Information Systems Frontiers 2:3/4, 333–347, 2000

Collaborative Support for Informal Information in Collective Memory Systems

Mark S. Ackerman and David W. McDonald
Department of Information and Computer Science University of California, Irvine Irvine, CA 92697
E-mail: {ackerman, dmcdonal}@ics.uci.edu
Website: http://www.ics.uci.edu/CORPS/ackerman.html

Abstract. **Informal information, such as the expertise of an organization or the workarounds practiced by a community, is a critical part of organizational or collective memory systems. From a user-centered perspective, a user merely wishes to get his work done, and to do this, he must solve his immediate problems. We have examined how to incorporate this problem solving into a collective memory, as well as how to incorporate the learning that accrues to it or from it. We report here on two systems, the Cafe ConstructionKit and the Collaborative Refinery, as well as an application, Answer Garden 2, built using these two systems. The Cafe ConstructionKit provides toolkit mechanisms for incorporating communication flows among people (as well as agents) into an organizational memory framework, and the Collaborative Refinery system provides mechanisms for distilling and refining the informal information obtained through these communication flows. The Answer Garden 2 application demonstrates the utility of these two underlying systems.**

Key Words. **computer-supported cooperative work, organizational memory, community memory, corporate memory, group memory, informal information, incremental formalization, information refining, information retrieval, information access, information systems, CMC, computer-mediated communications, help, collaborative help, CSCW**

Introduction

Knowledge resources exist everywhere within organizations and communities. There is great hope that in addition to formalized and standardized resources that these collectivities regularly use (e.g., databases, standard operating procedures, and software processes), other knowledge resources can similarly be tapped. However, the expertise of an organization or the everyday communication flows among people may be harder to describe and systematize. Yet, such resources are critical to include within a collective memory framework.[1]

The work described here investigates the collaborative mechanisms and architectures that could enable informal flows of information and communication to be included within collective memory and knowledge management systems. In this paper, we use the term ''informal'' in the computer science sense. It is not that this information is less valuable. Instead, we simply mean that informal information is more difficult to describe, evaluate, and systematize than other information. For example, problem-solving communication among product engineers is often critical as production nears. Computational support for this type of communication (and any learning that accrues to it or from it) is substantially different than the support required for data mining. In this, we follow Shipman and McCall [1] in their call for augmenting and supporting informal information flows. The act of collecting and disseminating informal information is a natural starting point for the iterative, evolutionary process of transforming informal information to working knowledge.

From a user-centered viewpoint, the problem of distributed help demonstrates the power of including informal information; distributed help displays many features of a collective memory problem. In general, many user communities have a problem with delivering help and user assistance, but this problem becomes especially acute in distributed communities. For example, astrophysics users are spread across the world, there are seldom many of them in one site, and

they may have need of relatively specialized help. Unfortunately, an astrophysicist is often left to sift through reams of documentation, find his way through mail archives, or pursue answers through trial and error. Normally, one attempts to examine the documentation or other help sources, and if there are others using the same software, one can wander out into a hallway in search of friendly colleagues.

For an astrophysicist, the problem is obtaining sufficient information to continue the task. For the user, a duality exists between the problem of help and the use of collective memory. Within an organization or community, individuals' information seeking requires finding the right part of the collective memory. Typically, collective memories include information repositories (e.g., information databases, filing cabinets, and documents). They can also include people (e.g., other organizational personnel) [2]. The collective memory to which a user has access includes at least the documentation, the system programmers, and his colleagues. However, the user may have great trouble finding the right piece of the collective memory that has the answer he needs. In other words, his access to the collective memory should be augmented. Some of that access will be to formalized information (such as to documentation or other knowledge sources); however, some of that access must also be to informal information (such as appropriate expertise or organizational work-arounds)

The distributed help problem, then, suggests the utility of three types of support for informal information in a collective memory:

- *Including human resources (i.e., people) in a collective memory.* Our view is that collective memory should not be restricted to static repositories. Information technology can support organizational memory in two ways, either by making recorded knowledge retrievable or by making individuals with knowledge accessible. Our emphasis here is on supporting both to form a dynamic enabler of organizational and community processes.
- *Incorporating communication flows*. While other research efforts (e.g., Foner [3]) focus on finding *the* correct person to answer a query, our emphasis here is on augmenting existing social mechanisms using a variety of computer-mediated communication (CMC) facilities. People already use bulletin board, chat, email distribution lists, and other CMC mechanisms to seek information. These social mechanisms have a long history of providing useful information support, but have not been fully incorporated into a collective memory framework.
- *Reducing the information in a collective memory to manageable proportions*. For a collective memory, we are primarily interested in distilling captured information into useful knowledge. Too often, users cannot cope with the vast amounts of material (some extraneous to the topic) that occur with informal communication flows. We would like to find ways to boil this information down into manageable amounts.

Meeting these basic requirements requires attention to the issues of *graceful escalation* and *collaborative refining* of answers (described fully below). We report here on the technical mechanisms required for this support for informal information. In this paper, we describe two computer-supported cooperative work (CSCW) systems, the Cafe ConstructionKit and the Collaborative Refinery. These systems provide different facilities for collective memory: The Cafe ConstructionKit augments the social network using CMC facilities to more easily obtain informal information, while the Collaborative Refinery helps distill informal information once it has been collected. We also report on an application that uses these two systems. Answer Garden 2 (AG2) is presented here to demonstrate the utility of both underlying systems in constructing a useful organizational memory application.

The paper begins with the two underlying systems. The Cafe ConstructionKit is briefly described (having been also briefly described elsewhere), and the Collaborative Refinery is presented in detail. We then follow a detailed description of the AG2 application, highlighting why it is an important application for organizational memory, as well as how useful the two underlying systems were in its construction. The paper concludes with a description of similar systems and with a summary.

Cafe ConstructionKit: a Toolkit for Sociality

> In the old days, we used to sit all in one room around the Mini. Everyone knew what was going

on. If I had a problem, I could just ask. Now we all sit in our separate offices [with workstations] and no one talks. . . .—Astrophysicist.

The Cafe ConstructionKit (CafeCK) is a CSCW toolkit for supporting sociality and information use in collaborative environments [4]. CafeCK was designed to simplify the construction of applications such as information filters, locator services, digital libraries, and other CSCW projects. The toolkit can be used to add a variety of CMC and information components, such as information retrieval mechanisms, email readers, bulletin boards, synchronous talk, and socially constructed spaces, to many applications.

To do this, CafeCK provides a set of reusable objects that include message transport for asynchronous and synchronous communication (including a Zephyr-like chat system, NetNews, and email), parsing for a variety of semi-structured protocols, private and public channels for narrowcast communication, message filters, user interfaces, and message retrieval by a variety of semi-structured methods. An application writer selects from these components, and after adding a small ''glue'' layer, can create a set of distributed processes to handle information retrieval, information access, or electronic communications.

CafeCK is implemented in C++ , Tcl, and Tk. Tcl serves as glue between the computational objects (Fig. 1); each object exports a number of Tcl verbs that allow it to be used by an application writer. By selecting from the set of available components (or by extending it) and by writing a simple Tcl program, an application writer can create a set of distributed processes to handle information retrieval, information access, or electronic communications (Fig. 2). By configuring the objects and providing the suitable Tcl program, *any* application can include the functionality of bulletin boards, chat systems, and electronic mail filters.

Because of this, CafeCK has an open architecture, unlike many information retrieval systems. It has a truly distributed architecture, where application writers can ''snap together'' applications. Unlike client-server architectures (e.g., that of Lotus Notes), CafeCK enables applications with many processes spread over many processors. The utility of this will be shown below with the Answer Garden 2 application.

CafeCK, in summary, augments the social network by providing mechanisms for routing and handling flows of communication and their informal information. The Collaborative Refinery system provides mechanisms for distilling that informal information once it is obtained for later use. Collaborative Refinery is described next.

Collaborative Refinery: Refining a Memory Repository

On my shelves were tons of unwinnowed material. . . . In the present shape it was of little use to me or to the world. Facts were too scattered; indeed, mingled and hidden as they were in huge masses of debris, the more one had of them the worse. . . . To find a way to the gold of this amalgam . . . was the first thing to be done (Bancroft [5], 1891, p. 135).

The Collaborative Refinery (Co-Refinery) provides mechanisms for handling individual and joint information spaces. Central to the Co-Refinery is the ability to individually and collaboratively view and manipulate information collections. It is especially useful in situations where an individual or group wants to refine and distill collections of materials such as shared artifacts, frequently asked question (FAQ)

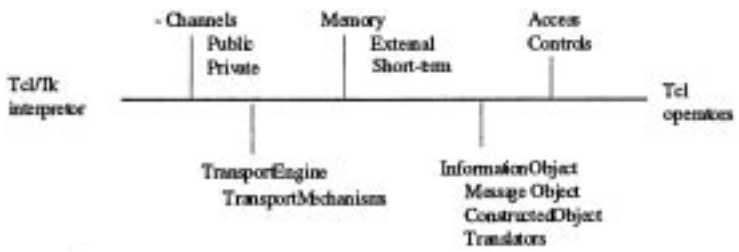

Fig. 1. *Tcl as the ''glue'' mechanism in CafeCK.*

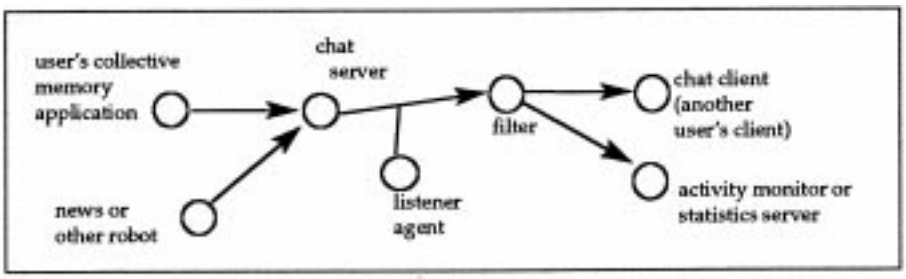

Fig. 2. *CafeCK application architecture (sample).*

lists, discussion list digests, and the like. The system will be described extensively below, using FAQ creation as its example.

The collaborative refining process

The Co-Refinery system supports an authoring process that includes four general activities: collecting, culling, organizing, and distilling. We assume that any of these activities, as well as authoring, may be done iteratively or in any order. Each activity is clearly important, although the major research contribution here is the support for collaborative distilling:

- *Collecting* is the phase in which information is gathered. In Co-Refinery, automatic collecting can be set up for information streams such as Usenet, synchronous chat channels, or distribution lists. In addition, manual collecting allows individual items to be submitted through the system directly or by e-mail. Collecting places items into an archive for later use in creating the FAQ.

  For example, for a FAQ about the Java user interface toolkit, one might wish to collect a set of Usenet threads about bugs and features of the various user interface components.
- After collecting the material, one must cull the collection for interesting material, and the lesser material must be discarded or ignored. *Culling* is a selection mechanism, identifying themes or threads that occur within a collection. A sizable reduction of material may be possible through culling the collection, making subsequent organizing and distilling easier. Culling reduces the apparent size of the archive, although in our current implementation, items are unreferenced rather than deleted.

  In the FAQ example, one might wish to cull out off-topic messages or messages without responses.
- *Organizing* allows one to group materials according to some classification schemes so to enhance their retrievability and understandability. Our current prototype relies heavily on outlining, user-defined indexing, and keyword indexing, but other classification mechanisms are clearly possible. In Co-Refinery, retrievability is enhanced by making the culled subset a fully identifiable element in the collection. In this way, organizing results in an addition to the collection.

  Using the Java FAQ example, one might organize message threads about the same user interface components together.
- The most important part of refining is *distilling*—boiling down the existing (and culled) materials in order to uncover the answers or knowledge. As with chemical or liquor distilling, the results should be a more concentrated or concise form of the original information. Creating or editing a summary or synopsis, for example, removes much of the tedious work of wading through extraneous or erroneous information.

  Distilling is described further in the next section.

Distillates

The result of distilling is a *distillate*. Our major emphasis has been on providing support for users to distill and refine the material, and so Co-Refinery generally provides intermediately processed material that can then be edited or authored more easily. The system currently supports a number of sample distillates. One useful intermediate distillate consists of merely concatenating selected Usenet messages. This allows an author/editor to further prune the selected information into one final distillate consisting of an authoritative answer; this behavior is very similar to what people currently do when they compile

a FAQ. Another sample distillate is temporally based; items in it vanish after a short period of time. Technical hotlines often have runs of questions, and this distillate solves the problem of communicating the temporarily needed answers.

Co-Refinery users specify distillate types by picking a *catalyst* to generate the appropriate rough distillate. A catalyst is a Tcl script that fits the combination of criteria supplied by the user. Catalysts extend the range of distillate types currently supported by Co-Refinery, since new catalysts can be added simply to the system.

Fig. 3 shows the results of refining, and can be considered as a snapshot of an iterative process. In Fig. 3, the leaves are raw information, perhaps Usenet or e-mail messages. The small document icons to the left of the outline represent a distillate. Authors and editors have created distillates for five of the threads, winnowing the material into answers for frequently-asked questions. One of these answers is shown in Fig. 4. (This is the heart of the AG2 application, below.) After being finished, some distillates can then take the place of the raw material in the archive. Note that some of these distillates could be intermediate; i.e., not shown to the public because they are under construction. Additionally, all distillates can be iteratively revised.

As mentioned, we assume that users move fluidly among these activities. We also assume that the refining is done iteratively and incrementally. Indeed, refining is not a complete solution to authoring. There will always be effort required to compose explanations of complex technologies and tasks. Refining reduces the overhead for that task, and simultaneously reduces information overload.

As well, we do not believe that all materials will be refined. It does not make sense in all cases to move data through information to organizational knowledge. For example, information that has a short-shelf life will not be refined, especially if the information also has a high throughput velocity. The cost would be prohibitive. Therefore, we have attempted to define some distillates that are suitable for temporally limited information. These distillates do not boil down the material, but they do make finding and retrieving the material easier.

In summary, we developed Co-Refinery to enable groups of people to collaborate in jointly or individually building answers and information repositories over time. The system allows groups of users

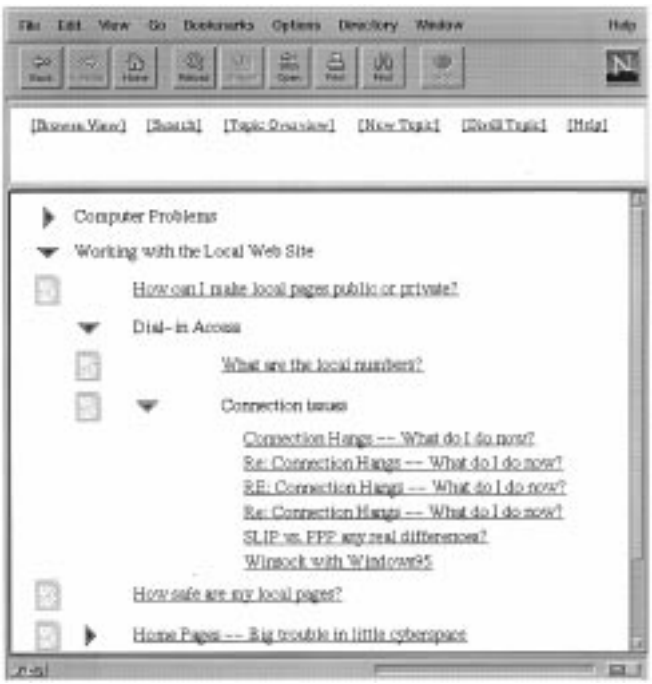

Fig. 3. *Co-Refinery screen with raw information nodes and distillates.*

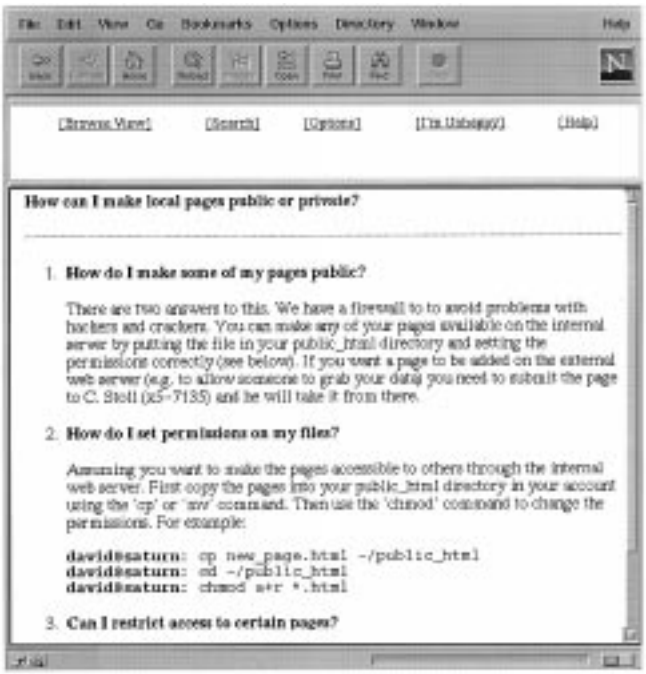

Fig. 4. *A Co-Refinery distillate.*

to interact in creating shared information artifacts and a common information space. Co-Refinery represents an alternative to many current attempts to completely automate the refining process, although the architecture allows automation mechanisms to be included. If collective memories are to include informal information (especially from email, chat, and bulletin board systems), it is critical to provide mechanisms for this distilling process. Otherwise the sheer volume of informal information obtained and stored will prevent people from later effectively using this stored information.

Co-refinery architecture

Co-Refinery components include objects for managing a collection archive of materials, constructing and maintaining a database of relationships for those materials, and generating a suitable presentation. Co-Refinery is implemented in C++, and the Web portion relies upon HTML 3.0 and Netscape HTML extensions.

Fig. 5 shows an overview of the Co-Refinery architecture and how its components interact. Within Co-Refinery, a collection is stored in two parts. The archive includes all of the items within a collection, currently stored as individual items in the file system. These items could include email, news, topic descriptions, distillates, or any other type of representable item. There are no logical restrictions on the types of items that can be placed in the archive. However, some types (e.g., audio, video) require additional meta-data to be effectively manipulated, distilled, and presented. Collaborative Refinery currently includes a sample Usenet news archiver.

Co-Refinery's database contains abstracted information about each of the content items in the archive. The database also contains information about the relations among the archive items. The sample Usenet archiver also creates the appropriate database changes.

Collaborative Refinery's architecture separates an intermediate representation from the final presentation. This allowed us to construct a presentation backend that could be modified to support other presentation schema such as text-only, Notes' text format, or a markup. Output from Co-Refinery's presentation generator is currently HTML, files, or e-mail.

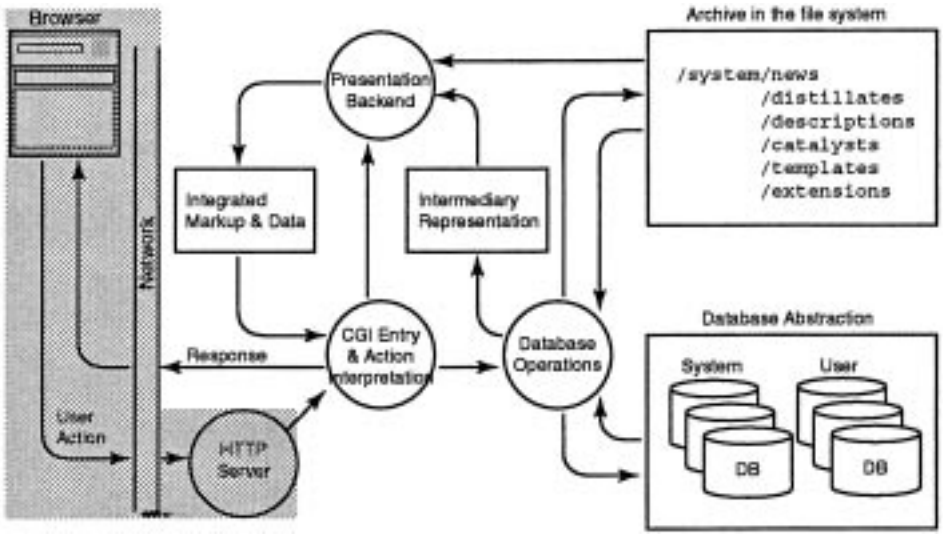

Fig. 5. *Co-Refinery architecture.*

Answer Garden 2

To test the usefulness of these support facilities for constructing informal information and collective memory, we constructed a prototype application to provide help to distributed users and customers. As previously mentioned, distributed help is an important problem for many users: it helps them remain on task at critical times. Any community, institution, or organization of any size often has a problem with answering questions in a timely manner. Yet, solving problems and completing tasks are often dependent on obtaining timely answers to specific questions.

What we would like is a surrogate for the friendly local expert or the hallway talk that people use to find customized help. Instead, we seek a system solution that reduces the cost of providing help, while at the same time provides better help than existing systems. To help users, we would like to have a system to narrowcast a question to the collectivity's experts or other users of the product. Such a solution must avoid the broadcast problem of flooding a community or organization with thousands of questions. To reduce the community's or organization's cost, we would like to allow users to easily browse through an information database (such as one built from frequently-asked questions, or FAQs, formed by the interaction of experts and users) and to have users easily help one another. These requirements form the basis of our system, Answer Garden 2.

Previous work, a system called Answer Garden reported in Ackerman and Malone [6] and Ackerman [7], allowed organizations to develop databases of commonly asked questions that grow ''organically'' as new questions arise and are answered. Answer Garden 2 (AG2) continues this work. It is a second-generation application for the same collective memory problem, investigating some of the issues encountered in field studies of the original system. A new architecture, enabled by the two systems described above, provides a customizable and adaptable set of software components that allow a variety of organizational and informational configurations. Furthermore, it offers a generalized solution to the problem of finding help for any information system.

Answer Garden

Before discussing AG2, and our subsequent investigations, it will be useful to briefly describe Answer Garden. This application still plays an important part in our current work.

Answer Garden supports organizational memory in two ways: by making recorded knowledge retrievable and by making individuals with knowledge accessible. In the standard configuration of Answer Garden, users seek answers to commonly asked questions through a set of diagnostic questions or other information retrieval mechanisms. Figs. 3 and 4 above showed Answer Garden reimplemented in the World Wide Web. (Other, third-party versions exist in

Fig. 6. *Answer Garden functionality in the Web.*

the Web [8] and in Lotus Notes.) Diagnostic questions guide the user through Web pages. Alternatively, the user may use a number of other information retrieval mechanisms to find the pages that may contain the answer.

If the user cannot find an answer or the answer is incomplete, the user may ask the question through the system. (This is the result of the user pressing the ''I'm Unhappy'' link in Fig. 4.) In the original Answer Garden, the system would then route the question to an appropriate human expert. This has been changed in Answer Garden 2 as will be discussed below.

In the original Answer Garden, the expert would then answer the user through electronic mail. If the question was a common one, the expert could insert the question and its answer back into the information database. Thus, users were not limited to the information in the system; if the information was not present, they could tap the organization's experts. As a result, the organization would gain a corpus of information, an organizational memory. Users could obtain expert advice without a high organizational cost. Other interesting properties of the system are discussed in Ackerman [7].

Field studies of Answer Garden's use [7,9] uncovered a number of issues. While the system was held to have worked, two issues were uncovered that are critical to the success of similar memory or help systems:

- Tying the social network into the system in a more natural manner. Answer Garden's dichotomy between experts and users was problematic. While there was nothing in the underlying technology to force this dichotomy, it was a simplifying assumption in the field study to have separate user and expert groups. Real collectivities do not function this way. Most people range in their expertise among many different skills and fields of knowledge. People can know things about systems and their tasks, even though they may not be able to answer specific questions. We would like to allow everyone to contribute as they can, promoting both individual and collective learning. However, mechanisms to allow each person to contribute must not overwhelm the other people who use the system. For example, broadcasting each question to every person in an organization or community will fail. AG2 offers several mechanisms to ameliorate the overload problem while allowing and providing for a range of expertise.

- Providing for the contextualization of answers, thus providing for the user's understanding of an answer. In the Answer Garden field study, most users either did not need contextualized information or were able to contextualize it themselves. However, a significant portion of the participants did need more context.

  For a user, the answer to a question may be present in the documentation. However, he may lack the required expertise to infer an answer or to even use an explicit answer without additional situational information.

  Providing the proper context is, unfortunately, difficult. We will return below to one way of potentially providing this context at low cost. Our mechanism also ameliorates the problem of providing answers at the right level and length of explanation.
- Easing the authoring burden. To obtain answers, the cost of authoring must be minimized. Furthermore, authoring answers, as an individual activity, promulgates the distinction between experts and everyone else. Composing content for answers takes as long as any writing takes, but we may be able to ease the mechanics of the process.

One might expect these issues to become increasingly problematic as the information becomes non-technical or the users become less sophisticated in the domain. For example, only astrophysicists can understand the scientific analysis tasks that create their questions about software systems. Astrophysicists will vary in their computer expertise, but few wish to spend time inferring the answer from substantial system documentation before continuing with their analysis tasks. And, the programmers who must currently compose the answers may not even understand the domain or its tasks.

Additionally, the field studies uncovered a number of technical issues, such as the need to use varying ''front-end'' systems such as the Web or Notes, to consider additional methods of finding experts, and to find better ways of maintaining the information database. These technical issues and the above social issues led us to reconsider the architectural design.

Answer Garden 2 (AG2)

Using CafeCK and Co-Refinery, AG2 consists of a second-generation architecture for organizational memory and collaborative help support.

These two components are used together as in Fig. 7. Raw information comes into the collection archive through CafeCK processes (such as News filters), by being explicitly sent to the archive through e-mail, or through filtering agents. It may be partially processed, and then is moved into the information database. At snap-shots or upon explicit queries (depending on a site's tailoring of AG2), the materials are built into Web pages, Notes documents, or flat files. In turn, the AG2 Web or Notes clients can send mail to CafeCK back-end processes that then handle the details of obtaining help. These CafeCK help processes will be described next.

There are several advantages to this architecture. First, the design cleanly separates the front-end of

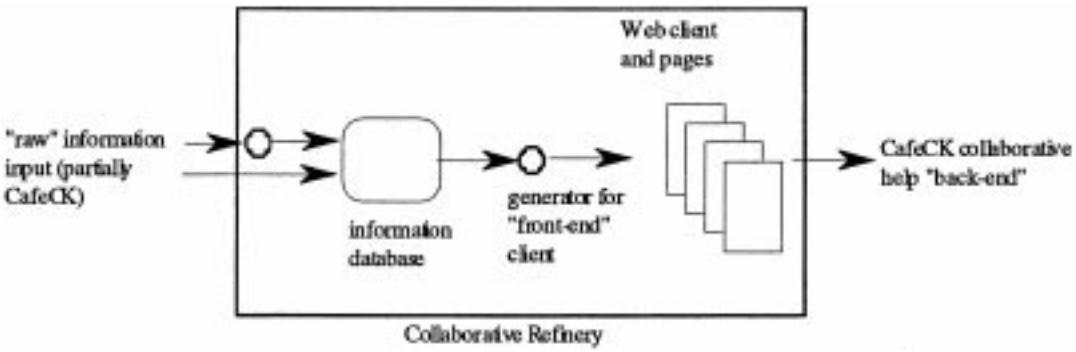

Fig. 7. *Answer Garden 2 (AG2) architecture.*

Answer Garden (i.e., the user client) from back-end needs. More importantly, it also decomposes the Answer Garden functionality into a set of distributed software services. This provides a high level of organizational flexibility; the services can be mixed and matched in order to provide additional flexibility. For example, by attaching an anonymity service, users of the system can send their questions anonymously. By attaching an anonymity service at another point in the distributed architecture, the experts answering the questions can also be anonymous. Or by not having an anonymity service at all, all users and experts can be known to one another.

Finally, the change in architecture makes much of the help functionality possible from any information system. This work, then, is generalizable to any information system.

The problem as a duality

AG2's ''back end'' can be viewed either as a collective memory system or as a distributed help system. Each of these views is the dual of the other.[2] By considering the ''back-end'' organizational memory problem in terms of its dual, collaborative help, we believe we have found mechanisms for reducing the context problem.

Above, it was noted that an open research issue was how to alleviate the users' need for contextualized information in solving their problems and finishing their tasks. Using collaborative help in a controlled manner can ameliorate this issue. Collaborative help functionality also provides help to users at their own explanation level and potentially with iterative diagnosis.

Staying local

Providing help from other people—such as colleagues on the same hall or other group members—allows people to seek help first from the people most likely to know the local context. Colleagues can judge a person's abilities, expertise, and situation, and can try to provide suitable information to solve the person's problem. Local participants are also more likely to provide information, since personal social ties are key motivators in providing assistance [10,11].

Always asking one's colleagues is, however, problematic. First, it is still costly to ask other people. AG2's repository of previously asked questions and frequently required information, however, attempts to reduce that problem. More importantly, one's colleagues may not know the answer. While staying local is important, it can also be organizationally dysfunctional [12] when there is no local expert available. In these situations, a means for escalating answers past the local group is required.

Graceful escalation

Using the facilities of CafeCK, we were able to simply construct an escalation agent for questions in AG2. This component allows the user to decide what to do if the question is not answered. It allows the user to consider whether to get answers from chat systems, bulletin boards, software agents, or other people.

The typical way that we envision the system being used is to gracefully escalate the help request until it can be answered. Because the escalation agent is a CafeCK process, the escalation can be quite flexible. The agent is currently programmed to follow organizational rules on the order of escalation, although this is under user control. It would be a simple matter to change this to provide different organizational rules, complete user control, or even heuristics (such as avoiding the chat facility when no other users are logged into their machines). No doubt other mechanisms could be found; this is a potential research question.

In our prototype, the user poses a question through his application. In Fig. 8, the user client is an AG2 front-end, but it can be any application that has asynchronous or synchronous communication capabilities. The application merely connects to a CafeCK process through, for example, e-mail. This CafeCK process, the escalation agent, is semi-autonomous, since it can be triggered either by the user or automatically.

As an example, imagine the following scenario. (Note that the underlying components for AG2 provide an enormous flexibility, so users' actual practices can vary widely from this.) A user, Fred, has a question about his data analysis package, and he would like to know how to correctly massage his data. He first looks through the existing questions and answers, either in a stand-alone AG2 information database or in an AG2 component of his data analysis application. Assuming that the answer is not there, or that he does not understand how to apply the information that is there, he composes a question and mails it off through his Web browser.

Instead of the question going to an expert, as it

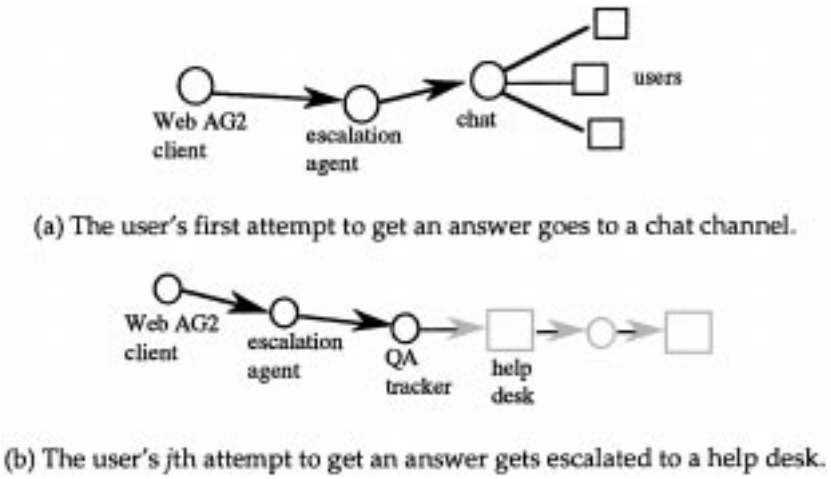

Fig. 8. *Two possible escalations for a question.*

would have in the original Answer Garden, the question goes to his escalation agent. This is, of course, invisible to Fred. The question is first sent to a synchronous chat system (Fig. 8a). We envision the chat system being set up with channels or subchannels for each work group, hallway, or other social grouping. If someone on the chat system can answer Fred's question, and is inclined to do so, Fred gets his answer immediately. As mentioned above, nearby colleagues (as measured by geographical, social, or intellectual distance) are most likely to answer his question with the correct and sufficient context.

In our prototype, after 5 minutes, the system pops up a window on the screen. In our scenario, the dialog box asks Fred whether he got an answer to his question, and if not, whether he would like to continue (Fig. 9). If Fred says to continue, the system routes the question to a NetNews (Usenet) bulletin board. (It is also conceivable that it would route it to a chat channel with a wider distribution, but the point is the same.) After another period of time, perhaps 24 hours, the agent again pops up a window on Fred's screen, asking whether he has received an answer. The process continues, perhaps routing the question to an expertise engine to find a suitable human expert, to a help desk (Fig. 8b), or to agents that search for information on the Web or in proprietary information sources (such as Dialog or Nexus). One can even imagine agents that hire outside consultants if the need is great enough.

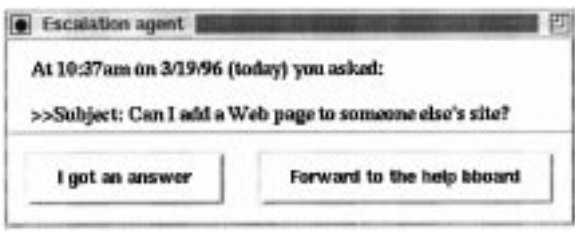

Fig. 9. *The escalation agent.*

In this manner, Fred or any other user is more assured of receiving a usable answer. Staying local lowers the cost, since organizational-level experts need not be used immediately; increases the chance of getting an answer, since colleagues may be more motivated to answer; and is more likely to provide context, since colleagues know the local situation. To be sure, this approach is not a panacea. While it does help provide the proper amount of contextual information to make the answer meaningful, and while there is a greater likelihood that the answer will be at the right explanation level and length, there are difficult issues surrounding the social organization of channel groupings and the like. Colleagues' time is hardly free.

Nonetheless, staying local (but thinking global) does allow group members to help one another, while preserving the capability to ask larger groups as well as experts. Furthermore, the dichotomy between experts and users is largely broken down.

Use of underlying systems

AG2 requires a number of services. These are supported by the underlying CafeCK and Co-Refinery systems. In addition to the basic communication services (chat, NetNews, e-mail) and the escalation agent, AG2 requires services to find experts, to provide basic statistical services, to make users anonymous, and to track users' questions. The capability to find a suitable expert is required, and AG2 currently uses a rudimentary rule-based finding mechanism in CafeCK. This is clearly a bottleneck for real use, but other researchers are developing better mechanisms for handling this problem (e.g., Kautz et al. [13]). The anonymity service allows users to ask questions anonymously. Organizations or communities might not want this service, in which case the service is merely omitted. The statistics service notes which communication mechanisms are used and also tracks the use of pre-existing answers. In a production system, users' questions should be tracked; otherwise, questions can slip away.

The design of CafeCK allows these components to be mixed and matched in a building block manner. Different organizational arrangements can be created through the architecture of the software components. Furthermore, each service can be tailored through its internal Tcl programs.

Services from the Co-Refinery system are also central to the AG2 application. Co-Refinery enables support for building information repositories of commonly requested answers and other information. If this is to be accomplished, low overhead is required for organizational or community members.

The original Answer Garden design assumed that building such a memory repository would occur through the everyday interaction of users asking questions and experts providing answers. However, authoring was still a significant task. The effort of writing explanations and formulating answers cannot be minimized. Nonetheless, Co-Refinery provides the additional mechanisms for refining answers from very raw information sources as well as removing unnecessary context. By using the Co-Refinery collecting, culling, organizing, and distilling facilities, people can find new ways of providing answers. By supporting collaborative activity, AG2 can further reduce the separation between users and experts, allowing for additional organizational learning. In summary, Co-Refinery and CafeCK provide AG2 with new mechanisms for authoring and editing large amounts of information.

Related Systems

CafeCK bears a similarity to message-bus architectures (e.g., Cagan [14]), although these earlier systems lacked the range of services built into CafeCK. They tended to concentrate primarily on the event handling and message routing aspects of CafeCK. CafeCK is also similar to a range of virtual reality systems. For example, Worlds [15] also permits flexible addition of services and message-passing. However, these systems do not emphasis collective memory functionality, graceful escalation, augmenting the social network, or finding expertise. Finally, some functionality within CafeCK is also close at this point to three production systems, Microsoft Exchange, Lotus Notes, and some of the functionality in Java JDK 1.2. CafeCK differs from Exchange in being object-oriented and heavily extensible and from Notes in having a truly distributed architecture. Notes is, in general, much closer to a standard client-server model. The 1.2 release of JDK contains objects for mail handling as well as other objects that could be extended for computer-mediated communication facilities, and future versions of CafeCK may use these JDK objects. None of these systems have both the range and the flexibility of CafeCK.

Co-Refinery's goals overlap with two research streams, collaborative authoring and shared workspaces. Quite a range of systems cover collaborative authoring in the Computer Supported Cooperative Work (CSCW) and the Hypertext communities, but in general, Co-Refinery implements a different form of collaborative authoring than that previously examined. Collaborative Refinery implements an authoring that is more like abstracting or digesting: In addition to the creation of text, Co-Refinery concentrates on the management of sources and collections and the distilling or digesting of those sources. Many of the previous collaborative authoring tools and systems focus on the creation of new text. For example, GROVE [16], Prep [17], Quilt [18], and ShrEdit [19] focus on geographically close, synchronous collaborations to create shared documents. This is also true for asynchronous authoring systems, such as MESSIE [20], Møljner [21], and PrepNet [22]. However, they

share with Collaborative Refinery a focus on distribution, although they used older protocols than http. Co-Refinery does lack SEPIA's [23] emphasis on smooth transitions between loosely coupled asynchronous editing and tightly coupled synchronous editing, and this would be a great asset for the editing of distillates.

The Co-Refinery also presents a shared workspace through the Web. However, as far as we know, Co-Refinery is the only shared workspace system that allows meta-data, content structuring, and the content itself to be modified through the shared space. Moreover, no other shared workspace system has the same range of support for the distilling process. For example, BSCW [24] and Contact [25] maintain form-based meta-data about the status of a collaborative artifact. Both, however, support only a web-based representation of an editing project, but do not specifically support the editing of the writing artifact as well. In Contact and BSCW a co-author would use the shared workspace to get a desired draft of some writing and then use some other system to edit the draft. GAB [26] projects a browsable hierarchy; however, it does not support the manipulation and modification of the shared space. None of these systems have the fluidity of the Co-Refinery authoring process, nor do any have any support for distilling.

Finally, AG2 is clearly related to a number of HCI systems. Help systems in general have been extensively studied in the HCI literature (e.g., Campagnoni and Enrlich [27], Kearsley [28], Aaronson and Carroll [29]). However, while collaborative help is a widely used organizational and social mechanism, it has not been properly considered within the literature. (But see Sproull and Kiesler [30], Finholt [31], and Okamura et al. [32] for notable exceptions to this.) Sproull and Kiesler [30] demonstrate the utility of asynchronous communication mechanisms for providing help within organizations. Our previous work [33] indicated the utility of synchronous chat systems, like Zephyr [34], for help. To our knowledge, no other system has escalation agents or the same range of help services.

There are a number of related CSCW systems that attempt to deal with collaborative information handling. Closest to AG2 are Grassroots [35], Community Memory [36], Spider [37], Designer Assistant (Living Design) [38], Group Memories [39], and BSCW [24]. Grassroots has different mechanisms for collecting, culling, and organizing information. These mechanisms are complimentary to those in AG2. However, Grassroots is lacking distilling features, since sharing the original documents is the major thrust of the work. Community Memory, like AG2, attemptsto build a collaborative information space, but it also lacks explicit support for refining answers or obtaining help. Spider argues for collaborative sense-making and discussion. We have adopted its general philosophy, but its major technical thrust is not on the help problem. Designer Assistant, similarly, provides for iterative information gathering and sense-making, but its major thrust is on design rationale. BSCW provides an alternative shared workspace environment, but as mentioned above, it lacks support for the refining activities in AG2. None of these systems, as far as we know, have collaborative help facilities.

Summary

Returning to a user-centered perspective, at times a user needs access to the right information to solve his questions. The user's problem, however, can be considered as need to augment the collective memory in such a way that it benefits the user as well as all of the social collectivities of which he is a part. The utility in examining help and collective memory together is noting the role of informal information in obtaining a dynamic and useful memory. In this view, a collective memory can not only be reconfigured as need be for the problems-at-hand, it can also successfully store the everyday knowledge of the organization or community.

This paper has examined some technical mechanisms for augmenting the use of informal information in the collective memory accordingly. We have tried to show that *graceful escalation* and *collaborative refining* techniques can be of benefit in using informal information. These techniques can ameliorate, respectively, the context and authoring issues.

We also described two CSCW systems, the Cafe Construction Kit and the Collaborative Refinery, that provide services to support informal information collection, refining, and routing. We tested the utility of these two underlying systems by constructing an application, Answer Garden 2, which used graceful escalation and collaborative refining services.

Acknowledgments

This project has been funded, in part, by grants from NASA (NRA-93-OSSA-09), National Science Foundation (IRI-9702904), Quality Systems Inc, Interval Research Corporation, the University of California MICRO program, the UCI Committee on Research, and the California Department of Transportation. We want to especially acknowledge Eric Mandel for his continuing insights into the needs of user communities, especially the astrophysics community. This project has benefited greatly from conversations with Tom Malone, Paul Resnick, Wanda Orlikowski, JoAnne Yates, Debby Hindus, Jonathan Grudin, Rob Kling, John King, Takeshi Ishizaki, Leysia Palen, Paul Dourish, Kate Ehrlich, Christine Halverson, and many others. The other members of our research group, Jack Muramatsu, Wayne Lutters, Keri Carpenter, Brian Starr, and Andy Tipple, contributed to this understanding of collective memory and collaborative help.

Notes

1. We will use the term *collective memory* to denote the common attributes of organizational, institutional, and community memory. The term has a related, but slightly different meaning in the historiographical and critical literatures, but there is no better term to denote memory in a range of collectivities.
2. In the language of linear programming, two *dual* forms exist for each particular problem. Both forms are valid, and users are free to solve the form that provides them with the most analytical tractability.

References

1. Shipman FM, III, McCall R. Supporting knowledge-base evolution with incremental formalization. In: *ACM Conference on Human Factors in Computing Systems (CHI'94)*. 1994:285–291.
2. Walsh JP, Ungson GR. Organizational memory. *The Academy of Management Review* 1991;16(1):57–91.
3. Foner LN. Yenta: A multi-agent, referral-based matchmaking system. In: *First International Conference on Autonomous Agents (Agents'97)* 1997:301–307.
4. Ackerman MS, Starr B. Social activity indicators for groupware. *IEEE Computer* 1996;29(6):37–44.
5. Bancroft HH. *Literary Industries: A Memoir*. New York: Harper and Brothers, 1891.
6. Ackerman MS, Malone TW. Answer Garden: A tool for growing organizational memory. In: *ACM Conference on Office Information Systems*. Cambridge, MA, 1990:31–39.
7. Ackerman MS. Augmenting the organizational memory: A field study of answer garden. In: *Conference on Computer-Supported Cooperative Work (CSCW'94)* 1994:243–252.
8. Smeaton C. The AnswerWeb. In: *WWW Conference*. http://www.cee.hw.ac.uk:80 /∼ calum/AnswerWeb /paper.html, 1995.
9. Ackerman MS. Definitional and contextual issues in organizational and group memories. *Information Technology and People* 1996;9(1):10–24.
10. Allen T. *Managing the Flow of Technology*. Cambridge, MA: MIT Press, 1977.
11. Kraut RE, Egido C, Galegher J. Patterns of contact and communication in scientific research collaboration. In: Galegher J, Kraut RE, Egido C, eds. *Intellectual Teamwork*. Hillsdale, NJ: Lawrence Erlbaum, 1990:149–172.
12. Blau PM. *The Dynamics of Bureaucracy: A Study of Interpersonal Relations in Two Government Agencies*. Chicago: University of Chicago Press, 1955.
13. Kautz H, Milewski A, Selman B. Agent amplified communication. In: *AAAI Symposium on Information Gathering from Heterogeneous, Distributed Environments* 1995.
14. Cagan MR. The HP SoftBench environment. *Hewlett-Packard Journal* 1990:36–47.
15. Tolone W, Kaplan S, Fitzpatrick G. Specifying dynamic support for collaborative work within WORLDS. In: *Conference on Organizational Computing Systems (COOCS'95)* 1995:55–65.
16. Ellis CA, Gibbs SJ, Rein GL. Groupware: Some issues and experiences. *Communications of the ACM* 1991;34(1).
17. Neuwirth CM, Kaufer DS, Chandhok R, Morris JH. Issues in the design of computer support for co-authoring and commenting. In: *ACM Conference on Computer Supported Cooperative Work (CSCW)*. ACM Press, 1990.
18. Leland MDP, Fish RS, Kraut RE. Collaborative document production using quilt. In: *ACM Conference on Computer Supported Cooperative Work (CSCW'88)*. Portland, OR: ACM Press, 1988:206–215.
19. Olson JS, Olson GM, Mack LA, Wellner P. Concurrent editing: The group's interface. In: *IFIP 3rd International Conference on Human-Computer Interaction (INTERACT'90)* 1990:835–840.
20. Sasse MA, Handley MJ, Chuang SC. Support for collaborative authoring via electronic mail: The MESSIE environment. In: De Michelis G, Simone C, Schmidt K, eds. *European Conference on Computer Supported Cooperative Work (ECSCW)*. Milan, Italy: Kluwer Academic, 1993:249–264.
21. Minor S, Magnusson B. A Model for semi-(a)synchronous collaborative editing. In: De Michelis G, Simone C, Schmidt K, eds. *European Conference on Computer Supported Cooperative Work (ECSCW)*. Milan, Italy: Kluwer Academic, 1993:219–231.
22. Neuwirth CM, Kaufer DS, Chandhok R, Morris JH. Computer support for distributed collaborative writing: Defining parameters of interaction. In: *ACM Conference on Computer Supported Cooperative Work (CSCW)*. Chapel Hill, NC: ACM Press, 1994:145–152.
23. Haake JM, Wilson B. Supporting collaborative writing of hyperdocuments in SEPIA. In: *ACM Conference on Computer*

Supported Cooperative Work. Toronto, Canada: ACM Press, 1992:138–146.

24. Bentley R, Horstmann, Sikkel K, Trevor J. Supporting collaborative information sharing with the World Wide Web: The BSCW shared workspace system. In: *World Wide Web Conference (WWW'94)*. Boston, 1994.
25. Kirby A, Rodden T. Contact: Support for distributed cooperative writing. In: Marmolin H, Sundblad Y, Schmidt K, eds. *European Conference on Computer Supported Cooperative Work (ECSCW)*. Stockholm, Sweden: Kluwer Academic, 1995:101–116.
26. Wittenburg K, Das D, Hill W, Stead L. Group asynchronous browsing on the World Wide Web. In: *World Wide Web Conference (WWW'95)*. Darmstadt, Germany, 1995.
27. Campagnoni FR, Ehrlich K. Information retrieval using a hypertext-based help system. In: *ACM Conference on Information Retrieval (SIGIR '89)*. Cambridge, MA: ACM, 1989:212–220.
28. Kearsley G. *Online Help Systems: Design and Implementation*. Norwood, NJ: Ablex, 1988.
29. Aaronson A, Carroll JM. Intelligent help in a one-shot dialog: A protocol study. In: *ACM Human Factors in Computing Systems and Graphics Interface (CHI + GI) '87*. Toronto, Canada, 1987:163–168.
30. Sproull L, Kiesler S. *Connections: New Ways of Working in the Networked Organization*. Cambridge, MA: MIT Press, 1991.
31. Finholt TA. *Outsiders on the Inside: Sharing Information through a Computer Archive*. Carnegie Mellon University, Ph.D. thesis, 1993.
32. Okamura K, Orlikowski WJ, Fujimoto M, Yates J. Helping CSCW applications succeed: The role of mediators in the context of use. In: *Computer-Supported Cooperative Work (CSCW'94)*. 1994:55–65.
33. Ackerman MS, Palen L. The Zephyr help instance: Promoting ongoing activity in a CSCW system. In: *ACM Conference on Human Factors in Computing Systems (CHI'96)* 1996:268–275.
34. DellaFera CA, Eichin MW, French RS, Jedlinsky DC, Kohl JT, Sommerfeld WE. The Zephyr Notification Service. In: *Winter 1988 Usenix Technical Conference*. Dallas, 1988:213–220.
35. Kamiya K, Roscheisen M, Winograd T. Grassroots: A system providing a uniform framework for communicating, structuring, sharing information, and organizing people. *Computer Networks and ISDN Systems* 1996;28(7–11):1157–1174.
36. Chaplin D. *Community Memory.* Department of Computer Science, Lancaster University, manuscript. 1994.
37. Boland RJ, Jr., Tenkasi RV, Te'eni D. Designing information technology to support distributed cognition. *Organization Science* 1994;5(3):456–475.
38. Terveen L, Selfridge P, Long MD. Living design memory: framework, implementation, lessons learned. *Human-Computer Interaction* 1995;10(1):1–38.
39. Lindstaedt SN. *Group Memories: A Knowledge Medium for Communities of Practice*. Department of Computer Science, University of Colorado, Ph.D. proposal. 1996.