HUMAN-COMPUTER INTERACTION, 2005, Volume 20, pp. 139–162

E-Mail Research: Targeting the Enterprise

Martin Wattenberg, Steven L. Rohall, Daniel Gruen, and **Bernard Kerr**
IBM Research

ABSTRACT

The research program at IBM's® Collaborative User Experience (CUE) group supports an e-mail system used by millions of people. We present three lessons learned from working with real-world enterprise e-mail solutions. First, a pragmatic, system-level approach reveals that e-mail programs are generally used idiosyncratically, often for many different goals at once. This fact has strong implications for both the design and assessment of new features. Second, we discuss how viewing e-mail as an element of corporate collaboration—not just communication—provides insights into problems with current systems as well as potential solutions. Third, we describe constraints imposed by the realities of software development and how they shape the space of feasible new designs. Finally, we illustrate these lessons with an overview of CUE research strategies in the context of

Martin Wattenberg is a computer scientist with an interest in information visualization and collaboration; he is a researcher in the Collaborative User Experience research group of IBM. **Steve Rohall** is a computer scientist with an interest in synchronous collaboration, information visualization, and network communications; he is a software architect in the Collaborative User Experience research group of IBM. **Dan Gruen** is a cognitive scientist with an interest in interface design, collaboration, and human activity management; he is a researcher in the Collaborative User Experience research group of IBM. **Bernard Kerr** is a designer with an interest in interactive systems, visualization, and collaboration; he is a design researcher in the Collaborative User Experience research group of IBM.

CONTENTS

an extended case study of one specific new technology: Thread Arcs. Although not all researchers work with an enterprise-level product team, we believe the experiences described here will be useful to anyone wishing to see their ideas ultimately implemented on a broad scale.

1. INTRODUCTION

How do you improve an e-mail system used by 90 million people? This article describes the e-mail research program of the Collaborative User Experience group (CUE) in IBM® Research, which aims to do just that, targeting the widely deployed Lotus Notes® platform. The mission of improving an existing broadly-used system presents unique constraints as well as opportunities. (Enterprise software, like politics, is the art of the possible.) In this article, we discuss some insights that sprang from the perspective of working with a product group that sells technology to entire corporations rather than individual consumers, as well as lessons learned from the process of working with a large-scale system. Our hope is that anyone wishing to see new e-mail features broadly implemented will find our experiences of use.

Because it is targeting a huge, diverse base of users, the research at CUE has generally taken the view that e-mail must be understood as a complex, multifaceted system. Beginning with the work of Whittaker and Sidner (1996), our group investigated the ways in which e-mail is "overloaded"—that is, used for many functions besides simple person-to-person communication. We also explored the high degree of idiosyncrasy in e-mail usage: there are huge individual differences in how electronic messages are handled. A consequence of overload and individual differences is that any approach to improving e-mail must be empirical and system-oriented, because any change to the system may have unexpected side effects.

A second aspect of CUE's research is that it takes place in the context of Lotus Notes, a long-lived platform for very general types of collaboration. This context has led us to emphasize how e-mail functions as a tool for collaboration as well as communication. For example, we have studied extensively how different users share the same inbox. These studies have revealed both important unmet needs for concurrent usage in existing e-mail programs, as well as suggesting new features and constraints that apply even to single-user e-mail.

Finally, we talk about some of the technical issues that constrain enterprise software. In particular, we discuss how issues of client-side processing versus server-side processing can lead to a dramatic mismatch between promising lightweight prototypes and the reality of implementation in a large-scale environment. We also talk about how some issues that are often seen as mere "implementation details," such as internationalization, have had an impact on acceptance of new features.

Although these constraints are important, they are not paralyzing. We close the article with an extended case study of a transfer of a new technology from our lab to the developers who implement shipping software. This case study gives examples of our own design process, false starts, and tactics used both to improve new technology and to evangelize it within the company. We feel the lessons and insights we have gained in the process can help any e-mail researcher who hopes to see his or her results broadly applied.

2. OVERLOAD AND IDIOSYNCRATIC USAGE

E-mail has become ubiquitous; everyone investigating e-mail has extensive experience with it. As a result, it is easy and not uncommon for designers and developers to make assumptions about e-mail usage based purely on personal experiences. At CUE, we have believed strongly that intuitions about e-mail must be backed by empirical findings. This belief stems not just from a concern for social science methodology but also from an attitude related to research aimed at an extremely large corporate audience: The products we design will not likely be used by people exactly like us. (This stands in contrast, for example, to the open source "scratch your own itch" philosophy of design [Raymond, 1998].) As it turns out, our emphasis on studying real-world usage in diverse settings has led to some critical insights.

An early CUE empirical study (Whittaker & Sidner, 1996) found two results that became touchstones for future investigations and designs. One was a phenomenon which they termed *e-mail overload:* People often used e-mail for several distinct goals, often far removed from interpersonal communication. (Unfortunately, *overload* has itself become an overloaded term, used also to describe the feeling of being overwhelmed by a high volume of e-mail; in this article, we adhere to the original meaning.) For example, inboxes became organizational devices for many people in their study, serving as "to-do" lists. Gruen, Sidner, Boettner, and Rich (1999) corroborated this usage. Overload is closely related to—in fact, enabled by—the phenomenon of "reinvention," (Sproull & Kiesler, 1991) in which users take features intended for one purpose and use them for another. An example of how users reinvent to support overloaded functions is how people often mark an important message as "unread" so it will be visually distinctive and continue to catch their attention—completely changing the meaning of the "unread message" marker.

Whittaker and Sidner reported another result that hinted at a broad theme in CUE's future research: idiosyncratic usage. When they investigated the standard e-mail folder-based organization scheme, they found that many users gave up filing altogether—but many other users did not give up, filing their messages frequently. Yet a third group performed occasional massive reorganizations. This threefold structure, with no group in the majority, is an excellent example of individual differences in e-mail habits.

Even users who were willing to file are a heterogeneous group. Fisher and Moody (2002) studied folder structures in detail and found a median of 73 folders with somewhere between 100 and 6,000 messages in each folder. Bälter (1998), who implemented a prototype foldering system while a visiting researcher in our group in 1999, made an additional important point: Many times users file their messages initially but eventually decide that it is not worth the effort. In a sense, this is another type of idiosyncratic usage: Even for a single person and a single task, usage patterns may change significantly over time as the e-mail environment becomes more crowded and less well-organized.

Further evidence of e-mail overload comes from analysis of messages themselves. In Chu, Eagan, Stern, and Moody (2003), 15 e-mail users were asked detailed questions about the purposes of specific e-mail messages. Hand-clustering of the survey results revealed three rough categories of messages: those with a workflow character that required a response, those that provided information with no response expected, and those that are part of a traditional conversation. Within these, 12 fine-grained categories were found, ranging in character from negotiation of meeting times to automatically generated records such as pay stubs.

These examples show that e-mail technology is used in idiosyncratic ways and for diverse purposes, with individual features and components of a program being reused in ways that might surprise their designers and developers. An e-mail program is a system in itself, with features whose value lies in their interaction and reinvention. One implication, which we explore further in the second half of this article, is that feature-by-feature design is untenable; instead, any feature must be designed with careful attention to its relation to other components. Another important implication is that design research must be intimately tied to empirical verification in real-world settings. It is easy to imagine a scenario in which a designer invents a new type of inbox that does an excellent job of showing unread messages but does a poor job as a "to-do" list. Such a design would look good to the designer and might fare well in a highly-targeted lab study. In a real-life deployment, however, the new inbox might be a disaster, because it did not support "to-do" list functionality.

3. A SOCIAL AND COLLABORATIVE PERSPECTIVE

A different set of data on e-mail usage came from studying groups who shared a single inbox. This practice occurs in many situations, ranging from an executive who allows an assistant to handle her mail to a group of workers at a help desk who respond to the same pool of incoming messages. Our lab was led to focus on these situations for several reasons. First, we believed that watching humans help each other would provide insight into how a computer could provide assistance. Second, our lab's connection with the Lotus Notes platform provided a natural impetus to study collaboration. Finally, informal feedback from customers indicated that there was a business need to support group inboxes. Our studies in this area proved valuable for several reasons: Not only did they provide new evidence of overload and improvements aimed at collaborative use but they suggested directions of future improvements even of single-user mail.

Muller and Gruen (2002) interviewed several groups, including help desk workers and members of college student organizations, who shared an inbox for a single e-mail address. Several issues cited by participants were highly specific to the context of sharing, for example, the construction of a group identity and the use of canned or "boilerplate" text. Two oft-cited concerns, however, were the difficulty of following the chain of conversation sparked by an initial message to the shared address (so that a single person could handle an entire conversation) and the challenge of coordinating responses and workflow among several people. Although these problems were especially urgent in the context of sharing an inbox, they pointed to areas of investigation that might benefit individuals as well.

The CUE lab also conducted an in-depth study involving semistructured interviews with 16 assistants who helped high-level managers manage their e-mail (Muller & Gruen, 2002). Throughout the study, we were struck by the way features of e-mail that had been developed for a single user were exploited to support collaboration between assistants and their executives. A feature allowing received mail to be edited was used to communicate additional information about a message (e.g., "this person spoke with you at the trade show last week"). Messages were forwarded between executives and assistants as a way of drawing attention to them and to discuss how they should be handled. Folders were used by some executive-assistant pairs to create ad hoc workflows. For example, the executive would place an item to be printed into a folder marked "Print." The assistant would print the message and then remove it from the folder or place in it a folder marked "Done."

A separate series of studies (Gruen et al., 1999) was conducted in which managers and assistants were interviewed about their work and e-mail practices and observed going through their correspondence while seated together. (Such meetings were standard practice and not artificially scheduled for the study.) The study was conducted with the goal of informing the design of a software "agent" to help users with their e-mail work, by modeling the kinds of assistance the agent would provide on observations of how human assistants helped the people they supported.

The assistants helped their managers in many ways, but two particular types of action stood out. Assistants frequently directed attention, by prioritizing lists of messages, actively interrupting their managers, or highlighting important sections of a message, such as the mention of a particular colleague. They also frequently spent time in "context creation," gathering additional pieces of information relevant to a particular message. Both behaviors address problems that could potentially be solved through software solutions as well; indeed, our work on threads, described later, can be viewed as a method of context creation.

The results of these studies on collaboration have two main implications. However, they emphasize the system-level perspective, articulated by Sproull and Kiesler (1991), that e-mail is woven into the general corporate system of coordinated activity. If our customers are to be believed, the design needs for coordination have been insufficiently explored. At the same time, it is evident that there is a fluid interplay among e-mail capabilities aimed at individuals and those aimed at groups. Solving a problem that is urgent for groups, such as tracking the context of a conversation, may turn out to be helpful for individuals as well.

4. TECHNICAL CONSIDERATIONS FOR ENTERPRISE SOFTWARE

We have so far discussed aspects of the e-mail design space that relate to usage patterns. An additional set of constraints stems from the technical and logistical considerations that are involved in creating enterprise software. We describe three examples, following, of features that were implemented in prototypes, liked by users, yet did not end up as part of a shipping product. In each case, what appeared to be "mere implementation details" turned out to be major roadblocks in moving from a prototype implementation to a scalable, enterprise-wide product. One common theme is that prototyping is often done using client-side technology, whereas current architecture for real-world implementation of large e-mail systems typically shifts much computation to banks of servers. Unfortunately, the mismatch between client and server technologies, often downplayed by researchers, turns out to be highly significant.

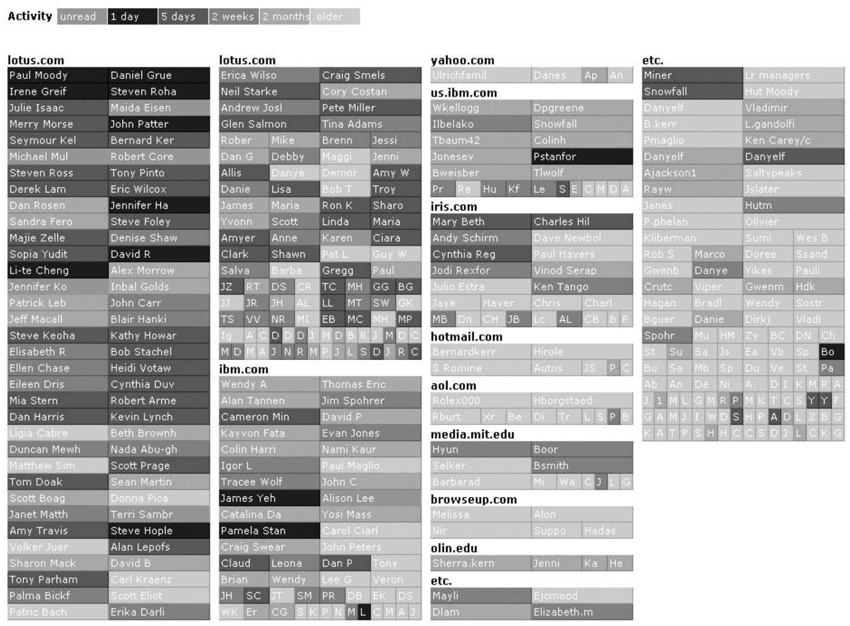
4.1. Example 1: Information Visualization

A frequent suggestion for improving the e-mail experience is the use of information visualization technology. We believe that is a good idea and the Thread Arcs case study in the second half of this article shows how visualization may succeed in practice. Nonetheless, the literature is filled with examples of visualizations that have not appeared in a product. Examples range from simple timeline-based systems, such as TimeStore (Yiu, Becker, Silver, & Long, 1997), to the elegant and information-rich views of PostHistory and Social Network Fragments (Viegas, Boyd, Nguyen, Potter, & Donath, 2004). Here we discuss an example of an e-mail visualization from our own lab (Figure 1) that was rejected by the product team and some lessons we learned in the process that may be relevant to other visualization projects.

This visualization, dubbed the *Correspondent Map,* is a kind of cross between a treemap representation (Shneiderman, 1992) and the Contact Map of Nardi et al. (2002). It displays all the people a user has corresponded with over the past year, with each correspondent represented by a small blue rectangle. The size of a rectangle indicates the volume of correspondence and the shade indicates how long ago the last e-mail occurred. (Orange rectangles represent unread messages regardless of age.) The rectangles are broadly arranged by Internet domain and within each domain listed by frequency of correspondence. The goal of the map is to provide a natural overview of one's inbox, organized by people.

The visualization also allows several convenient methods of interaction. Moving the mouse over a person's rectangle brings up a quick list of recent

Figure 1. **Correspondent Map.**



mail. Right-clicking provides a variety of options for sending mail to a particular person; a shift-click method is also provided for graphically defining lists of recipients. Finally, drag-and-drop capabilities allow for a number of shortcuts to common operations. For example, dragging a file from the Microsoft Windows®file explorer onto a person's rectangle will open up an e-mail composition window with that file attached.

When we created this demo, we believed it might have real potential as a new feature. The product team, however, thought otherwise. The problems in transferring this technology potentially apply to many visualization efforts. The first objection raised was that the map's typography was fragile and might hinder internationalization. In particular, the readability of the names in the rectangles depended on a set of ad hoc tricks—heuristics specifying when to use a full name, part of a name, or initials—that would not generalize easily to other languages. The second objection was based on seeing the map in natural usage (in a prototype system, ReMail 2.0, described later). The screenshot in Figure 1 looks reasonable but shows the map at full-screen size. At a much smaller size (as would be appropriate for a typical "side panel" in an e-mail program), the map is significantly less valuable, largely because the labels are no longer readable when all the rectangles are tiny. This objection shows why it is so important to consider components as part of a system, as described in Section 1, rather than individually. A final problem is that the type of data required by the map—a full list of all correspondents, with additional summary data on the history of messages with each one—requires a large amount of memory and is not easily encapsulated in traditional mail programmer interfaces; adding the map to a product would have required both significant server-side programming and a strong commitment to a thick client interface.

4.2. Example 2: Text Analysis and Computational Limits

It has long been suggested that e-mail organization could benefit from automatic text analysis (e.g., Segal & Kephart, 2000, describe one shipping example). Beyond the need for tools to help information retrieval and organization, e-mail has additional structure that may facilitate computational analysis. Our lab has created several prototypes that use text analysis. In each case, we've observed the same pattern: intense initial interest and positive feedback, combined with a failure to become part of a shipping product. We believe that in each case, the same concerns—based on both software architecture and user-experience considerations—have arisen. We describe one simple example here.

The CUE lab observed that dates embedded in message text are a critical component of many e-mails. Indeed, for some of the types of e-mail identified in Chu et al. (2003), such as meeting negotiations and record-keeping messages, dates and times may be the most critical piece of information. For such mails, it would be useful to identify the dates for visual highlighting, search and retrieval, and calendaring. (Similar ideas can be seen elsewhere, e.g., in the context of note-taking in Lotus Agenda®). Because dates and times frequently are described informally ("lunchtime on Tuesday"), the identification task is not completely straightforward. Stern (2004) created a date recognition engine, based on a library of hand-tuned regular expressions, that achieved 85% precision and 96% recall.

When this feature was demonstrated to potential customers at a trade show and other corporate briefings, it generated tremendous interest from users who felt it would remind them of events they would otherwise miss. This type of customer feedback is usually influential on product team strategy but these features have not been adopted as of this writing. Two considerations seemed to influence the decision. First, as with the Correspondent Map, internationalization considerations were an issue; although handcrafting a set of regular expressions was feasible for a prototype in a single language, it was clearly problematic for a large set of languages. Second, and probably more important, was that Stern's (2004) methods (despite their efficacy in our client-side prototype), because of their reliance on regular-expression-based matching, were too computational-intensive for the implementation that would have been necessary in the large-scale server-side product architecture.

4.3. Example 3: Instant Search

A final example of an extremely popular feature that was not transferred to a product specification was an “instant search” box that was part of a larger prototype (ReMail 2.0, discussed later). Users could type a phrase into a text field and—keystroke by keystroke—a list of messages containing that phrase would be shown. (Similar features can be found, for example, in Microsoft Entourage®.) This fast, incremental search was considered by many users who applied it on their own mailboxes to be both pleasant and efficient for navigating a large inbox.

Despite wide support for this feature, it came at a computational and architectural cost. We had implemented it in a context where all message subject lines had already been loaded into main memory and so a search could execute in a fraction of second. Unfortunately, it was difficult to translate this strategy into a model where most computation occurred on servers. Performing a new search with every keystroke ran into constraints of bandwidth and server-side computational cycles. It might have been possible to overcome these hurdles but even then the latency associated with network operations would have meant that the search would often feel sluggish rather than instantaneous.

4.4. Moving Past Roadblocks

All three examples discussed earlier underline the slippery nature of research aimed at enterprise-wide systems. In each case, researchers created a fully-functioning prototype that, for unanticipated reasons, turned out not to scale. Some of the factors, such as concerns about internationalization, are relatively easy to take into account in future investigations by researchers. But the client-server mismatch is a deeper issue and one that we feel deserves further study. Are there architectures that might somehow combine the centralization of server-side computation with the ease of development of client-side software? It may be possible that the desirability of features such as “instant search” will foster the development of just such an architecture. Until such architectures are found, we believe that researchers need to take the server-side nature of many current large-scale implementations into account from the beginning.

5. THREAD ARCS: A CASE STUDY

We now turn to an extended case study of the development of a particular feature that has been handed off to the product team, a graphical navigation device called a *Thread Arc* that displays a small map of the structure of an e-mail conversation thread. The sequence of work on Thread Arcs began with an empirical investigation, continued with design sketches and small working prototypes, and culminated in a fully working model that was itself tested empirically. This was certainly not the only feature that our lab has worked on, and it was not the only piece of research to have an impact on the product team, but we have chosen to focus on the development of this one component as an illustration of a successful strategy that addresses the three themes of enterprise e-mail: overload, collaboration, and technical constraints.

5.1. Empirical Investigation

It is widely recognized that conversation threads are a helpful organizational device, both in e-mail and other online communication media such as Usenet news. (Among many examples are (Donath, Karahalios, & Viegas, 1999; Smith, Cadiz, & Burkhalter, 2000; Venolia & Neustaedter, 2003.) In empirical studies performed by our lab, we too had encountered our own strong hints that threads were critical. For instance, the investigations of shared inboxes had emphasized the utility of keeping reply chains in one place and the studies of managers and assistants demonstrated the value of gathering context around an individual message.

Several factors make threads particularly central to e-mail use. They combine messages that are typically related both in content and in workflow, they are easily understood by end users, and there are well-known algorithms to compute threads efficiently (Zawinski, 1997). For these reasons and others, thread-related interfaces are part of many current e-mails programs (e.g., Lotus Notes and Microsoft Outlook®).

One of our first pieces of work on threading was a quantitative study, in keeping with our belief in grounding design in the analysis of real-world systems. In 2001, Fisher and Moody (2002) created a software tool to analyze e-mail archives of a set of participants taken from within IBM. Faced with the challenge of obtaining enough data to calculate structural relations in their participants' e-mail while preserving the privacy of the e-mail content, they created a data extraction program that participants ran against their own mail which encrypted all sender and recipient names and subject line text, gave each message a unique identifier, and preserved links between parent and response messages. Thus the data files accessed by the experimenters contained the necessary structural information to analyze thread sizes and structure without disclosing any recognizable information about people or topics of conversation. Sender and recipient names, for example, were converted to one-way hashes, a method which ensured privacy but allowed cross-referencing between users so the experimenters could see how different users viewed any given conversation thread.

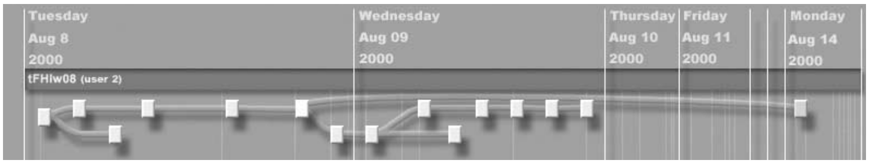
The analysis showed that approximately a third of messages that had been retained over a 2-month period were elements of threads with more than one message. Even these "nontrivial" threads were typically short, with 56% containing only two messages and only 13% containing five or more messages. Two months of e-mail messages were collected from 57 participants (ranging from 85 to 3,318 messages per user, reflecting differences both in how much mail participants received and how much they retained.) The data collected helped to establish parameters for the number of messages a thread visualization would typically need to support. In particular, it suggested that a visualization should be optimized for small numbers of messages, with the ability to scale to very complex threads taking a second priority.

Subsequent studies within our group have corroborated Fisher and Moody's (2002) findings (Kerr, 2003). In addition, these studies revealed a bimodal distribution of threads: thread trees tend to be either "bushy" (many messages are a response to a single document) or deep (each message has a single reply, as in a chain of conversation). As suggested by Venolia and Neustaedter (2003), this dichotomy may reflect different common uses of e-mail; for example, informing a group of users and requesting feedback from each versus having an extended back-and-forth conversation with a small number of users. It may also depend on the particular threading algorithm we used (Zawinski, 1997), because in certain cases, when a message from a thread has been deleted, different algorithms may guess at different reply structures.

5.2. Design Mock-Ups

Besides knowledge of thread statistics, Fisher and Moody's (2002) study had a second benefit, namely the creation of a large reference database of sample threads that could be used to create simple, stand-alone mock-ups of designs that exploited thread structure. Several design mock-ups were created in 2001, including a "map" of a thread (see Figure 2). This diagram showed both reply relationships and chronology information by displaying a tree overlaid on a timeline. The color-coding of the nodes represents the relationship of the message senders to the recipient. Note that the time scale on the x-axis is nonlinear, so that days with little or no activity are shown compressed. This avoids the problem of large gaps in the time display experi-

Figure 2. **Thread tree superimposed on a timeline.**



enced in other systems which have used timelines (Kudo, Tanaka, & Kosecki, 1997; Jovicic & Baecker, 1999).

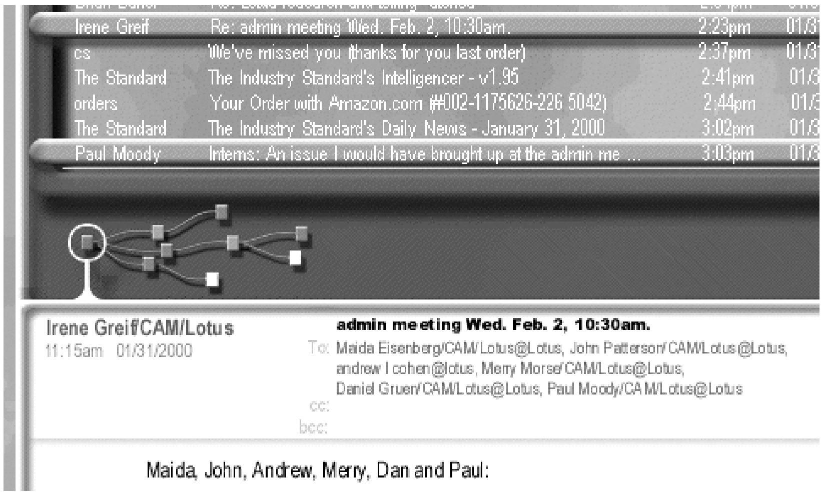
Other mock-ups from the same period explored how a visualization of the conversation thread (or "thread map") might fit into an e-mail client and interact with other components. An example can be seen in Figure 3. In addition to drawing a map of a thread, all of the messages in the thread were highlighted in the inbox.

5.3. Vision Piece

A lightweight method of design exploration is the creation of inspirational mock-ups of potential future features, made without the constraints of actual implementation. (One famous example is the Apple® Knowledge Navigator video [Field, 1987]; Vannevar Bush's [1945] description of a hypothetica "Memex" may be considered another.) In 2001, we created our own such "vision piece" (see Figure 4), which was shown to a large audience at the Lotusphere 2002 trade show.

Our experience provides an excellent example of why this method is valuable. The mock-up showed a hypothetical e-mail client integrating mail, chat, and presence awareness, along with several "magical" features such as the ability to automatically recommend who should participate in a meeting. Audience reaction was very positive to all features but the one that drew an audible gasp was the ability to send an entire thread of mail to another person by dragging an iconic representation of the thread into a chat window—ironically, the single most easily-implemented aspect of the demo! This reaction underscored the importance of threads as an easily-grasped organizing principle. It also seemed to highlight the importance of e-mail as a collaborative system: Dragging a thread into a chat window illustrates cooperation between user interface components that usually are separate, as well as the importance of features that go beyond one-to-one, asynchronous communication. Although an audience gasp is anecdotal in the extreme, it was clear that the broad positive reaction to the vision piece as a whole helped engage the attention of the product team—probably far more than any of the studies cited earlier.

Figure 3. **Early design prototype showing thread visualization incorporated into an e-mail client.**



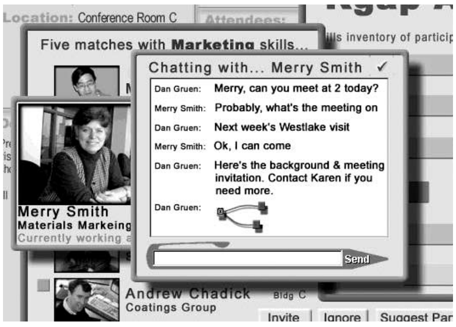
Irene Greif Re: admin meeting Wed. Feb. 2, 10:30am. 2:23pm 01/3
cs We've missed you (thanks for you last order) 2:37pm 01/3
The Standard The Industry Standard's Intelligencer - v1.95 2:41pm 01/3
orders Your Order with Amazon.com (#002-1175626-226 5042) 2:44pm 01/3
The Standard The Industry Standard's Daily News - January 31, 2000 3:02pm 01/3
Paul Moody Interns: An issue I would have brought up at the admin me ... 3:03pm 01/3

Irene Greif/CAM/Lotus
11:15am 01/31/2000

admin meeting Wed. Feb. 2, 10:30am.
To: Maida Eisenberg/CAM/Lotus@Lotus, John Patterson/CAM/Lotus@Lotus, andrew l cohen@lotus, Merry Morse/CAM/Lotus@Lotus, Daniel Gruen/CAM/Lotus@Lotus, Paul Moody/CAM/Lotus@Lotus
cc:
bcc:

Maida, John, Andrew, Merry, Dan and Paul:

Figure 4. **Portion of a scene from the vision piece, showing a chat that resulted from a meeting announcement that had arrived in e-mail. Note how the thread containing the meeting announcement and related documents has been dragged into the chat as a single entity.**



Location: Conference Room C
Five matches with **Marketing** skills...
Chatting with... Merry Smith ✓

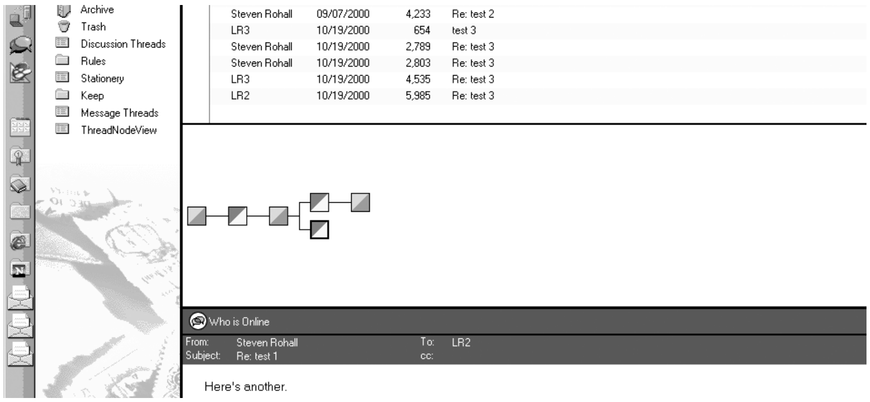
Dan Gruen: Merry, can you meet at 2 today?
Merry Smith: Probably, what's the meeting on
Dan Gruen: Next week's Westlake visit
Merry Smith: Ok, I can come
Dan Gruen: Here's the background & meeting invitation. Contact Karen if you need more.
Dan Gruen:

5.4. Prototypes

Modifications to Existing E-Mail Programs

In keeping with our belief that e-mail designs need to be tested as part of a functioning system, we built a thread viewer as a modification to the standard Lotus Notes e-mail template (Figure 5). This visualization shows messages as square nodes in a tree and displays information on senders and recipients us-

Figure 5. **Thread visualization integrated into the Lotus Notes® R5 mail template (2000).**



```
Steven Rohall   09/07/2000   4,233   Re: test 2
LR3             10/19/2000     654   test 3
Steven Rohall   10/19/2000   2,789   Re: test 3
Steven Rohall   10/19/2000   2,803   Re: test 3
LR3             10/19/2000   4,535   Re: test 3
LR2             10/19/2000   5,985   Re: test 3
```

From: Steven Rohall To: LR2
Subject: Re: test 1 cc:

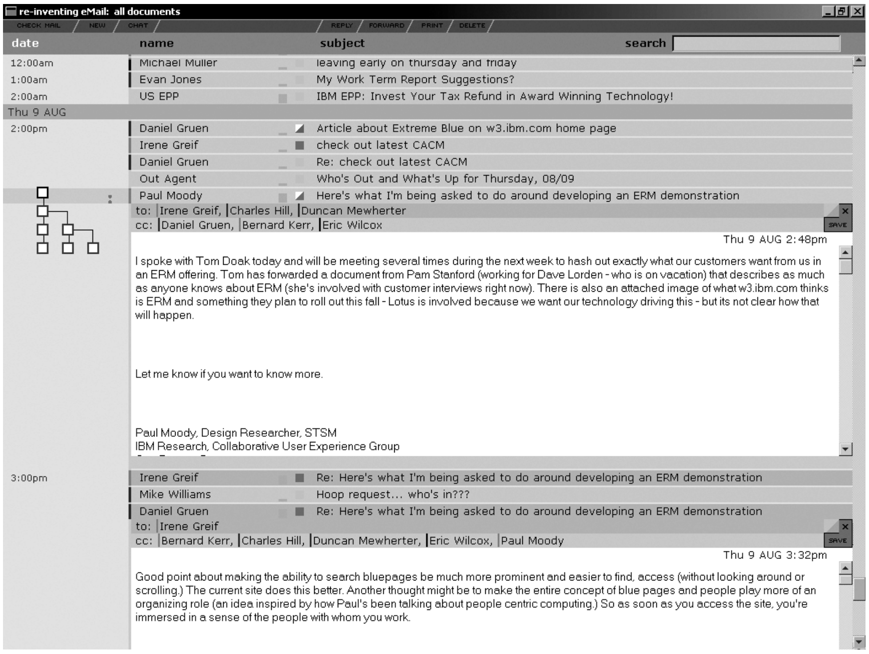
Here's another.

ing a dual color scheme. The color of the upper left corner of a node represents information about the sender and the color of the lower right corner represents information about the receiver. The prototype turned out to be valuable at identifying faults in the interface. For example, it became immediately clear that the color scheme was problematic for the many messages with multiple recipients; in addition, there was widespread confusion about the color scheme in general. Just as bad, technical constraints limited our ability to assess the new view's benefits. For instance, it proved difficult, in the context of Notes, for the thread view to update to reflect the thread of the currently selected message in the message list.

ReMail 1.0 Prototype

The technical constraints that hindered the Notes helper component convinced us that we needed an e-mail client that would provide us complete control over the interface. In the summer of 2001, work began on the "ReMail 1.0" (for "reinventing e-mail") e-mail client utilizing dynamic HTML and JavaScript® for the user interface to allow rapid development and Lotus Notes for the underlying e-mail storage, so that users could try the system with their real corporate mail (see Figure 6). In addition to thread-based navigation, ReMail 1.0 incorporated several other features that are beyond the scope of this article, including methods of message preview, capabilities for user annotation of messages, synchronous collaboration, and integration of chat and e-mails. Here we focus on the novel aspects relating to threads but further details may be found in Rohall, Moody, Gruen, and Kellerman (2001).

Figure 6. **ReMail 1.0 prototype (2001).**



12:00am | Michael Muller | leaving early on thursday and friday
1:00am | Evan Jones | My Work Term Report Suggestions?
2:00am | US EPP | IBM EPP: Invest Your Tax Refund in Award Winning Technology!
Thu 9 AUG
2:00pm | Daniel Gruen | Article about Extreme Blue on w3.ibm.com home page
Irene Greif | check out latest CACM
Daniel Gruen | Re: check out latest CACM
Out Agent | Who's Out and What's Up for Thursday, 08/09
Paul Moody | Here's what I'm being asked to do around developing an ERM demonstration
to: Irene Greif, Charles Hill, Duncan Mewherter
cc: Daniel Gruen, Bernard Kerr, Eric Wilcox
Thu 9 AUG 2:48pm

I spoke with Tom Doak today and will be meeting several times during the next week to hash out exactly what our customers want from us in an ERM offering. Tom has forwarded a document from Pam Stanford (working for Dave Lorden - who is on vacation) that describes as much as anyone knows about ERM (she's involved with customer interviews right now). There is also an attached image of what w3.ibm.com thinks is ERM and something they plan to roll out this fall - Lotus is involved because we want our technology driving this - but its not clear how that will happen.

Let me know if you want to know more.

Paul Moody, Design Researcher, STSM
IBM Research, Collaborative User Experience Group

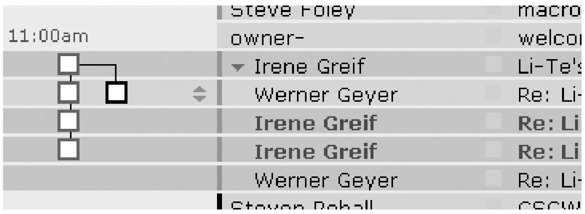
3:00pm | Irene Greif | Re: Here's what I'm being asked to do around developing an ERM demonstration
Mike Williams | Hoop request... who's in???
Daniel Gruen | Re: Here's what I'm being asked to do around developing an ERM demonstration
to: Irene Greif
cc: Bernard Kerr, Charles Hill, Duncan Mewherter, Eric Wilcox, Paul Moody
Thu 9 AUG 3:32pm

Good point about making the ability to search bluepages be much more prominent and easier to find, access (without looking around or scrolling.) The current site does this better. Another thought might be to make the entire concept of blue pages and people play more of an organizing role (an idea inspired by how Paul's been talking about people centric computing.) So as soon as you access the site, you're immersed in a sense of the people with whom you work.

Although the ReMail 1.0 prototype never became sufficiently polished for day-to-day e-mail use, we were able to gather useful feedback through a more limited deployment where we allowed users to run it on a subset of their inbox data and through semistructured interviewing. In addition, we received informal feedback on the prototype and its features at the Lotusphere 2002 trade show, where we allowed potential customers to try out the application. In both cases, we opted to gather general informal commentary rather than run focused, controlled experiments. The main reason was simply that we felt this prototype was in the early stages of design, so we wanted to cast as broad a net as possible.

The ReMail 1.0 interface relied strongly on thread structure, which it exploited in several ways. When a message was selected, a thread map visualization would be drawn next to it. The map was drawn with a transparent background, so that even for large threads, it would not completely obscure the rest of the interface. Selecting a node in the thread map would cause the message list to scroll to the selected message. Users liked this feature, particularly when a thread was long and not all of the messages were visible in the message list.

Figure 7. **A gathered thread, expanded so as to show all thread messages.**



```
               Steve Foley        macro
11:00am        owner-             welcoı
               ▾ Irene Greif      Li-Te's
                 Werner Geyer     Re: Li-
                 Irene Greif      Re: Li
                 Irene Greif      Re: Li
                 Werner Geyer     Re: Li-
               Steven Rohall      CSCW
```

In addition, when the user selected a message, the other messages in that thread shown in the inbox were highlighted in a subtle brown (darker gray in the diagram). Users could also "gather" a thread with a single click, so that all messages were placed together instead of being scattered throughout the message list. A gathered thread is shown in Figure 7.

The gathered thread was placed in the list at the location of the most recently-received message. This feature allowed users to keep track of an active thread without cluttering their message list. By default, a gathered thread was drawn in a collapsed fashion taking up two lines in the message list, one for the original "root" message and one for the most recent message in the thread (Microsoft Outlook now has a similar feature). As with the map, the ability to easily gather a thread within the inbox echoed the "context creation" that was observed in human assistants in earlier studies (Gruen et al., 1999).

One frequent comment from potential customers turned out to be helpful. The simplified thread map in ReMail 1.0 was essentially topological, emphasizing the parent–child relationships in a thread. This was not so much a conscious design decision as a compromise forced by implementation constraints but it turned out to serve as a useful test. Many people who tried the prototype at the Lotusphere trade show commented that the tree structure obscured timing relationships, which they believed to be at least as important—giving substance to our earlier intuitions. This type of comment is exactly what we hoped to elicit through informal interviews and is one we might have missed in a more narrowly focused test.

ReMail 2.0 Prototype: New, Stand-Alone, E-Mail Client

Although the ReMail 1.0 prototype was a helpful platform for experimentation, it soon became clear that for a realistic test of new components—including thread-based navigation and organization—we would need an architecture able to work with users' full existing e-mail databases. Most of our potential users had thousands or even tens of thousands of archived messages. The ReMail 1.0 interface, built on DHTML, was not able to scale to such large sets of data. Users also informed us that because ReMail 1.0 lacked certain features, such as folders, they would be unlikely to use the prototype for an extended period of time. Unfortunately, implementing these critical components in JavaScript seemed problematic.

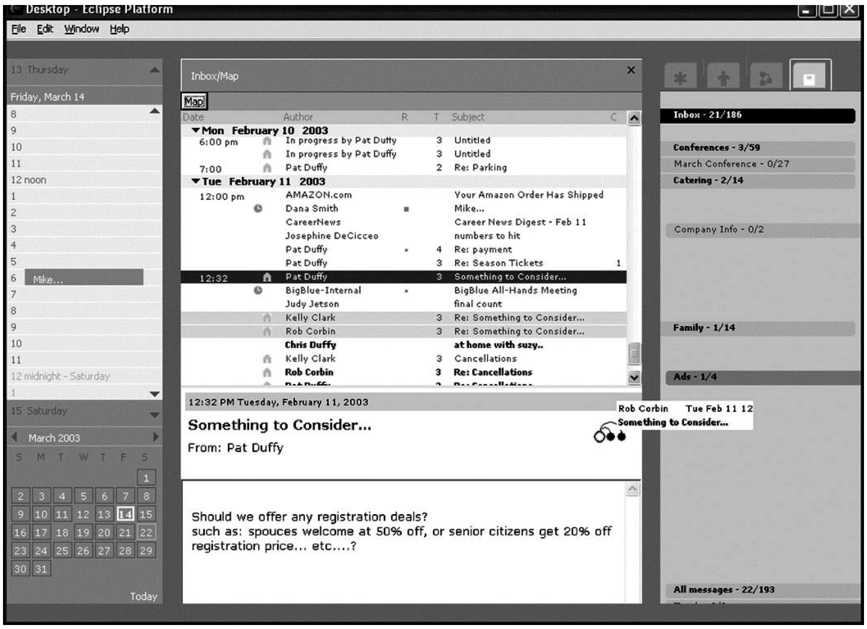
Because of these concerns, we decided that we needed a new testbed system and that it should be a ground-up rewrite. In 2002, we began a new prototype, dubbed ReMail 2.0. Figure 8 shows a screenshot. The new prototype was built with a Java user interface on the Eclipse platform and used a relational database for flexible searching and scalable storage of messages. Messages were pulled from a range of sources, including Lotus Notes mail, Lotus QuickPlaces®, and four standard mail and news formats. ReMail 2.0 was installed and used by dozens of users within IBM, who provided copious feedback (Gruen et al., 2004).

Although our focus in this article is on the parts of the interface related to threading, a brief walk-through of the application will be helpful in what follows. Following convention, the interface includes a list of messages (top center) and a pane for displaying message bodies (bottom center). A calendar is anchored on the left. The message list is tightly integrated with the calendar, allowing messages to be dragged and dropped onto a day or time to create a calendar item; conversely, calendar items can be clicked to scroll to relevant messages. This feature, explicitly meant to support the kind of overload found in Whittaker and Sidner (1996), was one of the most frequently praised by our users. On the right is a pane for organizing messages into folder-like groups. (For further details on the ReMail 2.0 interface, see Kerr, 2003.)

More importantly, the flexibility of a Java-based user interface gave us a chance to try several different ways to map a thread. One method was a revisiting of the diagram in Figure 2, showing thread-tree topology using a branching diagram. A second method was the Thread Arc (Figure 9). A Thread Arc shows messages as dots, reply relationships via curving links, and message sequence by a linear left-to-right arrangement (addressing the objections of users of the ReMail 1.0 prototype). The fact that it shows sequence rather than arrival time distinguishes it from such visualizations as (Smith et al., 2000) and is the key to its compactness. Its compactness, in turn, was viewed positively by the product team. Unlike thread visualizations such as Figure 2, a Thread Arc (Figure 9) can fit into a small corner of the message-body pane. As with the Correspondent Map, discussed earlier, the fact that we could observe Thread Arcs as part of a larger system was critical to discovering the importance of a visualization scaling down in physical size.

The dots that represent messages show the selected message as a hollow circle and can be colored according to different variables; for example in Figure 9, the same thread is shown with message one and then message five selected. They are also clickable, allowing easy navigation between messages in

Figure 8. **ReMail 2.0 prototype (2002).**



13 Thursday

Friday, March 14

8
9
10
11
12 noon
1
2
3
4
5
6 Mike...
7
8
9
10
11
12 midnight - Saturday
1

15 Saturday

March 2003

S	M	T	W	T	F	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Today

Date	Author	R	T	Subject	C
Mon February 10 2003					
6:00 pm	In progress by Pat Duffy		3	Untitled	
	In progress by Pat Duffy		3	Untitled	
7:00	Pat Duffy		2	Re: Parking	
Tue February 11 2003					
12:00 pm	AMAZON.com			Your Amazon Order Has Shipped	
	Dana Smith	■		Mike...	
	CareerNews			Career News Digest - Feb 11	
	Josephine DeCicceo			numbers to hit	
	Pat Duffy	▪	4	Re: payment	
	Pat Duffy		3	Re: Season Tickets	1
12:32	Pat Duffy		3	Something to Consider...	
	BigBlue-Internal	▪		BigBlue All-Hands Meeting	
	Judy Jetson			final count	
	Kelly Clark		3	Re: Something to Consider...	
	Rob Corbin		3	Re: Something to Consider...	
	Chris Duffy			**at home with suzy..**	
	Kelly Clark		3	Cancellations	
	Rob Corbin		**3**	**Re: Cancellations**	

12:32 PM Tuesday, February 11, 2003

Something to Consider...

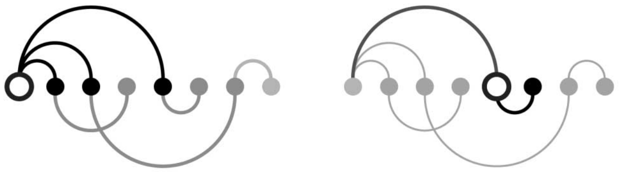
From: Pat Duffy

Rob Corbin Tue Feb 11 12
Something to Consider...

Should we offer any registration deals?
such as: spouces welcome at 50% off, or senior citizens get 20% off
registration price... etc....?

Inbox - 21/186
Conferences - 3/59
March Conference - 0/27
Catering - 2/14
Company Info - 0/2
Family - 1/14
Ads - 1/4
All messages - 22/193

Figure 9. **Thread Arc design.**



a thread, and provide message summary information via a "hover over." This level of integration—exactly what had been missing in the first Notes-based thread prototype—was broadly liked.

5.5. Thread Arc Testing

Many features of ReMail 2.0 drew interest, including the thread maps. There was a corporate desire, however, to perform further testing before new features, including thread maps, were incorporated into a product.

The CUE lab conducted a formal usability test of a small group ($n = 5$) of users to determine the utility of the new features, with special emphasis on the thread. These tests were run on a preconstructed e-mail database in a usability lab and took between 2 and 3 hr. As part of the test, the ReMail 2.0 prototype was configured to include a choice of two kinds of thread maps to determine which would be most advantageous. The results showed that users like the ability to easily navigate among related items and the Thread Arcs were slightly preferred to an alternative tree diagram used (which was similar to Figure 3).

Kerr (2003) then built a stand-alone working prototype that would allow users to see visualizations of their own mail in the context of an e-mail client. Eight users were shown this prototype and allowed to switch between different visualizations methods; after seeing the results, they were asked to rate and compare the visualizations. They were also asked about key qualities that they felt were important in visualizing their e-mail threads. The results confirmed the importance of showing the chronology of messages and the need to show both bushy and narrow threads compactly. It also suggested that Thread Arcs were well suited for the size and type of conversations found in users' real mail and, on balance, did a good job at satisfying all of the qualities that users valued in small-scale thread visualizations.

Finally, when the ReMail 2.0 client was complete, a general study was performed (Gruen et al., 2004). Here users ($n = 9$) were asked to use ReMail 2.0 as their primary client for over a week and were then interviewed about their experience with its advanced feature set. Once again, the feedback consistently pointed to the value of connecting related items in conversational thread and the value of the Thread Arcs.

The combination of these studies gave the product team more confidence in the Thread Arc concept and they began work on implementation using our prototype code as their starting point—completing a sequence of research begun years before.

5.6. What Makes a Successful Strategy?

Our strategy throughout the development of Thread Arcs has been directed by the three themes of enterprise research identified earlier. First, knowing the pitfalls of overload and individual differences, we have taken a system-oriented approach to design and assessment and have expended a great deal of development effort to see how various thread navigation devices interacted with other components under realistic usage. Second, the design of the thread map reflects collaborative concerns. A thread map is really a map of collaboration; additionally, the issues of context and connection addressed by the thread map arose in all our studies of collaboration. Finally, the fact that Thread Arcs were transferred to the product shows both a good choice of problem area—the small amount of information needed to create a thread map is compatible with a client-server architecture—and considerable effort by our researchers working directly with server-side product-team coders to resolve technical issues.

This last point raises an important question: To what extent should practical engineering considerations influence research, especially research on user interfaces? Although there are obvious benefits to presenting a product team with feature suggestions that are easily implemented, it is arguably at least as important for a research team to push beyond what is merely easy. Indeed, classic vision pieces such as the Apple Knowledge Navigator video present interfaces which we still do not know how to implement—yet they have been enormously influential. Our experience suggests a rule of thumb: It is fine for a research team to work on hard-to-implement ideas provided that it is always understood what are the obstacles to implementation. In the case of the Correspondent Map, blithe assumptions about client-side processing and use of the English language led us to ignore interesting and important considerations faced by many visualizations. However, the ReMail vision piece was effective despite showing certain "magical" elements, partly because it was made clear just how far it was from easy realization. To sum up, researchers should push technological boundaries, but they can only do so effectively if they understand where the boundaries lie.

6. CONCLUSION

6.1. The Pragmatic System-Based Approach

This article has provided an introduction to the perspective, opportunities, and constraints inherent in research aimed at influencing very large-scale e-mail systems. We have highlighted three key challenges: the multifaceted and idiosyncratic nature of e-mail usage by individuals, the fact that e-mail is not just a communication device but often a collaborative system used to coordinate group activity, and the existence of constraints stemming from the need to scale to large, diverse audiences. We have described various methodologies to meet each of these challenges: the importance of empirical work to understand the full complexity of the e-mail system, the necessity of building working prototypes to understand how new features affect a functioning system, and the importance of technical simplicity and scalability for distributing features widely. We have also described some of the tactics required for new ideas to become adopted in nonacademic settings—the Lotusphere "vision piece," for example, is very far from a peer-reviewed research article but certainly played a role in the transfer of ideas from our research lab to the development team.

6.2. Future Directions

We believe that the system-oriented approach to e-mail suggests several promising directions for future investigation. First, in many settings—such as a large corporation—significant "extra" data is available beyond what is found in standard e-mail headers. To give one example, corporate directories can be used to extract a variety of data on senders and recipients. Thus in the future, it may be desirable to automatically highlight messages that come from a user's supervisor or otherwise use data from an organizational chart for display and message retrieval.

Second, we feel that more attention is needed to the case of e-mail as a collaborative activity. We have found that e-mail inboxes are often used collaboratively and relatively little development has gone into creating features that ease the special problems of cooperative and concurrent use.

Third, there is currently a mismatch between the technical architecture used in implementing large-scale systems (distributed computation performed on sets of servers) and the client-centric architecture of many popular e-mail systems. Although moving computation to the server side has significant benefits to a corporation, it has also—in our experience—led to increasing difficulty in adding graphical, data-intensive, or computation-heavy features to e-mail programs. We believe this tension—between the convenience of centralized architectures and the power of client-side ones—will turn out to be an important theme in coming years and that it is important to find techniques to exploit the benefits of both.

Finally, we have found a consistent theme that users need help navigating the e-mail system, both individually and (as with shared inboxes) in the context of a workflow. Given the current flood of e-mail and the need to respond in near real-time, we believe aids for managing attention and coordination will become increasingly critical. We are currently working on a new framework for semiautomatic classification of e-mails into particular "activities," allowing both organization and integration into other tasks performed in a corporate setting.

NOTES

Acknowledgments. Thanks to everyone in the Collaborative User Experience group who participated in our research, many of whom are mentioned by name in the text. Special thanks to Paul Moody, Bob Stachel, Kushal Dave, Seymour Kellerman, Steve Foley, Robert Armes, John Patterson, Mia Stern, Eric Wilcox, Carol Galvin, Sandra Kogan, and Irene Greif. And thanks to interns Suzanne Minassian, Derek Lam, Kayvon Fatahalian, Evan Jones, Shaun Park, Ka-Ping Yee, Selina Chu, James Eagan, and Regina Tassone.

Background. The IBM Collaborative User Experience Research group was formerly known as Lotus Research.

Authors' Present Addresses. Martin Wattenberg, Steven Rohall, Daniel Gruen, Bernard Kerr, IBM Research, One Rogers Street, Cambridge, MA 02142. E-mail: mwatten@us.ibm.com.

HCI Editorial Record. First manuscript received June 16, 2003. Revision received April 14, 2004. Accepted by Steve Whittaker. Final manuscript received November 15, 2004. — *Editor*

REFERENCES

Bälter, O., (1998). *Electronic mail in a working context.* Unpublished doctoral dissertation, Royal Institute of Technology, Stockholm.

Bush, V. (1945, July). As we may think. *The Atlantic Monthly, 176*(1)*,* 101–108.

Chu, S., Eagan, J., Stern, M., & Moody, P. (2003). *Classifying with email types: A user study* (IBM CUE Technical Report).

Donath, J., Karahalios, K., & Viégas, F. (1999). Visualizing conversations. *Proceedings of the HICSS-32 1999 Conference.*

Field, R. (1987) *Knowledge navigator.* Cupertino, CA: Apple Computer.

Fisher, D., & Moody, P. (2002). *Studies of automated collection of email records* (UCI ISR Technical Report ISR-02-04). Irvine, CA: Institute of Software research.

Gruen, D., Rohall, S. L., Minassian, S., Kerr, B., Moody, P., Stachel, B., et al. (2004). Lessons from the ReMail prototypes. *Proceedings of the 2004 Conference on Computer Supported Cooperative Work.* New York: ACM.

Gruen, D., Sidner, C., Boettner, C., & Rich, C. (1999). A collaborative assistant for email. *Proceedings of the CHI 1999 Conference on Extending Abstracts on Humn Factors in Computing Systems.* New York: ACM.

Jovicic, S., & Baecker, R. (1999, June). Time-Based archiving and retrieval of email. *Workshop on history-keeping in computer applications.* Workshop presented at the HCI Lab, University of Maryland, College Park, MD.

Kerr, B. (2003)., THREAD ARCS: An email thread visualization. *Proceedings of the IEEE Symposium on Information Visualization.* New York: ACM.

Kudo, M., Tanaka, X., & Koseki, Y. (1997). Information visualization for electronic mail management. *Proceedings of the Visual 1997 Conference.* New York: ACM.

Muller, M., & Gruen, D. (2002). *Collaborating within—not through—email: Users reinvent a familiar technology* (IBM CUE Technical Report 2002-10). Armonk, NY: IBM.

Nardi, B., Whittaker, S., Isaacs, E., Creech, M., Johnson, J., & Hainsworth, J. (2002). Integrating communication and information through ContactMap. *Communications of the ACM, 45*(4), 89–95.

Raymond, E. S. (2001). *The cathedral and the bazaar.* Sebastopol, CA: O'Reilly

Rohall, S. L., Gruen, D., Moody, P., & Kellerman, S. (2001). Email visualizations to aid communications. *Proceedings of the 2001 IEEE Symposium on Information Visualization.* New York: ACM.

Shneiderman, B. (1992). Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics, 11,* 92–99.

Smith, M., Cadiz, J. J., & Burkhalter, B. (2000). Conversation trees and threaded chats. *Proceedings of the 2000 Conference on Computer Supported Cooperative Work*. New York: ACM.

Sproull, L, & Kiesler, S. (1991). *Connections: New ways of working in the networked organization*. Cambridge, MA: MIT Press.

Stern, M. (2004). Dates and times in email messages. *Proceedings of the 9th International Conference on Intelligent User Interfaces*. New York: ACM.

Venolia, G., & Neustaedter, C. (2003). Understanding sequence and reply relationships within email conversations: A mixed-model visualization. *Proceedings of the SIGCHI 2003Conference on Human Factors in Computing Systems*. New York: ACM.

Viegas, F., Boyd, D., Nguyen, D., Potter, J., & Donath, J. (2004). Digital artifacts for remembering and storytelling: PostHistory and social network fragments. *Proceedings of the Hawaii International 2004 Conference on System Sciences*.

Whittaker, S., & Sidner, C. (1996), Email overload: Exploring personal information management of email. *Proceedings of the SIGCHI 1996 Conference on Human Factors in Computing Systems: Common Ground*. New York: ACM.

Yiu, K., Baecker, R., Silver, N., & Long, B. (1997). A time-based interface for electronic mail and task management. *Proceedings of the HCI International 1997 Conference on Design of Computing Systems*. Elsevier.

Zawinski, J. (1997). *Message threading*. Retrieved January 2004, from http://www.jwz.org/doc/threading.html