCopy
```

sets up the `General_Question_With_Copy_Callback` instead to send a copy of the question to the user as well as the expert. (Note this specification may be qualified by either node name or node type.)

D.1.4. Comment #4: customizing the mailer popup.

The communication service pops up a message box in which the user can enter his question or comment. The following changes to the user interface for this message popup can be made from the resource file.

1. Different user instructions. Any text can be specified.

2. A form for the user's response. For example, an application writer might want to specify a bug report outline as the text for a bug report node. (The current resource file has an example of this.)

The mailer popup can be customized for all nodes or for specific node names if you wish. Application writers wishing to customize the callbacks putting up the mailer popup should:

- Check whether the user is allowed to use mail. (Check on the global variable `global_info.use_mail`.) If the use_mail Boolean is set to false in the resource file, the application should bind the mailer button (usually "I'm unhappy") to a callback popping up a dialog box telling the user he cannot send messages. This can be done with `Comm_Excuse_Me`.

- Create the mailer_passback structure. The `Comm_Initialize_Mailer_Passback` routine can be used to do this. This structure is required for the various Comm routines.

Note that the mailer popup uses a Knedit widget. The Knedit widget allows all of the commands of the SBrowser node, including bold and italic text for instructional text to the user. As well, it supports buttons and all the other features of a Knedit widget.

D.1.5. Comment #5: determining the user's mail address

Determination of the user's return mail address is not straightforward. Many users do not receive mail on the same machine as they run Answer Garden. In addition, many machines do not return the proper domain name needed for complete addressing. AGS uses the system calls `hostname` and `gethostbyname` for a first attempt. If these do

not return the proper address, the application writer or site manager must fill in the `mailMachineName` (e.g., athena.mit.edu) or `domainName` (e.g., mit.edu) resources. The `mailMachineName`, which should be a full specification, overrides the `domainName` resource.

D.1.6. Comment #6: a sample communications callback

A writer adding a new Communication Service callback can use the existing building blocks. For example, the following code is the callback for sending a mail message out to the appropriate set of experts. This routine calls a series of `Comm` routines as well as a routine to gather statistics.

If the application writer wished to send additional header fields, he could write only an additional routine and place it between Comm_Write_Outgoing_Header and Comm_Write_Outgoing_Body.

```
void Mailer_Send_Callback(w,client_data,call_data)
     Widget w;
     XtPointer client_data;
     XtPointer call_data;
{
   XtPointer generic_pointer;
   MailerPassback *mailer_passback;
   char *mail_file;
   int rc;

   mailer_passback = (MailerPassback *)client_data;

   Stat_Button_Hit(w);
   Comm_Initialize_Mailer(mailer_passback,&generic_pointer,&mail_file);
   if (generic_pointer) /* ie, if fp != NULL */
      {
           Comm_Write_Outgoing_Header(mailer_passback,generic_pointer);
           Comm_Write_Outgoing_Body(mailer_passback,generic_pointer);
           Comm_End_Write(mailer_passback,generic_pointer);
           rc = Comm_Send_Mail(mailer_passback,generic_pointer,mail_file);
      }
   if (rc != NotAbleToSendMail)
      Stat_Sent_Mail(w);  /* for successful mail launch */
   Comm_Destructor(mailer_passback);
 }
```

In this code, a statistics routine is first called, and then the `Comm` routines are initialized. This is followed by the body of the routine that writes the header and body of the message, followed in turn by closing the temporary mail file and sending the mail out. The final routine clears up any dynamic memory.

D.2. Adding new node types

The definition of a `NodeType` (previously called a `SortaObj`) is found in AG.h. The compile-time definitions for the fields can be found in Globals.c. The current data structure includes the following fields:

field	description
`name`	Name of the node type. Names should not include blanks since they will be used in Xt resource files.
`create_function`	A routine to be called when each node is created.
`class_create_function`	A routine to be called when the first node is created.
`initialize_function`	A routine to be called at startup time. Initialization routines are called in the order of the data structures in Globals.c. Node type code should not assume initialization order.
`virtual_node`	A boolean to indicate whether the node type has a physical file associated with it.
`IR_search_function`	A routine to be called to present an information retrieval engine with the body text or the header fields of a node. Used primarily for virtual nodes.

pre_nodeinfo_edit_function	A routine to be called before the header and Node Service information is edited. May be used to pre-set fields.
post_nodeinfo_edit_function	A routine to be called after the header and Node Service information is edited. May be used to calculate and manipulate fields. Also used to pop up an editor for additional semi-structured fields.
pre_edit_function	A routine to be called before the editor is loaded. May be used to manipulate the physical file before editing.
post_edit_function	A routine to be called after editing but before the file is written. May be used to manipulate the data to be written to the physical file.
edit_function	A routine to edit the node data. If NULL, defaults to the AGS editor. Note that this may be a function that starts up an external editor, such as a graphics editor or Emacs.

There is one filled-in NodeType per node type. The NodeType array in Globals.c is processed when AGS starts up. Although it is currently safe to change code pointers during run-time, this is not guaranteed.

The types for the data structure are currently:

```
typedef Widget (*AGCreateProc)();
typedef void (*AGSProc)();

typedef struct
{
  char *name;
  AGTypeToken token;
  XrmQuark name_quark;
  AGCreateProc create_function;
  AGSProc class_create_function;
  AGSProc initialize_function;
```

```
  Boolean virtual_node;
  Boolean class_created;
  XtCallbackProc help_callback;
  XtCallbackProc question_callback;
  XtCallbackProc close_callback;
  AGSProc pre_nodeinfo_edit_function;
  AGSProc post_nodeinfo_edit_function;
  AGSProc pre_edit_function;
  AGSProc post_edit_function;
  AGS edit_function;
  String IR_search_function;
} NodeType;
```

An example of a filled-in `NodeType` is:

```
NodeType grapher_obj = {"Grapher", 0, NULLQUARK,
                                      Show_Grapher,
                                      NULL, NULL, False,
                                      NULL, NULL, NULL,
                                      NULL, NULL, NULL,
                                      NULL, NULL, NULL};
```

The `create_function` is called each time a node of this type is created or opened through AGS. It usually contains some amount of information processing code as well as setting up the Xt widgets for the node. It returns the widget id for the toplevel shell for the node. The `class_create_function` is called when the first node is created, and the `initialize_function` is called when AGS starts up. Usually the `initialize_function` is used to set up dynamic callbacks and one-time data initialization.

The `name` of the node type is used to access node type specific resources from the various resource files. It may be any string that does not contain blanks. The `virtual_node`boolean (set to either `True` or`False`) tells whether this node type is created using physical storage.

The three callback procedures can be set to provide default callbacks for the help, question, and close visual buttons that appear on each visible node. This is only one mechanism for setting these callbacks.

The various editor fields are called before and after the node's header fields are edited (using the NodeInfo editor as the default), as well as before and after the node's body text is edited (using the NodeFile editor as the default). Note that writers must insert a `post_nodeinfo_edit_function` if they wish to add an editor for handling their own header fields. That is, if a node type is a semi-structured object, containing additional header fields that only the node type knows, the author should provide a method for editing those fields. Writers may substitute their own editor (or an external editor) using the `edit_function` field.

The other fields are private. The `class_created` Boolean should always be set to `False`.

The most important function is the `create_function`. In the `create_function`, the node type writer will create all the Xt widgets required by the node (assuming no dynamically created widgets or popups). A `create_function` has the following API:

```
static Widget
create_function(w,node_info,geometry,node_type_string)
     Widget w;
     NodeInfoPtr node_info;
     XtPointer geometry;
     char *node_type_string;
```

The `node_info` parameter is merely passed to various AGS routines, one of which is required in the create_function while the others are optional. The `node_info` is used as an opaque type without any use of its internals. The widget identifier and the geometry specification are required to create an Xt top level shell. The `node_type_string` is provided for the information of the node type writer; it is the name of the node type as specified when setting up the `NodeType` structure.

The first widget created should be a `topLevelWidgetClass` shell (in the Xaw set). It must be created with a `XtCreatePopupShell` so the geometry management can work correctly. If the geometry specification is not `NULL`, it should be passed to the `XtCreatePopupShell` routine so Xt will use it to have the X server position the window correctly.

Besides the shell widget, the writer may create any number of additional widgets. The writer should remember that he will need to create a number of visual buttons as well. The routine `Util_Open_And_Bind_ButtonBox` creates the necessary number of buttons with the preferred system callbacks.

The system callbacks set in `Util_Open_And_Bind_ButtonBox` are dynamic callbacks. Thus, in the usual case, three visual buttons are created, bound to the `XtNcloseCallback`, `XtNhelpCallback`, and `XtNquestionCallback` callbacks. These system callbacks have been set to the `Dynamic_Close_Callback`, `Dynamic_Help_Callback`, and `Dynamic_Question_Callback` callbacks, which in turn default to `General_Close_Callback`, `General_Help_Callback`, and `General_Question_Callback`. By having a level of indirection in the calling sequence, the callback names can be exported to the resource file as Ascii strings, and therefore alternative callbacks can be specified by the application writer through the resource file.

These visual buttons (which may differ in number from three depending on the application) are created in a Form widget (in the Xaw set). The node type writer is responsible for placing this Form in the appropriate position in the node.

If the `Util_Open_And_Bind_ButtonBox` routine is used, application writers can easily add additional visual buttons to all node types without changing the node types'

code. In fact, all that needs to be done is to specify the number of buttons in the resource file and their associated dynamic callbacks.

The node type code should call `XtRealizeWidget` (to allow the widgets appear on the screen) and then `XtPopup` (to make the widgets appear on the screen since the shell is a popup). The final step is to return the shell id to the calling routine; AGS uses this shell to later find the node.

The final calling sequence, then, is:

```
AG_Geometry(shell,node_info);
XtRealizeWidget(shell);
XtPopup(shell,XtGrabNone);
return(shell);
```

D.3. Node Service

The Node Service was not built to be extensible. However, a large number of routines are available to help node type writers. In addition, the Node Service file (of existing nodes) is editable. This section describes both.

D.3.1. The Node Service file (AGNodeFile)

The AGNodeFile contains one record for any node that has been created permanently. This file is the index for maintaining a large, external information database. (Temporary nodes are maintained within the Node Service only for the lifetime of the AGS session.) The format is

```
node name    #node type    #label    #file service    #data
```

If there is no physical location, the default is the same name as the node name. The '!' character is the comment character in the AGNodeFile. A line begun with '!' will not be processed, for example:

```
! This is a comment line.
```

And, AGNodeFile can include other node files. To include another file, use:

```
#include filename
```

Thus several Answer Garden information databases can be treated either separately (perhaps for editing) or together (for the end-user's perspective and use).

Several things should be noted about this file:

- Since the AGNodeFile is Ascii, it may be edited in any suitable text editor. It may also be automatically generated or appended.

- It is assumed that the AGNodeFile can be read and written in the directory specified by the Directory resource in the Xt resource file. This can be any directory including the app-defaults directory (for the resource file itself) or the directory containing the information nodes.

- Several routines modify the AGNodeFile. The `NodeService_Add_Node`, `NodeService_Replace_Node`, and `NodeService_Delete_Node` routines all modify the permanent AGNodeFile. The existing AGNodeFile is backed up first in AGNodeFile.bak and then rewritten. Writes are elementary; that is, they are not cached so state is maintained in case of system failure.

The default File Service assumes the data field is the location of the physical file, represented by an Ascii string. If the string begins with a '/', it is assumed to be an absolute address. If it does not, it is assumed to be relative to the Directory resource in the Xt resource file. Note that the default File Service assumes one node per file. (The Ascii node type is not an exception. The default File Service still assumes one file per Ascii node; however, it also assumes there is no AGS header.) Alternative File Services could, however, place many nodes per file. As an AGS application grows to include thousands of nodes, this will be required for efficiency.

D.3.2. Node Service routines

Node Service routines are often useful in node type code. They include routines to handle general I/O for nodes, to access and modify AGNodeFile values, and to access and modify header fields for individual nodes.

As a note, routines that access the Node Service, even indirectly, often pass an opaque structure pointer called a `NodeInfo`. Application writers should not access its contents directly. `NodeInfo`'s can be obtained from the node's name or its shell window id. A `NodeInfo` is passed to the create_function in the node type code as well.

The available routines include:

`NodeService_Add_Node`
`NodeService_Replace_Node`
`NodeService_Delete_Node`
	Routines to add, replace, and delete nodes from the internal Node list and to write out changes to the AGNodeFile. Replacement changes the node fields -- node name, label, type, and location. (Note location was an earlier name for the data field.)

NodeService_Add_Temporary_Node NodeService_Replace_Temporary_Node NodeService_Delete_Temporary_Node	Routines to add, replace, and delete nodes from the internal Node list. No changes are made to the AGNodeFile.
NodeService_Request_Fields NodeService_Request_By_Name	Routines to obtain all node fields, either by NodeInfo or by name. These node fields are usually from the AGNodeFile file. (Temporary nodes, however, will have valid node fields without an entry in the AGNodeFile.)
NodeService_Get_Location NodeService_Get_Label NodeService_Get_Node_Name NodeService_Get_Type NodeService_Has_Node_Changed NodeService_Get_Shell NodeService_Get_Text	Routines to obtain specific node fields including those for open nodes. These calls should be used instead of direct File Service calls. Some of the fields are appropriate to only nodes with open files. (Note location was an earlier name for the data field.)
NodeService_Get_Nonpredefined_Headers	Routine to get additional fields defined for only this node type (or even node). That is, this routine provides access to the additional header fields in semi-structured node types.
NodeService_Request_Node_Label	obsolete routine

NodeService_Free_Buffer	Routine to request that the File Service free any allocated memory for the node's text data. (This routine will not free memory within a node's widgets.)
NodeService_Find_Open_Node	Routine to determine whether a node is open (i.e., on-screen or iconified).
NodeService_Check_NodeInfo	Routine to check whether a NodeInfo is valid.
NodeService_Set_Defined_Headers NodeService_Set_Nonpredefined_Headers	Routine to reset the node header fields (e.g., the fields for node expert or author). Usually followed with a Save of the node.
NodeService_Get_String_From_Type NodeService_Node_Int_From_Type NodeService_Node_Type_From_Int NodeService_Get_Type_From_String	Convenience routines to get the name of the node type from the Type value (or the reverse).
NodeService_Get_First_Member NodeService_Get_Next_Member	Routines to traverse the Node list. Can be used in information retrieval engine routines.
NodeService_Open_File NodeService_Save_File NodeService_Save_New_File	Routines to open or save nodes into their physical files. These routines should be used instead of the File Service routines.

NodeService_Register_Open_Node NodeService_Register_Closed_Node NodeService_Register_Closed_Edit NodeService_Register_Open_File	Routines to register open nodes or deregister closed nodes. May be used by node types that create virtual nodes. Normally called by AG_Create_Node, AG_Close_Node, or editor routines.
NodeService_Get_FileInfo NodeService_Get_EditFileInfo NodeService_Get_EditNIPInfo	Used internally by the AGS editor.

D.4. Miscellaneous utility calls

AGS has a number of miscellaneous calls that can be used from any node type or service code. Three major routines that may be particular valuable in AGS applications are:

AG_Create_Node	Routine to open a node given a node name. This routine can be called within a node type's Initialize procedure.
AG_Close_Node	Routine to close a node given a node name. This destroys the node rather than iconifying or unmapping it from the screen.

AGS_Get_Resource	Routine to retrieve an Xt resource from the Xt resource database given a resource name and resource class. The resource is fetched on the strings:
	`AGS_app.node_name.resource_name`
	and
	`AGS_app.node_type_name.resource_class`
	where the `node_name` is the name of the node and the `node_type_name` is the name of the node type. The calling code specifies the resource_name and resource_class. AGS substitutes the name of the program for AGS_app.
	
	Examples of such strings are:
	`AnswerGarden.RootGrapher.stack`
	and
	`AnswerGarden.Grapher.Stack`
	to determine the correct window stack for the first grapher node in the Answer Garden application. The code then uses the more specific value present in the resource database (which results from the resource file).

Other routines that can be used in AGS applications or elsewhere include:

strcmplo strncmplo	Routines to compare two strings (or substrings) without regard to case. These routines do not exist in some Unix flavors, or they change names.
AGstrcpy AGstrcmp AGstrlen	Macros that implement strcpy, strcmp, and strlen with an explicit check for the NULL pointer. Keeps Sun workstations from melting.
Remove_Tabs Remove_Control_Marks Util_Remove_WhiteSpace Util_Remove_Leading_WhiteSpace Util_Remove_WhiteSpace_Until_LineEnd	Routines to handle string buffers. Used by parsers.
Util_Search_Array	Routine to search through a generalized array for a string member.
Util_Move_Token_To_String	Routine to handle parser token and place in a C string.
Util_Replace_String	Routine to replace the contents of a string in allocated memory. Frees up previously allocated memory and allocates new memory.

Util_Get_Timestamp Util_Get_And_Check_Timestamp Util_Get_Date Util_Get_LocalDate	Routines to get time and date from the system clock.
Util_Get_UserName Util_Get_UserName_With_Machine	Routines to get the user's name with her machine name. The latter routine is not trivial and makes a large number of assumptions.
Util_Get_UserInfo	Routine to read the system password file to get the user information within it.
Util_Extended_Inspection	You deserve a prize for reading this far. Send electronic mail to me (currently ackerman@ics.uci.edu), and I will send you a box of crackerjacks.
User_History_List_Add_Member	Routine to add a history list item.
User_History_Print User_History_Make_List	Routines to either pretty-print the history list or to place the history list in an array suitable for display.
Util_Find_Shell_Name	Routine to determine the name of the node given a shell id.

There are also a series of statistics routines that write out user statistics to a file. These routines may be turned off by compiling with the NO_STATISTICS flag.

The first three calls are in AG.c; the other calls can be found in Util.c. The macros can be found in AGmacros.h.

D.5. Customization

This is an admittedly incomplete list of customization resources available in AGS:

resource	description	comments
`editMode`	whether nodes can be edited by the user	Site managers can also change the keystroke and mouse bindings to prevent the edit menu and/or turn off editing capabilities when compiling.
`directory`	location of the information database files (for relative addressing in the application)	A string naming a file that starts with a '/' is assumed to be an absolute address.
`statisticsFile`	if statistics are to be kept, this is where they are written	Should be a valid file name.
`mailerType`	what mail process is being used (binmail, mail, or mh)	
`globalExpertList`	a backup list for global experts if a node does not have one	
`organization`	the organization for the user (used in outgoing communications)	
`useMail`	allow outgoing communication for the user?	Note that this is settable per node so there can be some nodes without mail ability.

`useLocalExpert`	is there a local expert (such as a help desk person)	The default value is False.
`localExpertList`	a backup list for the user (used in outgoing communications)?	
`mailName`	user's e-mail address (usually inferred by the program, but may need to be set explicitly under certain conditions)	This is the first of several resources to try to construct a valid mail path.
`mailMachineName`	the machine name for the mail server if different than the user's machine	
`domainName`	the domain name for the mail server if different than the user's machine or if it cannot be inferred	
`nodeExpert`	the default expert list for new nodes	
`questionCallback`	the name of the callback for user communications	This can be overridden to change the functionality of the application. The default value is a dynamic callback.
`helpCallback`	the name of the callback used for help	This can be overridden to change the functionality of the application. The default value is a dynamic callback.

`closeCallback`	the name of the callback used for closing a node	This can be overridden to change the functionality of the application. The default value is a dynamic callback.
`maxGrapherHeight`	maximum height for Graphers (if larger, they have a vertical scrollbar)	
`maxGrapherWidth`	maximum width for Graphers (if larger, they have a horizontal scrollbar)	
various	text for the mailer pane	Set as with any text widget.
`tracking server address`	location of the tracking server for outgoing questions	Not used in the current version
`startupNode`	the node that is opened first when an AGS application starts up	Default is the Control node
`root`	the node that is opened when the third-from-the-left visual button on the Control node is pressed	
`rootGrapher`	the node that is opened when the fourth-from-the-left visual button on the Control node is pressed	