# Where Did You Put It? Issues in the Design and Use of a Group Memory

Lucy M. Berlin, Robin Jeffries, Vicki L. O'Day,
Andreas Paepke, Cathleen Wharton*
Software Technology Laboratory
HPL-93-11
February, 1993

collaborative work, information sharing, information search and retrieval, group memory, group conventions

Collaborating teams of knowledge workers need a common repository in which to share information gathered by individuals or developed by the team. This is difficult to achieve in practice, because individual information access strategies break down with group information — people can generally find things that are on their own messy desks and file systems, but not on other people's.

The design challenge in a *group memory* is thus to enable low-effort information sharing without reducing individuals' finding effectiveness. This paper presents the lessons from our design and initial use of a hypertext-based group memory, *TeamInfo*. We expose the serious cognitive obstacles to a shared information structure, discuss the uses and benefits we have experienced, address the effects of technology limitations, and highlight some unexpected social and work impacts of our group memory.

Internal Accession Date Only
*S.E.E.D. at HP Labs, Summer 1991, University of Colorado, Boulder, Colorado

# Where Did You Put It? Issues in the Design and Use of a Group Memory

**Lucy M. Berlin†, Robin Jeffries†, Vicki L. O'Day†, Andreas Paepcke†, Cathleen Wharton‡**

†*Hewlett-Packard Laboratories*
*1501 Page Mill Rd., Palo Alto, CA 94304*
*E-mail: berlin@hpl.hp.com*

‡*University of Colorado at Boulder*
*Department of Computer Science and Institute of Cognitive Science*
*Boulder, CO 80309-0430.*

## ABSTRACT

Collaborating teams of knowledge workers need a common repository in which to share information gathered by individuals or developed by the team. This is difficult to achieve in practice, because individual information access strategies break down with group information — people can generally find things that are on their own messy desks and file systems, but not on other people's.

The design challenge in a *group memory* is thus to enable low-effort information sharing without reducing individuals' finding effectiveness. This paper presents the lessons from our design and initial use of a hypertext-based group memory, *TeamInfo*. We expose the serious cognitive obstacles to a shared information structure, discuss the uses and benefits we have experienced, address the effects of technology limitations, and highlight some unexpected social and work impacts of our group memory.

**KEYWORDS:** collaborative work, information sharing, information search and retrieval, group memory, group conventions.

## INTRODUCTION

In a previous study of the information needs of technical workers in many domains, we confirmed that computers are used to save masses of ad-hoc, mostly textual information crucial to people's work [11]. In our own field of software research we exchange design ideas and alternatives, schedules and constraints. We track other projects and the names of contacts. We exchange software pointers, bugs, and tricks of use. When we need an information nugget to make progress, that item — the bug work-around, decision, phone number or procedure — becomes critical.

Currently such sharable information is individually archived and managed. People save what they expect to need, using electronic mail folders and files.

However, such individual caches have four disadvantages:

- Each individual has the overhead of deciding what to save, where, and how to manage changes, updates, and weeding.
- As team members' tasks change, they must go to others to find information they deleted when it wasn't relevant, or information they never saw.
- New members of a project do not inherit a store of project-related information.
- When any person leaves, much of their saved expertise is lost to the rest of the team.

For these reasons, organizational redesign experts, software reuse experts, and many knowledge workers have expressed a desire for a shared repository for informal, group-relevant information [12, 2, 4] However, there are two major obstacles to such a shared repository. First, ad-hoc, textual data doesn't fit traditional information models — models designed for published documents or database records. Second, individual information-management strategies do not map well onto group information.

As part of an ongoing project on shared information access, we have designed TeamInfo, a prototype *group memory*. The first four authors have been using TeamInfo for our project's information needs since the summer of 1992. As of Nov 1992 it contains some 1500 items, including an initial set of 1000 items copied from our individual e-mail folders. Our research goal for TeamInfo is to help identify the features and social issues crucial to a useful group memory, to expose conflicting individual strategies for managing information, and to serve as a basis for trying out alternate solutions.

We define a *group memory* broadly as a common repository of on-line, minimally structured information of persistent value to a group. It is a variation of Walsh and Ungson's concept of an organizational memory [15], but tailored to the needs of a collaborating work-group. A group memory is still a broad concept; a group memory must be appropriate to its context — the heterogeneity, stability, computer sophistication, goals, and social environment of its users.

Our TeamInfo prototype is designed as a shared repository

for information useful to multiple people in a small, stable team of knowledge workers. We decided to initially focus on stable teams, because their shared experiences help them anticipate which information might be in the group repository. We chose knowledge workers because their tasks require problem-solving. They are a particularly appropriate group, because their problem-solving requires shared information about workflow, tasks, and hardware and software environments[12].

We view TeamInfo as a shared library of informal information — a repository of items of long-lived value. It is explicitly not another communication channel that bombards users with time-critical information. Users may browse to see what's new, ask to be notified of acquisitions in a given category, or simply search for information relevant to a current task. Our current focus is on the types of information that engineers and computer scientists save on the computer for longer-term reference: meeting notes, design documents, software installation instructions, bug work-arounds, pointers to reports, bits of information about interesting projects and products, and personal recommendations (e.g., restaurants).

TeamInfo combines features of other systems that address aspects of the data overload faced by individuals and teams. TeamInfo provides mixed manual and automatic classification of mail-based information, hypertext links, notification, and a browser- and query-based interface with simple full text retrieval.

Systems with goals related to TeamInfo include electronic mail filters such as the Information Lens [10] and Strudel [13]. These help users organize their e-mail and facilitate the structuring of discussions. However, they do not support the sharing of such information after it has been saved. Issue management systems such as gIBIS [5] also help teams track design alternatives, supporting positions and objections, but they require a very structured representation of discussions. And, systems such as Answer Garden [1] provide a decision-tree-based structure for recording software questions and answers, but do not span as broad a range of information types as TeamInfo, nor do they provide automatic classification or full text retrieval.

Our initial prototype does not try to cover all technical issues or information types. We omit security levels beyond Unix security, which is reasonable given our focus on single teams. The TeamInfo prototype handles only textual data; we believe that gives enough value while avoiding the complex issues of indexing voice, bitmaps, schematics, or video.

We provide full-text regular-expression searching, but our research focus is not on text retrieval. We agree with Evans that real-world problems of complex text management require representations that can recognize concepts from the relations among words, "but the facilities required to solve [such] problems are very complex. In general, they are not portable, not extensible, and not easy to implement or maintain." [7]. So, even though TeamInfo's platform is extensible to multiple text retrieval platforms, we have chosen not to immediately add lexicons, thesauri, or latent semantic indexing. Instead, we are trying to avoid the complex task of automatic concept recognition by TeamInfo's mixed manual and automatic classification.

After introducing TeamInfo, this paper describes our project's experiences with developing a shared classification, the effects of TeamInfo's design and technology choices, and the emerging social and work impacts of a group memory.

## TEAMINFO: A GROUP MEMORY PROTOTYPE

TeamInfo is group memory application based on the prototype Group Memory Manager, GMM, developed by Cathleen Wharton during her summer internship at HP Laboratories. [16]. Both versions are built on top of a general-purpose hypertext system, Kiosk, also developed at HP Laboratoriess [6]. Kiosk provides a flexible nodes and links model implemented using Unix text files, an InterViews-based interface [9], and inter-process communication and concurrency control using the BART *software bus* mechanism [3]. Kiosk also provides the Cost$^{++}$ automatic linking tool, which enables us to specify declarative rules to extract patterns from files and to link messages to the appropriate classification nodes.

We chose an electronic mail-based input model to TeamInfo for four reasons. First, e-mail submission minimizes the hurdle of submitting information by enabling users to stay in the context of an e-mail reader or editor. Second, since much of our information comes via e-mail, users can easily forward the useful nuggets of information they receive — items such as software installation notes, product ordering information, bug work-arounds, and technical report abstracts. Third, the team's e-mail discussions can be trivially archived in TeamInfo by adding the repository to the messages' *cc* list. Fourth, any other on-line document types can be easily converted into e-mail messages.

TeamInfo first parses each mail message and extracts data from structured fields such as sender, date, subject, and message-id. If a message is part of a conversation thread, i.e. a set of messages and replies on one topic, TeamInfo locates the message's predecessor and links the two messages.

Next, TeamInfo classifies the message, linking it to one or more group-defined categories. Messages are classified using a combination of automatically extracted and sender-provided information. Senders assign each item to one or more of a small number of categories, such as *Literature*, *Events*, or *Miscellaneous*. Senders may also specify any number of keywords or phrases that are potentially useful to finding the message, using a free vocabulary. Cost$^{++}$ then scans the message and pattern-matches to the expressions, terms, and phrases that we have specified as indicative of a category.

In order to alert users of potentially interesting additions without inundating them with notifications, we began sending periodic summaries of new contributions via e-mail. This is similar to a library's sending a monthly list of new acquisitions. If the sender wants a team member to see the item quickly, one uses the usual mail mechanism and adds that person to the mail's

*cc:* list.

Thus, in our model, users go to TeamInfo when they have a question to answer, or when they see an interesting contribution title. TeamInfo supports both query-based and browser-based search. The query-based access uses regular expression searching over the messages' contents. Since one can sometimes more effectively recognize appropriate items than specify a good query to retrieve them, TeamInfo also supports browsing and navigation through the classification hierarchy.

## FORCED COGNITIVE COHABITATION: Reconciling Long-Lived Filing Habits

An inherent problem of a shared repository is that individual finding strategies do not work for a group. Individual filing systems have four advantages. First, there is no negotiation overhead: users do not have to discuss explicitly their classification conventions. Second, consistency is less important: users remember their own classification rules, the changes, and the exceptions. Thus, they can guess a small set of classes where they might have put the item. Third, users are less dependent on a classification. Since they had read all the items they'd saved, they are likely to remember a unique word or phrase, and be able to use use text search. Fourth, if the initial searching attempts fail they will try different strategies, since they *know* the item is there, somewhere.

A group memory does not have these benefits. Users won't know another user's idiosyncratic rules and exceptions, won't be sure an item is there, and won't necessarily be able to search for a remembered unusual word or phrase. Thus, the first challenge in a common repository is for the group to develop some classification conventions that allow individuals freedom of expression, while maintaining some constraints that will ensure a high rate of findability.

Since our goal was to have TeamInfo span all of a team's shared information needs, we didn't think that unstructured text items would include enough of the descriptors needed for findability *by someone other than the person who had submitted an item.* We adopted a two-pronged approach to facilitate findability. First, as mentioned above, we added a *keywords* field to the e-mail messages, allowing authors to specify likely search terms. Second, we hypothesized that browsing would become more important in a group repository — that searchers would want to cast a net broadly and then try to recognize the right information, rather than trying to guess at the right retrieval terms. Thus, we designed a group classification to serve as an initial clustering. The classes serve two purposes: they enable browsing and they serve as natural contexts for queries.

We decided to design a simple classification: one with no more than ten top-level classes and a maximum of two levels of hierarchy. The goal of simplicity was driven by the desire to minimize the overhead for submitting an item to TeamInfo, and the knowledge that complex controlled vocabularies require extensive training for effective indexing and retrieval. On the other hand, given the very small (<0.20) agreement in users' spontaneous word choices for objects [8], we felt that a small controlled vocabulary was necessary to give an initial clustering.

We expected to sit down, agree on a single, simple classification, and be done. Given our similar project goals, computing environment, and research interests, our only concern was that we were too homogeneous to have interesting differences in personal styles. We were wrong. Very wrong. Fortunately we had approached this systematically – after an initial discussion we listed our personal votes for the top ten (or fewer) categories, then sat down to create a group list. We audiotaped our discussions, so as to capture the reasons for individual positions and the habits that underlie different choices.

Over a few meetings we agreed on a set of core candidates we could live with. These included our two major project foci: *DIME* and *Group Memory*; a set of secondary activities (such as our study group, *Datalunch*) which became categories under *Sidelines*; and *Project-Miscellaneous.* Beyond that, we had categories such as *Events*, *Technology Hacks* (for messages about hardware and software tools, tricks etc), *People+Projects*, *Topic Tracking* [with subcategories], *Archive* [with subcategories] and finally a global *Miscellaneous.*

### It's not enough to agree on a set of categories

Beneath our surface agreement on the categories lay crucial differences, exposed when we compared how we would classify a set of test messages based on project members' activities and recent e-mail. We differed along the following five dimensions:

1. purists and proliferators
2. semanticists and syntacticists
3. scruffies and neatniks
4. savers and deleters
5. the expected *purpose* for which the item is saved

Here are descriptions and examples of the differences in each dimension:

*Purists and proliferators* Many items might fit into multiple categories. The purists wanted to put things into one place as an author, and were willing to look into multiple places as a searcher. As one said, "I prefer this to proliferating [virtual] copies all over the place. As a reader, I want the buckets uncluttered and thematically clear." For the proliferators, no one choice was correct — for example, an article about project "Papyrus" clearly belonged in both *Literature* and *People+Projects*; to them it would be frustrating to not be able to find all Papyrus-related things within *People+Projects*; to have to also search *Events*, *Literature*, and perhaps even *Project-Misc.*

*Semanticists and syntactists* For the syntactists, structural and episodic clues are important for retrieval, and unlike seman-

tic classification, they are unambiguous. Thus, the syntactists wanted articles and issues discussed at Datalunch, our weekly study group, to be visible under *Datalunch*. On the other hand, he semanticists wanted to find the Datalunch discussion of Nardi's latest groupware study under *CSCW*, and the next week's discussion of ethics in video research under *Miscellaneous*. Those who use episodic cues were quite discomfited at the thought of playing detective to find the semantic category that *someone else* decided a Datalunch item belongs into. The proliferators had an easier time choosing — they would put each item into both. However they too wound up being affected by others' styles: they expected to be able to find the items under both the semantic and syntactic categories.

*Scruffies and neatniks*

We drastically differed in our optimal classification granularity – the scruffies had originally wanted TeamInfo to have just five top-level categories while the neatniks are used to living with up to three hundred fine-grained (and hierarchical) e-mail folders. The neatniks deplored the loss of clustering in TeamInfo caused by stuffing items on a dozen software packages onto *Hacks*, and our information on all relevant projects under *People+Projects*. To each person, the effort required to adopt other's style is burdensome at exactly the wrong time: the scruffies want to minimize the up-front cognitive load of filing, and are willing to spend more effort at retrieval time; the neatniks, on the other hand, want items to be pre-organized, and browsable. They prefer to spend up-front time to reduce the retrieval cost.

When we settled on a relatively small number of categories, we really weren't settled: the neatniks asked for documents to at least be given keywords that might permit future subclassification, or personal views at smaller granularity. Only extended use may show whether the scruffies will continue to generate these keywords, and whether such informal indexing meets the needs of the neatniks.

*Savers and deleters*

Our individual styles also affected what we each thought belongs in TeamInfo. The savers would like TeamInfo to include individual wisdom of potential group utility, such as restaurant recommendations, or the steps and pitfalls to editing a book. The deleters wouldn't have dreamt of cluttering TeamInfo with items that don't represent project-related group activities, or with e-mail design discussions of unknown future value.

*Purpose-based filing*

Even after we agreed on the set of TeamInfo categories, the category that seemed appropriate depended on our roles and expected future tasks. For example, here the engineers and manager disagreed on how to file a report we had written, evaluating another HP project's CD-ROM interface:

**researcher1:** I think that's *Technology Tracking*
**manager:** No, this is ...definite[ly] ...I would put in *People+Projects*.
**researcher1:** I don't care about *the people* — I mean this is a technology that I am tracking, right?
**manager:** I save those things because at some point as a manager I am going to need to know this for something. That is, I am going to need to know: "who is the manager of it?" ..."what's going to be different in it?" ...That is exactly what *People+Projects* are [for] and it will be hard to get me to change to put that under *Technology Tracking* although I can see exactly what your argument is for it.
**researcher2:** And I would put it under, in my old categorization, under *Project-Miscellaneous* and under *People+Projects* because it is [both] something that we did as a minor activity and it has to do with an ongoing project that I might be tracking. ...[much of] my technology tracking is likely to be under *People+Projects* because there's a lot of technologies that I track based on who is doing [what].

Over the discussion sessions we developed models of one another's idiosyncratic styles, so that we could compensate somewhat when searching (e.g. "Ah, she might have put it under *People+Projects* or *Events*"), but our theories weren't reliable, and remain very foreign to our natural habits. Given that our predictions are unreliable even within this ideal case — a stable group — we expect that individuals in more dynamic groups would have many more problems. New group members would have to learn not only the group's categories and filing rules, but also the fact that only some people file things under multiple categories, and that some hardly ever put items under *People+Projects*.

### Personal vs. Group Information Habits

The differences described above are based on long-lived filing habits and our varied goals for the information. Thus, even after we had agreed on common categories in TeamInfo, we were faced with ongoing dissonance between the group memory classification and our personal e-mail folders and our natural tendencies. This dissonance was immediately visible in meetings when we re-visited TeamInfo policy issues (such as whether items could be in *Miscellaneous* plus another category). It was hard to remember what we'd agreed to, and what each person remembered tended to drift toward the person's initial position.

## DESIGN AND TECHNOLOGY CHOICES

To learn from any prototype one must ask how well the task model is supported by the fundamental data model choices, and where the prototype's usefulness is affected by technological obstacles. Here we give our observations to date of the utility of two data model decisions in TeamInfo: (1) the hypertext model and (2) the model of mail-based input. We also discuss our experiences with and technical requirements for notification and for the searching and browsing interface. These issues are also applicable to the design of other shared information systems such as electronic bulletin boards and asynchronous conferencing systems.

*Is Hypertext a good model for a group memory?*

We find that hypertext links add value at the implementation level, but at the user's level a group memory requires a layer above hypertext. The user model of TeamInfo is that of documents put into one or more thematic categories, not of nodes linked to classification nodes. Users should not care that, at an implementation level, documents are linked to classification or query nodes – it is more familiar to think of the classes as an indexing hierarchy.

Hypertext links provide an understandable conceptual model for threaded messages in an e-mail conversation, but the user-level link-following operations are little used. When reading an e-mail conversation we find it choppy to bring up a separate window for each conversational turn. To handle that, we've thought of presenting the messages as speech acts in a conversation, and having the system build a transcript to be read as a consistent whole, perhaps with an index showing the author, date, subject and keywords for each message. Such a linear transcript would reduce the overhead of reading, make it easier to maintain context, and could simplify skimming or searching within a vaguely-remembered conversation.

The hypertext data model provides a clean way to link information. We link messages and their replies, queries and their results, classification nodes and the items within them, and keywords and the items which refer to them. The Kiosk platform provides a powerful filtering mechanism that is link-aware – we can search for, say, all messages by person X by searching the "author" link. Again, that is useful functionality, but it can be provided by pre-computed indexes or by queries over database records (such as provided in Tapestry [14]); it does not require the "author" attribute to be represented by a link.

*Mail-based information*

We are quite happy with the choice of mail-based input to team memory. It is easy to *cc:* TeamInfo on e-mail discussions and to forward information we get via e-mail. Also, many useful attributes of an item (date, sender's name, ID, reply-to ID, original sender...) are automatically generated. Thus, the author needs only to fill out the class and keywords fields in the augmented mail header. Since people sometimes forget to fill in those fields, TeamInfo provides graceful failure by also running the automatic classification rules on all new items. A more sophisticated solution would be to run the classification at send-time, and ask the sender to verify whether the identified categories are correct. This would reduce the cognitive load to the sender and avoid missing categories, but it would require extensions to each different mail interface used by the team members.

However, the mail model does have an impact on our interaction style. Items within TeamInfo are editable, so there is no *technical* reason keeping us from changing a submitted item (e.g. a bug report, or a set of instructions). However, that's strongly not in the model of e-mail. For that to feel comfortable within the e-mail model would require a mechanism to mark and timestamp the revisions and authors, and to allow us to notify others that an item has been modified.

Similarly, the e-mail model doesn't support shared dynamic documents. Dynamic documents — such as a software release to-do list, a list of open questions, a bibliography file, or a file of useful commands for infrequent activities — do not just get appended to; information is added in the middle, updated, and sometimes deleted. We generally haven't put such potentially sharable documents into TeamInfo, even though Kiosk does provide concurrency control. We are beginning to analyze why that is. It may be because we're still switching our work styles to make use of TeamInfo, but we are also bothered by fundamental obstacles such as TeamInfo's lack of a browser of "commonly accessed documents"; the lack of change bars and time/namestamps for changes; and each person's desire to use a favorite editor to search and edit complex documents.

*Will the real sender please stand up?*

As we said, messages sent to individuals are often forwarded to TeamInfo. In the initial design we became erroneously listed as the item's author. A similar problem occurred whenever we forwarded a team-member's messages to TeamInfo if we noticed that the original author forgot to *cc* TeamInfo. Fortunately with the mail-based input we were able to fix this, since a mail message preserves info on both senders, as well as both dates, message-ID's and subject lines. Using these, TeamInfo is now able to sort messages by their original send dates, and figure out *reply-to* links appropriately, even if the original message in a discussion thread is submitted after some of the replies.

The multiple IDs, subject lines, dates, and authors in forwarded messages pose a challenge when designing the browser-based interface. Which name, date, and subject line should appear in the message header? Which is the real message-id for the item? We chose to present the most recent (and usually most informative) subject line, but the original date and author. Using the original date and author helps compensate for us forgetting to *cc* TeamInfo. It also allows anyone to resend an item (e.g. about a new software release) without appearing to become the message's author. On the other hand, we miss the low-cost *mutual awareness* we could get by knowing who forwarded a message to TeamInfo; the sender's name provides an indirect window onto team-members' expertise and interests.

*Notification*

In TeamInfo, new items may be added without other people being notified of the additions. This makes sense for items that are of only future relevance to others: for example, when one of us began using a laptop computer, others did not want to see the nuggets of knowledge, strategies and work-arounds he was acquiring until we began to use laptops ourselves. Thus, individuals can avoid being bombarded by items with only potential future relevance. On the other hand, there are advantages to notification. If one has even glanced at an item that's in such a repository, one may recall the title, a keyword, or an un-

usual phrase, and will search more creatively and persistently than if one doesn't know it's there.

We have experimented with both styles, and we believe that notification does aid findability, but it has to have low cognitive overhead. We believe that the notification messages shouldn't interrupt a train of thought or demand immediate action. The messages should be informative, directly manipulable, and come at recipient-controlled times. Notification messages need to show the item's summary line, and let recipients individually mark, acknowledge or directly view the items. We are experimenting with periodic summaries of messages recently added to TeamInfo, and are designing customizable notification based on individuals' registration of their interests.

Many external messages are sent to the whole team. In our project, the social convention is that anyone may forward these to TeamInfo. To avoid redundant copies, we've sometimes used the notifications to check whether someone else has already submitted a particular item to TeamInfo. However, it would make more sense to see this *within the e-mail reading context*; just like there is a 'deleted' flag on each message header, one could imagine an 'archived' flag. This unfortunately would require a centralized model of e-mail, one in which all recipients view the same copy of a mail message. With a central e-mail server, one person's archiving action could trigger the 'archived' flag for other readers in a team. Of course, with distributed responsibility, the more common problem may be that nobody will submit a message to TeamInfo. That is a question of social conventions — notification or central e-mail with an 'archived' flag makes distributed responsibility possible, but will not alone make it work.

*Searching and Browsing*

The initial prototype provides full-text searching with optional case sensitivity, but without structured search of e-mail fields. Since the documents are e-mail messages, we really wanted to be able to express restrictions based on fields such as date range, author, or keywords. We also wanted to be able to express structural constraints such as "search these two categories, and their subcategories" or "search all messages in this thread". Both of these features are planned but not yet available.

We do use the query-based interface, but primarily for brute-force search when we know that an item is there somewhere, and when we remember some unique phrase that we can search for. However, the limitations on its expressiveness and its slower speed inhibit its use, even among those who expected to always use queries.

We were able to get by with the more limited search capabilities once we revised the browser view's message summary lines to show date, author, subject, and also keywords. After that, the query-results browser generally gave enough information on each message to let us quickly find the target items. However, this wasn't true for design discussions, in which a half dozen or more messages had the same title — again suggesting the need for a more task-oriented representation of conversation threads.

We had strong differences in how much we expected to use the browser interface – some of us expected to always use queries and never to use the browser based interface, while others felt they could often recognize a hit more effectively than they could describe it in a query. The final answers are not yet in; our use patterns still vary as a result of changes in the functionality, presentation information, and the performance of the query and the browsing interfaces. However, browsing seems especially useful in three situations: (1) when we have an imprecise notion of *when* the item was created, so that a time-sorted view helps in searching, (2) when we expect to recognize the item but don't quite know how to describe it, and (3) when we're not sure whether the right information is in TeamInfo but figure we know where it is likely to be. Browsing gives a better feel for what's in a category, the time sequence of its items, and the keywords that have been used to describe items.

Browsing also leads to serendipitous finds. We had seeded TeamInfo with around 1000 items from our personal folders. Since these items were often forgotten, our browsing led to unexpected "discoveries" ranging from forgotten article pointers, to text formatting tricks, to rollerblading recommendations. The browser's quick overview of its contents enables *grazing* — semi-random wandering through the space, stopping at interesting-looking messages.

Our use of the browser and the query interfaces confirm our desire for an integrated query and browser mechanism supporting incremental refinement. The mechanism should include queries over previously retrieved sets, relevance feedback on matching items, and a browser of query results — sorted by category, and with secondary sorts by time, number of hits, author, etc.

## WORK AND SOCIAL IMPACT OF TEAMINFO

Use of TeamInfo is slowly becoming a habit, both for information saving and for retrieval. Its use requires one to remember that an item is worth archiving for the group or that an item is likely to be in TeamInfo. As with most repositories, the more we find TeamInfo useful, the more we remember to use it and the more willing we are to add information.

TeamInfo is changing our information management style. We are increasingly submitting items to TeamInfo without *cc*'ing team members, items such LaTeXtricks, recommendations, and sometimes older information that ought to be in TeamInfo. We used to toss away many such items once they were no longer needed. Instead, we are starting to ask each other "do you want to see such in TeamInfo," and are often getting the answer "yes." Thus more information of potential relevance is archived, rather than being discarded or disseminated to others who have no immediate need for it.

Even though we are in the early phases of evaluating the system's utility for specific tasks, many areas of social impact are

already apparent: we are facing issues of trust in TeamInfo's longevity; we've evolved a *curator* role; we are beginning to see effects on our e-mail, information saving and organizing habits; we are seeing issues of privacy and ethical use; and are dealing with issues of reward and social pressure.

*Dependence on the group memory*

Saving items in TeamInfo rather than one's personal files requires trust in the system's longevity. This is an issue for us, as it will be for any users of a group memory. Individuals are used to taking their personal e-mail folders with them as they switch projects, sometimes even companies. With TeamInfo, each item now raises the question of whether to save it in one's individual mail folders or to save it in TeamInfo. The latter decision requires trust that TeamInfo will last for the useful lifetime of the information, or that the information will be easily exportable if the individual leaves the group. Clearly, much of the information does not only belong to the group; it is often useful to and "belongs" to individuals even independently of their role in the group. Thus, to be trustworthy in a dynamic organization, a group memory must provide export facilities that let individuals retrieve copies of relevant items which they and their team-members had entrusted to the repository.

*The curator role*

The Cost$^{++}$ automatic classification tool uses a set of declarative rules to match text expressions to classes. This is extremely useful for classifying a corpus of messages, but the rules require significant experience to use well. One has to become aware that terms like "report" are verbs as well as nouns, that *CHI* should be specified case-sensitive, but *LaTeX* shouldn't. Since classifications are cached in the items, incorrect automatic classifications must be repaired by a custom script. The amount of detailed knowledge required for these functions encourages specialization, and thus we have made one person be the system's *curator*.

In TeamInfo's case, the curator does not make sure the right things are archived; that is still a distributed responsibility. However, she or he is alert to and fixes classification problems, and brings up classification issues for group discussion.

*Changes in e-mail use*

TeamInfo subtly but definitely changes our use of e-mail. It requires some cognitive overhead to decide whether a message to a team-member or a reply is worth archiving, and if so, to give it meaningful keywords. It requires more effort to slip into the group's mindset and classify it using the group's conventions. In addition, we try to catch ourselves sending mail without cc'ing TeamInfo; and we have trained ourselves to check if team-members' messages went to TeamInfo. That too requires vigilance.

The permanence and sharability of items in the repository raises issues of trust and etiquette. People usually feel fine about saving sensitive messages in their own e-mail folders, but not necessarily in a group repository. Thus, even if thematically a message sent to an individual belongs with other items already in the memory, it may not be put there. Thus, there is a tension between the group-memory culture of sharing relevant information, and the usual e-mail etiquette that treats as private e-mail sent to an individual.

The persistence of items in the repository has other effects as well. Some of us feel a subtle pull to be more cogent, more serious, and more accurate in choosing keywords. In regular e-mail one often writes subject lines whose meaning is clear in the current context, such as "the list you asked for", or whose meaning is clear once the message is read. One sometimes writes amusing or cute titles, where the pleasure to the recipient is worth the loss of subject line information. Thus, a major software release may have the title "try it, you'll like it." – or a bug report the subject line "arghh". Such levity may work less well when the messages are archival and the titles become a major retrieval cue. This would be a pity; the informality and levity help maintain the personal relationships in a team, and more informative subject lines do take more effort to compose.

*Rewards*

Since we are trying to figure out the necessary features for a useful group memory, we are currently under a moral contract to use the system. Thus, it is hard to evaluate whether TeamInfo's utility outweighs the overhead of contributing, or how individuals will decide whether to forward information sent to the whole group, and when they will rely on someone else to do that. The summary of submitted messages may exert a subtle pressure, by making it visible who is contributing and who isn't. [1] However, longer use and experiments with other groups will be needed to explore reward issues, to see whether the utility outweighs the overhead of putting material in, and what factors the utility depends on.

*Evolution of categories and project direction*

In individual information files, individuals choose when to begin new categories, or whether to modify their organization to reflect name changes. As individuals, we are often comfortable with inconsistencies. For example, if we have conference-based e-mail folders, we might continue to file the *INTERCHI* items under *CHI*, and not bother with an ephemeral category. However, such a simplification will only make sense to those team members who are already aware of the relationship between *INTERCHI* and *CHI*. Thus, in a group repository, people feel the need to maintain group consistency, despite the additional overhead.

Division of categories also triggers group discussion. In our individual files, we create new categories when an old one gets too big, or when a subsidiary activity takes on a life of its own. Generally, we don't take the effort to extract the relevant

[1] This is reduced by the fact that forwarded messages' summary lines show the original author, not the TeamInfo contributor.

messages from the old combined category — we just remember the divergence time and search accordingly. However, in a group repository this is not acceptable. The old items (that belong to the new category but are hidden within the old general category) would be inaccessible to those team members who are not familiar with the repository's history. Thus, evolution of categories requires group agreement on the new structure, and then it requires work by the curator to retroactively reclassify items.

Our discussions about new categories sometimes force us to re-examine the group's goals. Because we have to agree what belongs in TeamInfo and on how to classify items, we wind up discussing whether an activity is part of a project major focus or a sideline, and how sidelines such as task forces may affect our future direction. It's not yet clear whether such discussions are premature or if they are beneficial (by making us grapple with project goals and tradeoffs earlier). In either case, the introspection and discussions are an impact we hadn't foreseen.

## SUMMARY AND CONCLUSIONS

This paper explores the feasibility and implications of a *group memory*, a shared repository of minimally structured information of long-lived value to a group. Our design experiences and pilot use of TeamInfo illuminate some of the social and technological issues inherent in this groupware tool.

Even though TeamInfo is an early prototype, it has become useful to our project. Our initial observations are confirming the utility of design choices such as e-mail based input, mixed manual and automatic classification, periodic notification, and the need for integrated browser-based and query-based capabilities for information finding. Our problems with the TeamInfo prototype also confirm other (familiar) requirements for effective information retrieval: fast query and browser response, a good query language with boolean and proximity operators, and an interface that facilitates iterative refinement of the queries.

As a system used for real tasks, the TeamInfo prototype has been invaluable in exploring the cognitive aspects of information sharing. It has helped expose the conflicts between saving information in a personal vs. a group repository, the social issues of rewards and responsibility for the care and feeding of the group memory, and the subtle changes in our communication styles and our use of e-mail.

This paper has shown that information management strategies that are appropriate for individuals break down in a shared information repository. We have described how individuals' long-lived filing habits differ along a number of fundamental dimensions, complicating the design of a shared information space. Our experience shows that even after agreeing on categories for TeamInfo, our filing styles made us likely to classify the same items differently. Individual differences in the favored retrieval clues, the tendency toward filing under single or multiple categories, the expected use of information, and the willingness to expend effort at filing time versus retrieval time, affect (1) *which* classes seem appropriate for an item, (2) *how many* categories we put an item into, and (3) whether we each *choose to spend the up-front energy* to add keywords to further describe an item.

With extended use of TeamInfo, we may identify potential solutions to the forced cognitive cohabitation in a group information system. Extended use is also needed to expose long-lived social barriers to shared information. Other important issues include the scalability of the classification and information retrieval mechanisms, and the social issues related to ownership, deletion, and changing group membership.

As researchers in shared information, we believe that group memory systems have the potential of being useful. However, this will not be an easy task. As this paper has shown, many cognitive and social issues must be addressed in order to develop a successful design. We hope that our observations of use, our technological suggestions, and the research hypotheses raised by our reflective use of the TeamInfo prototype will serve as a guide to practitioners and applied researchers in CSCW and group information retrieval.

## ACKNOWLEDGEMENTS

Mike Creech, Dennis Freeze and Mark Gisi made major extensions to Kiosk to address the needs of TeamInfo's task domain. Mike Creech, Mark Gisi, Bob Leichner, Bonnie Nardi, and Glenn Trewitt also gave helpful comments on earlier drafts.

## References

[1] Mark S. Ackerman and Thomas W. Malone. Answer Garden: A tool for growing organizational memory. In *Proceedings of the Conference on Office Information Systems*, 1990.

[2] Victor R. Basili, Gianluigi Caldiera, and Giovannni Cantone. A reference architecture for the component factory. *ACM Transactions on Software Engineering and Methodology*, 1(1), Jan 1992.

[3] Brian Beach. Connecting software components with declarative glue. In *15th Annual International Conference on Software Enginnering*, 1992.

[4] John Seely Brown. Research that reinvents the corporation. *Harvard Business Review*, 69(1), 1991.

[5] Jeff Conklin and Michael L. Begeman. gIBIS: A hypertext tool for team design deliberation. In *Hypertext '87 Proceedings*, November 1987.

[6] Michael L. Creech, Dennis F. Freeze, and Martin L. Griss. Using hypertext in selecting reusable software components. In *Hypertext '91 Proceedings*. ACM, 1991.

[7] David Evans. First order solutions to second-order problems in information manageement. In *Proceedings of the Bellcore Worshop on High-Performance Information Filtering*, 1991.

[8] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11), 1987.

[9] Mark A. Linton, John M. Vlissides, and Paul R. Calder. Composing user interfaces with InterViews. *IEEE Computer*, 22(2), 1989.

[10] Thomas W. Malone, Kenneth R. Grant, Franklin A. Turbak, Stephen A. Borbst, and Michael D. Cohen. Intelligent information-sharing systems. *Communications of the ACM*, 30, May 1987.

[11] Vicki L. O'Day and Andreas Paepcke. Understanding information needs in technical work settings. Technical Report HPL-92-123, Hewlett-Packard Laboratories, 1992.

[12] Calvin H. P. Pava. *Managing New Office Technology*. The Free Press, 1983.

[13] Allan Shepherd, Niels Mayer, and Allan Kuchinsky. Strudel: An extensible electronic conversation toolkit. In *Proceedings of the Conference on Computer Supported Cooperative Work*, 1990.

[14] Douglas Terry, David Goldberg, David Nichols, and Brian Oki. Continuous queries over append-only databases. In *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data*. ACM Press, 1992.

[15] James P. Walsh and Geraldo Rivera Ungson. Organizational memory. *Academy of Management Review*, 16(1), 1991.

[16] Cathleen Wharton and Robin Jeffries. Understanding the role of structure in information filtering in the context of group memories: Some application and user requirements. In *Proceedings of the Bellcore Workshop on High-Performance Information Filtering*,. Bellcore, 1991.