A Persistent Chat Space for Work Groups: The Design, Evaluation and Deployment of Loops

Thomas Erickson, Wendy A. Kellogg, Mark Laff,
Jeremy Sussman, Tracee Vetting Wolf, Christine A. Halverson, Denise Edwards
IBM T. J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598 USA
{snowfall|wkellogg|mrl|jsussman|tlwolf|krys}@us.ibm.com, dc.edwards@verizon.net

ABSTRACT
Loops is a text-based computer mediated communication system aimed at small- to medium-sized corporate work groups. We begin by discussing the goals of the system and the rationale behind its design, particularly its treatment of non-conversational text. Next we describe its realization in an implemented system, and discuss how an early working version of the system was 'group tested,' and the changes that lead to. We then discuss its deployment within our organization, and provide examples of how it's used. We conclude with reflections on the usage patterns of Loops and their implications for the design of similar systems.

Author Keywords
CMC, CSCW, Conversation, Chat, Design, IM, Instant Messaging, Social Proxy, Awareness, Online Environments

ACM Classification Keywords
H.5.3 [Information Interfaces and Presentation]: Group and Organization Interfaces – Computer supported cooperative work, evaluation/methodology, asynchronous interaction, synchronous interaction

INTRODUCTION
For the last several years we've been engaged in designing online conversation spaces for distributed work groups. Our aim is to design "socially translucent" systems [5, 6] – systems that provide a social context for interaction by providing cues about users' presence and activities. We claim that such systems can, by taking advantage of the human ability to draw inferences from traces of activity, support social processes (e.g., imitation; peer pressure) that permit groups to function effectively.

Our approach to making social information visible employs two tactics: social proxies and persistent conversation. Social proxies are minimalist graphical representations of the presence and activity of participants; their aim is to provide a sense of the activity in an online system without eliminating all vestiges of privacy. Persistent conversation refers to text-based computer mediated communication (CMC) that persists over time – that is, it is similar to chat except that all conversations are logged and are always visible to participants.

Up to this point our work has been embodied in a first-generation system called "Babble." Babble is an online, conversation-centric system designed to support small to medium-size workgroups. In a series of publications we've described the design of the system [7], the "social translucence" rationale behind it [5], and studies of deployments and adoption of the system [1]. In most of this work we have kept the focus on Babble's most notable feature, its social proxies.

This paper draws on this previous work, but opens up some new areas of discussion. Its primary aim is to describe the design and evaluation of a second generation system called Loops. This paper focuses on the way in which text has been used in unexpected ways in the Babble system, and the design of new interface elements for Loops intended to support the observed uses. We'll conclude by discussing some of the usage patterns in Loops, and taking a closer look at a particular Loops community.

BABBLE AND RELATED WORK
Before turning to design issues, we'll situate Babble and Loops relative to other work. While they are not quite like any other collaborative environment, they blend features from a variety of systems.

In terms of its functionality, Babble [7] resembles a multi-room chat system, with three differences. First, the conversation in Babble persists across sessions, supporting what we call a "blended synchrony" model of chat. That is, conversations may be synchronous or asynchronous, with their remarks being separated by seconds, minutes, days or months. Second, the structure of Babble's conversation space is user-definable: anyone can create, modify, rename, or reorganize rooms. Third, Babble uses visual cues to enhance its users' mutual awareness of one another's location, movements and activities, and to alert its users to the location of new information in the environment.

In terms of look and feel, Babble and Loops resemble multi-channel chat systems (e.g.,[23]), with their transcripts of real time conversation and their lightweight conversation model. They are also akin to the instant messaging systems that are now widely used in corporate work places [11, 14, 15], particularly in their ability to support both coordinating talk and in-depth work conversation. Babble's and Loops' user-extendable set of 'places' (i.e., discussion topics) resembles that of many online asynchronous conversation systems, and has similarities to MUDs and MOOs (e.g., [2, 3, 4, 21]), in that they persist over time, and support both opportunistic encounters and structured events.

The two tactics that Babble and Loops use to support mutual awareness – persistent conversation and the social proxy – likewise have a variety of antecedents. Persistent conversation hearkens back to the beginnings of online community in systems like EMISARI [12] and The Well [18], continues in applications like CSILE [20], CaMILE [9] and TeamRooms [19], and is most recently manifested in web boards and blogs. That text-based conversation is a rich source of social information has been well-documented, especially by Cherny [3], albeit in a non-persistent case. Second, the social proxies of Babble and Loops provide a number of visual cues about users in an attempt to provide increased awareness. In this they bear similarities to systems that support workspace awareness (e.g., [8]), and to work on visualizing chat users [22].

DESIGN RATIONALE

The design of Loops was shaped by our experiences with the Babble system. Over the five years of its project life, Babble was deployed to about two dozen groups, mostly within IBM. Deployments were to three sorts of groups: small, close-knit but distributed work groups; large, globally distributed communities of interest; and *ad hoc* task-forces that existed for a relatively short period of time. Several, though not all, of the deployments were studied (see [1, 6]). On the one hand, Babble's conversation model and social proxy seemed quite successful, but on the other there were a number of recurring problems that we wished to address in the next generation system.

Retaining and Enhancing What Worked

Our experience with and studies of Babble left us convinced that it got quite a few things right. The principle features of Babble that we wished to retain were:

- The lightweight, blended synchrony, just-start-typing conversation model.
- The social proxy and other features that created an awareness of participants' presence and activity.
- The sense of history and inhabitation that resulted from the persistence of conversation and other activity traces.

Four New Requirements

At the same time, our experience with Babble led us to several new requirements.

Supporting Deployment and Updating

In our five years of work with Babble, we repeatedly encountered difficulties in deploying and updating it. The Babble client, written in Smalltalk, was 2+ megabytes in size; to do an installation or an update, the users (some of whom were consultants and sometimes worked over dialup lines) had to download and run an install package. This left the timing of the install up to each person, and thus installs were often staggered across several days. This also made it more likely that those who installed Babble immediately would log on and, finding no one to talk to, not be inclined to return. In short, every release of a major update disrupted our users' communities and ran the risk of causing a deployment to fail. We wished to remedy this problem.

A Changeable Look and Feel

We wanted to be able to easily alter the look and feel of the user interface, and to allow our visual designer to directly work in the medium rather than creating design prototypes to be reinterpreted by a programmer. As noted by Houde and Sellman [13], most development environments do a good job of supporting design or programming, but not both. We value aesthetics, and wanted our next system to provide as much support for iteration in the visual design as it did in the functional design.

Support Membership in Multiple Communities

Initially, we had envisioned that Babble would be used as an online environment for distributed work groups, with each group having a single Babble. But as people in a workgroup became accustomed to Babble, it was common for them to want to adopt it for other uses such as large communities of interest that wanted a long-term collaborative space, and *ad hoc* task-forces that needed a collaborative space for projects of limited duration. It became apparent that those who found Babble useful wanted multiple Babbles for use with multiple groups. We resolved to address this need in the new system.

Support 'Publishing' of Non-Conversational Text

Another recurring problem, both in our own use of Babble as well as in virtually all of our deployments, was the need to 'publish' text outside of conversations. Let's explore the genesis of this requirement in more detail.

To begin with, we need to say a bit more about how structure is created in Babble. Just as operating systems allow users to create files and folders, so Babble allows users to create conversation topics, and categories which contain them. Furthermore, users can re-name topics and categories, and restructure the resulting hierarchy, just as they do with files and folders. Thus, as a Babble deployment develops, its structure grows more complex as multiple users expand, modify and (rarely) delete elements.

For the purpose of this paper, we will look at the structures of 5 Babble deployments after periods of 6 to 10 months. The first row of table 1 shows the total amount of structure (the number of conversational topics and categories) created in each Babble (these counts exclude automatic archives

Babble Deployment (Months of use)	**B1** (10)	**B2** (6)	**B3** (6)	**B4** (6)	**B5** (6)
Total structure (Number of all topics + categories)	141	147	170	62	126
# Announcements % of total structure	1 1%	0 0%	2 1%	2 3%	3 2%
Project related structure % of total structure	14 10%	9 6%	19 11%	1 2%	53 42%
'Office' related structure % of total structure	90 64%	109 74%	46 27%	38 62%	12 26%
Structure used for public talk, and other purposes) % of total structure	36 26%	29 21%	103 61%	21 34%	58 46%

Table 1. Percent of structure devoted to non-conversational (rows 2-4) and conversational (row 5) activity in 5 Babbles

generated by the system, and structure deleted by users, the latter being quite rare). The point of interest here is that users have created quite a lot of structure, even though there are a relatively small number of regular, active users who contribute to conversations (typically 10 to 20, but around 30 for B1). Why is this happening?

To get a better understanding of what was going on, we classified the topics and categories created in each Babble. The classification was carried out by a single researcher, according to a general taxonomy based on our observations of various Babbles. We had observed that users created topics and categories in Babbles for at least four purposes:

- **To make public announcements**. Most often these are named "Announcements;" other examples are "Heads Up!," "News," and "Kittens Free to a Good Home!" Often these are positioned (or written in all caps) to attract attention.
- **To provide information about events and projects**. For example, one Babble uses project names as categories, and underneath the project name uses topics with names like "Current status", "Meet the project members", and "Tell us what you think!" In general, this approach of using some topics to contain static information, and designating particular topics as specifically for conversation, occurs across all Babble deployments.
- **To create office's or "personal places."** Offices are topics, or, most often, hierarchies of categories and topics, that 'belong to' and are named after a user. Offices are used in a manner similar to blogs; users typically provide information about themselves, may post essays or work in progress, and encourage 'visitors' to leave comments. A common form for an office is:

  Bob's Place
    About Me
    Talk with Me

  The first item is a category that contains two topics: a profile of Bob, and a place for leaving notes or chatting. Offices might contain other topics such as a "My schedule" or "Contact information."
- **As places for public conversation**. These categories and topics are intended for public conversation. They include the "Common's Area," the default topic in all Babbles, and other user-created topics, examples being "Bad Jokes," "IRL" (in Real Life), and "Knowledge Management." These categories and topics exemplify the way we envisioned Babble being used

Table 1 shows, for each Babble, the number and percentage of categories and topics that fall into each of these classes. The principle finding, for the purposes of this paper, is depicted in the last row of Table 1: the percentage of structure devoted to supporting public conversation ranges from 21 to 61 percent. (And this estimate is a maximum: this classification also includes topics and categories used for purposes other than the four listed above, although other non-conversational types of use are rare.)

The takeaway is rather paradoxical: in Babble, conversation topics are often not used for conversation; rather, a significant amount of user-created structure is devoted to 'publishing' non-conversational text. There are two inter-related issues: visibility and accessibility. With respect to visibility, users sometimes wish to make some non-conversational text prominent. The most obvious example is using "Announcements" as a topic name, which both signals its content and positions it at the top of the alphabetically sorted list. More generally, all five Babbles exhibit attempts to make some topics more prominent by using numbers or punctuation characters as prefixes to control their sorting order. Even when high visibility is not required, users often wish to create non-conversational text that is easily accessible – that is, visible as soon as another user enters the topic, rather than being 'submerged' in a stream of conversation. To achieve these ends users employ a number of naming tactics to deter others from adding comments to a topic[1]: occasionally the topic name explicitly indicates others should not talk (e.g., "About Me (read only)," but more often users juxtapose topics (as in the "Bob's Place" example), where one is clearly marked to indicate that it is intended for conversation. Other examples include "My Schedule" / "Chit Chat," "Interview with Bob" / "Audience Questions," and "<Project Name> Status" / "Discuss <Project Name> here." To summarize, throughout the deployments users are doing more than creating places to talk: they are trying to 'publish' non-conversational text, and using a variety of naming tactics to ensure that their text has the visibility and accessibility it merits.

[1] An alternate way of avoiding this sort of problem is to provide means for users to control write (etc.) access to their topics. Babble and Loops have avoided this approach, because one of the goals of the underlying research program is to explore the extent to which social mechanisms – for example, the development of norms amongst mutually visible and known participants – can eliminate the need for rigid technical mechanisms. See [1, 5, 7].

Summary of Design Rationale

The design of Loops was shaped by a confluence of factors. We felt that Babble's mechanisms for supporting conversation worked well, and wished to retain them. At the same time, our experiences in deploying Babble and observing its use over the long term, provided new design requirements that shaped the design of the Loops system in several ways. First, the requirement for supporting *easy deployment and updating* pushed us in the direction of creating a web-based application. Keeping the application code entirely on a server, eliminated the problems of requiring users to download and install new versions of the application, and of keeping users in sync. Second, the requirement for an *easily changeable look and feel*, in tandem with the decision to go with a web-based application, led us to implement the Loops client in Macromedia's Flash®, an environment that allowed us to create sophisticated interactive animations that can play in browsers. Finally, the third and fourth requirements – *support membership in multiple communities* and *support 'publishing' of non-conversational text* – were addressed in the user interface, which we turn to next.

THE LOOPS SYSTEM

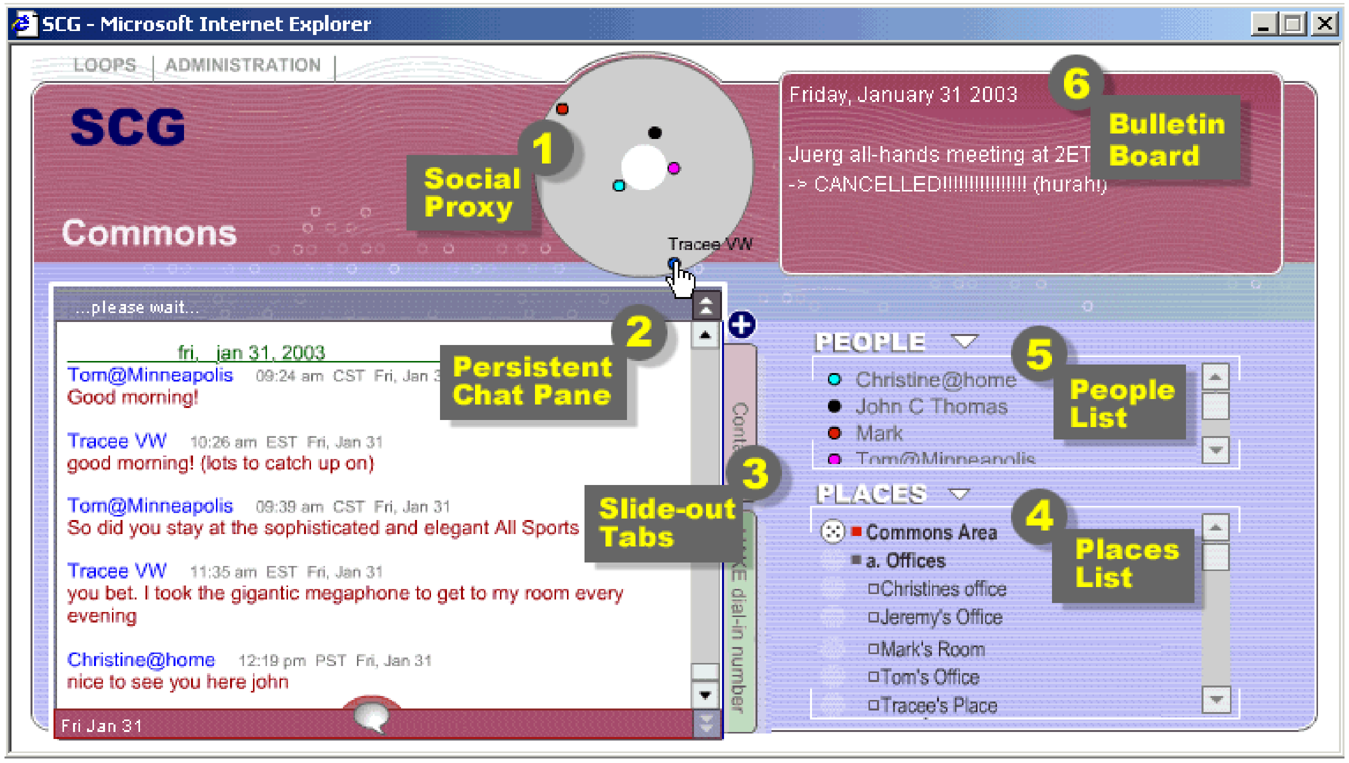
Loops consists of a set of user-definable places, each of which can contain a conversation, URLs, visible non-conversational text, and people, as well as user interface elements for seeing who is present, viewing, navigating and modifying the environment. The user experience is that people log in to a Loops server and move from place to place, reading conversations that have changed in their absence, contributing new comments, and encountering other users as they do so. As with Babble, the ultimate goal is that Loops feel like an inhabited place where users may 'hang out' during the day as they work on their computers, or into which they may occasionally venture to see what has happened in their absence.

An Overview of the User Interface

The user interface elements of Loops are shown in figure 1:

1. The **social proxy** depicts people as dots, showing who and how many are in the place and their activity levels.
2. The **chat pane** is where those in the place 'talk.'
3. Each place can have **slide-out tabs** that can contain publicly viewable and editable text and URLs.
4. The **places list** shows the places, indicates which have new content, and provides a menu of place commands.
5. The **people list** shows those logged in, and provides a menu of person-oriented commands.
6. Each place has a **bulletin board** that is viewable and editable by all those in the place.

More holistically, figure 1 shows the "SCG" Loop "Commons." The social proxy (1) shows that there are five people in the Commons place, three of whom are actively



Friday, January 31 2003

Juerg all-hands meeting at 2ET -> CANCELLED!!!!!!!!!!!!!!!! (hurah!)

...please wait...

fri, jan 31, 2003

Tom@Minneapolis 09:24 am CST Fri, Jan 3
Good morning!

Tracee VW 10:26 am EST Fri, Jan 31
good morning! (lots to catch up on)

Tom@Minneapolis 09:39 am CST Fri, Jan 31
So did you stay at the sophisticated and elegant All Sports

Tracee VW 11:35 am EST Fri, Jan 31
you bet. I took the gigantic megaphone to get to my room every evening

Christine@home 12:19 pm PST Fri, Jan 31
nice to see you here john

Fri Jan 31

Christine@home
John C Thomas
Mark
Tom@Minneapolis

Commons Area
a. Offices
Christines office
Jeremy's Office
Mark's Room
Tom's Office
Tracee's Place

Figure 1. The Loops user interface, including 1) a visualization of the presence and activity of participants, 2) a chat area the supports synchronous or asynchronous conversation, 3) public slide out tabs that can hold editable text and URLs; 4) a list of places, 5) a list of people who are present, and 6) a public bulletin board the can contain editable text and URLs. NB: The image has been edited to remove about a third of its height; the gray circles and rectangles are callouts and not part of the interface.

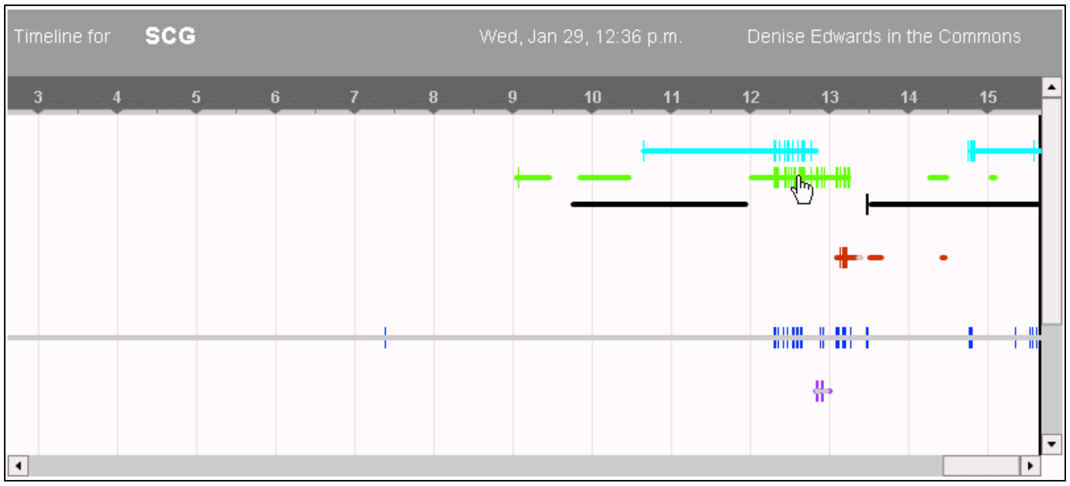


Figure 2. The Loops timeline proxy shows the last week's worth of activity, with each user represented in a single row; mousing over traces shows the name and location of that user at that time

talking in the chat pane (2). From the point of view of the user whose screen we are seeing, there is no new content elsewhere in the Loop – otherwise there would be red indicators next to other places in the places list (4). The two tabs (3) contain a list of telephone numbers for the Loop's members, and a dial-in number and access code for the group's weekly conference call. The bulletin board (6) contains a reminder of an upcoming meeting, with text stating that it has been cancelled, and a subsequent reaction.

Now we will take a somewhat more in-depth look at functionality.

Awareness and Conversation

Because the awareness and conversation models of Loops are derived from the Babble system, and are not the focus of this paper, we'll keep our remarks brief.

The chief awareness interface element is the social proxy (callout 1, figure 1). The circle represents the place being viewed; the colored dots represent people. A dot shown inside the circle means that that user is in the current place; when users are active (meaning that either they type or click) their dots move to the inner (white) core of the circle (as with the dots at 1, 3 and 8 o'clock), and then, over the course of 15 minutes, they drift to the edge of the circle (as with the dots at 5 and 10 o'clock). Mousing over a dot reveals the name of the user, and mousing down on a dot brings up a menu of commands for either changing one's preferences (if it's one's own dot), or for interacting with other users (if it's another's dot).

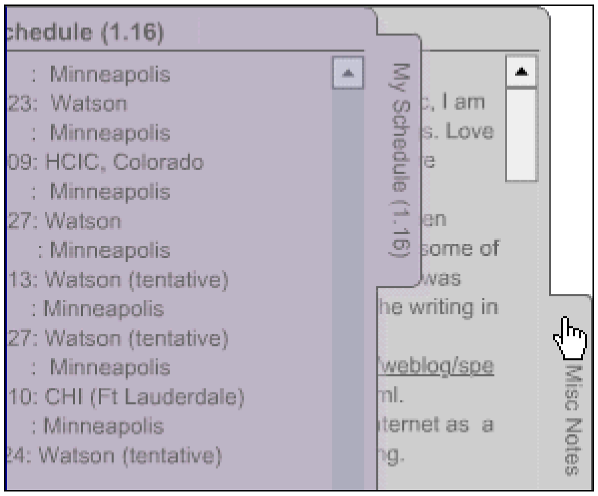
Loops also contains a timeline, a social proxy (figure 2) that shows who has been present over the last week, and how often they have spoken. In this proxy, each user is represented in a row: they leave a flat line if they are present, and they make a blip when they speak. Thus, the timeline in figure 2 shows six people, all of whom have spoken between 12:00 and 14:00. Mousing over the lines, as with the other proxy, reveals information about the speaker and time and place of speaking, and mousing down brings up a command menu. This proxy opens in a separate window; it appears to be most frequently used by those who are (formally or informally) in the role of running the community.

The persistent chat pane (callout 2 of figure 1) displays a conversation as a time-stamped list of comments in a single window, enabling either synchronous or asynchronous talk. Comments are added by clicking on the "speech bubble" button at the bottom of the chat pane, or simply by beginning to type; this brings up a floating window in which the comment may be composed. The use of a floating composition window – unlike that provided by many synchronous chat clients – is to enable those writing comments to move from place to place while composing a comment, thus making it easier to compose synthetic or integrative comments. Once a user posts a comment, it immediately appears in the conversation. For users who are in other places, the name of the place turns red to indicate the new content, and when they enter the place, chat text that is new since their last visit is shown in red.

Bulletin Boards and Tabs for 'Publishing' Text

Loops tabs and bulletin boards are the design response to the requirement to provide a means of publishing non-conversational text.

Bulletin boards (figure 1, callout 6) provide a means for posting text and URLs in a highly visible place. Each place has its own bulletin board, and its text may be edited by anyone in the place. When new or changed text is posted to a bulletin board, the new text is signaled to those in the place by the background color of the bulletin board fading out and then fading back in with the new text displayed. If there is more text than fits in the visible area of the bulletin board, a scroll bar appears. We anticipated that bulletin boards would be used for purposes ranging from announcements and reminders (as seen in figure 1), to MUD-like scene setting (e.g., "You see a messy office."), based on our observations of Babble usage.



: Minneapolis
23: Watson
: Minneapolis
09: HCIC, Colorado
: Minneapolis
27: Watson
: Minneapolis
13: Watson (tentative)
: Minneapolis
27: Watson (tentative)
: Minneapolis
10: CHI (Ft Lauderdale)
: Minneapolis
24: Watson (tentative)

c, I am
s. Love
e
en
some of
was
he writing in
weblog/spe
ml.
ternet as a
g.

Figure 3 [cropped]. Two tabs: one fully opened, the second sliding out over the first.

Likewise, tabs (figure 3) provide a place for non-conversational text that needs to be accessible, but doesn't need to be made as obvious. The tabs peek out from behind the conversation pane. Clicking on the tab causes it to slide out, revealing the (editable) text on it. Each place can contain up to three tabs; places begin without tabs, and users can create them by pressing the "+" button (above the top tab in figure 1). The lower part of each tab (not shown) provides access to controls for setting its background color, clearing its content, and deleting the tab. We expected that tabs would be used for activities such as sharing schedules (as in figure 3), lists of URLs, and keeping to do lists.

The Loops Launcher

Finally, figure 4 shows the Loops launching screen, the design response to the requirement for supporting membership in multiple communities. It provides a single location where users of multiple Loops can sign on once and access all Loops of which they are members. It also provides a place where Loops users can create their own Loop (via the "Administration" menu). The new Loop is automatically set up, and the creator can designate who has access to it (provided they are a member of another Loop; otherwise a system administrator has to add them to the Loops server). We had hoped to have each Loop's icon reflect its degree of

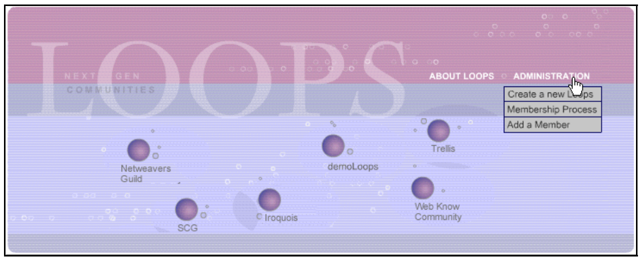


Figure 4 [cropped]. The Loops Launcher provides a view of all Loops communities hosted by a server; it provides single sign-on access to users of multiple communities.

activity, but this was not possible during our implementation time frame.

THE DEVELOPMENT AND EVALUATION OF LOOPS

Loops went through a period of iterative development and evaluation. Early versions were prototyped and tested by the development team using well known methods (e.g., [24]). However, at some point feedback is needed from a broader constituency. As Herbsleb et al. [11] have noted, a dilemma plagues groupware developers: how do developers gather user input for applications whose user experience is fundamentally a collective one, when the preliminary nature of the software is such that it is likely to deter collective adoption? Our response to this dilemma was to run user tests in which our 'users' were existing groups. We decided that we would invite pre-existing groups, with experience interacting online, to use our system for a limited time trial that we termed a "test drive."

The Test Drives

We identified and recruited two groups for our test drive. One, Netweavers, was an existing Babble community with a couple of dozen core members; the other, Trellis, was a small group of four, two of whom had a well established mechanism for remote collaboration involving the use of instant messaging and the telephone. Because the Netweavers' organizer was concerned about disrupting the community, we agreed to a limited time trial of four days.

Our primary method of gathering information during the test drives was to observe and participate, noting confusions, questions, comments, and signs of emerging practices. Since many people were typically present at the same time, and since they had been explicitly asked to provide feedback, critiques often took on a dialectic character. Sometimes agreement about problems emerged quickly; at other times disagreements arose and led to discussions revealed differences in assumptions, values, etc.

We also analyzed log files and the Loops' conversations to obtain a more quantitative picture of what occurred. Over the 4 days of the test drive, 26 people accessed the Netweavers Loop, created their own accounts, and spent time there, trying out features, providing feedback, and engaging in the combination of banter and wide-ranging discussion that characterizes online activity in the Netweavers' Babble. The Trellis test drive was more open-ended: the results reported here come from about two weeks of use, almost entirely from the 2 most experienced collaborators, though all 4 members logged in at least once.

In general, both groups made extensive use of Loops. Between them, the Netweavers and Trellis groups produced approximately 42,000 words (or 3,300 and 5,000 words per day of use, respectively). Individuals varied considerably in their usage patterns, but the median user logged on to Loops 3 times, and spent about 3 hours on line. We took the number of users, frequency of use (including return visits), and amount of content produced as a sign that the system was basically usable.

To get a clearer picture of users' preferences and priorities, we printed out transcripts of all discussions (about 42,000 words of text), and did a rough content analysis to identify problems, controversies and suggestions. This generated a list of 126 comments (82 from Netweavers and 44 from Trellis, with some overlap). From this we developed a structured survey that could be completed in no more than 10 minutes. The main portion of the survey made four to five statements about each UI element, and used a 7-point scale to quantify agreement; the survey concluded with open ended questions, including queries about which interface elements merited the most screen space. The survey was emailed to the 30 participants shortly after the end of the Test Drive; 22 completed the survey.

The results confirmed our impression from the test drive that Loops was basically usable. One question (directed to the 17 Babble users who participated) showed a preference for Loops over Babble (14 agreeing, 2 neutral, 1 disagreeing), provided its performance problems and obvious bugs were addressed. The test drive also provided information about details of the design. Among the things we discovered were that people wanted a wider chat pane, smaller bulletin boards, better performance, page-at-a-time scrolling, and changes to other details of the interaction design. These were subsequently implemented (NB the user interface shown in the previous section reflects these and other changes based on the test drive results).

With respect to the non-conversational text publishing, the test drive provided us with useful information about tabs and bulletin boards. The survey showed positive responses to both tabs and bulletin boards: 50% agreed that tabs were useful (9% disagreeing and 27% neutral), and 68% agreed that bulletin boards were useful (9% disagreeing and 14% neutral). While it is wise to be cautious about users' positive reactions to new interface features before they have had time to live with them, the fact that most of the test drive participants were experienced Babble users left us cautiously optimistic. Going into the test drive we were also concerned that participants might think that the tabs and bulletin boards were private spaces, rather than areas that all users could read – however 68% and 64%, respectively, reported no initial confusion.

One other aspect of the survey – which is, to our knowledge, novel – is that we also administered it to the development team. We decided it would be interesting to take the survey ourselves, answering not with our opinions, but with our intuitions about what users would say. Our self-survey was the same as that administered to the users, except that the standard 7-point scale was extended to provide two other ratings: "all over the map", for when we thought users would have a variety of opinions; and "no idea", for when we didn't think we knew what users would say (although the "no idea" idea rating was used very rarely!). The results were that the team's intuitions were correct for 7 of 25 questions and incorrect for 11 of the 25 questions; for the remaining 7 questions, the team itself did not agree on how users would respond. This addition to the survey process provides a nice indication of its value as a design tool, and helps counter the *post hoc* tendency to believe that the user study results were 'obvious.'

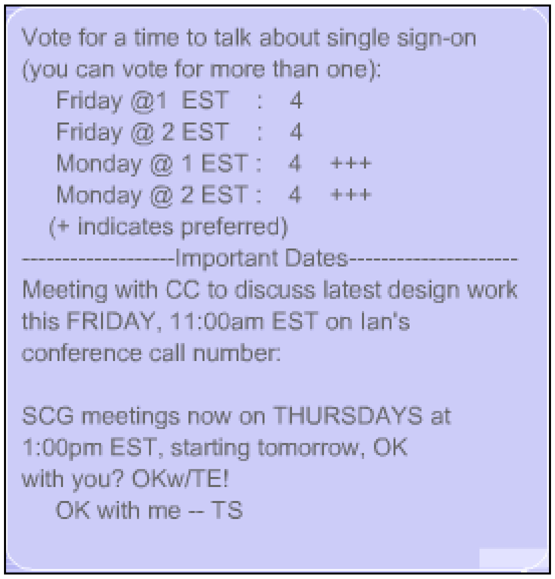
DEPLOYING LOOPS

Although the test drive was helpful in fixing usability problems, the short term nature of the test drive doesn't allow us to establish whether or how the new non-conversational features of Loops are useful to work groups. Thus, we moved on to deploy the system and observed its use under more realistic conditions. We begin by describing some of the ways in which Loops users have been observed to use the tabs and bulletin boards. Then we turn to the question of the overall success of Loops deployments, and provide a profile of one of the most successful Loops.

Usage of Tabs and Bulletin Boards

In general, tabs and bulletin boards have been used much in the way we envisioned.

Bulletin boards are typically used for announcements; figure 5 shows three examples. The second example (under "Important Dates") is a typical one: it states the time and call in number for a recurring phone meeting. The first and third examples are more interesting. The first shows the bulletin board being used to arrange meetings. Here, one person has proposed a set of possible times for a meeting. Initially the organizer put a "1" next to each time, indicating that she could make each; others came along and incremented the numbers as was appropriate. Later, another user added a plus sign as away of indicating that a time was preferred. The


```
Vote for a time to talk about single sign-on
(you can vote for more than one):
    Friday @1  EST    :   4
    Friday @ 2 EST    :   4
    Monday @ 1 EST :   4    +++
    Monday @ 2 EST :   4    +++
   (+ indicates preferred)
-------------------Important Dates--------------------
Meeting with CC to discuss latest design work
this FRIDAY, 11:00am EST on Ian's
conference call number:

SCG meetings now on THURSDAYS at
1:00pm EST, starting tomorrow, OK
with you? OKw/TE!
    OK with me -- TS
```

Figure 5. Two forms of bulletin board usage on one bulletin board: voting for a meeting time (top and bottom), and announcements (middle).

third example shows a similar case, except here participants are initialing the announcement to indicate agreement. Note that there is no way to identify who has written what on the bulletin board, and thus this type of use requires everyone to trust that their colleagues will not 'cheat' by casting multiple 'votes' or forging initials. While we've observed other more ludic uses of bulletin boards – drawing character graphics pictures, playing tic-tac-toe (very awkwardly) – most uses are for announcing meetings and reminding of deadlines.

Tabs function similarly to bulletin boards, though they are not used for announcements or scheduling. Generally they are used for lists (of phone numbers, emails, URLs), schedules, and (occasionally) for rough notes. Occasionally attempts have been made to use them as a collaborative editing tool: In one case, tabs were used to compose a piece of text, with the chat pane being for question, answers and comments. However, as tabs do not provide an edit lock, support any sort of rich text, and indeed provide only a narrow writing area, this has not proved to be a viable use.

While these uses of tabs and bulletin boards are nothing out of the ordinary, they provide a useful boost in functionality, particularly in tandem with the other features of Loops. For example, one might find a Loops place devoted to a particular project, where the bulletin board is used to announce the next meeting time, a tab holds the number and passcode for the conference call, and the chat pane is used to take notes as the meeting occurs.

Deployments

Loops has been deployed to about six groups.[2] The success of our deployments has been mixed, although it is a bit difficult to specify what counts as success. There are at least three possible definitions of success: that the system is sufficiently functional that the group is able to use it to interact; that the system enables the group to achieve one or more goals; or that the system, once taken up, becomes part of the group's practice and is used for as long as the group exists. Each of these definitions has problems. If the system is usable but doesn't meet the needs of the group, it is a rather weak definition of success. If the system enables the group to achieve one or more goals, success depends on how ambitious the goal is – supporting a two hour brainstorming session is easier to achieve than providing a permanent online group meeting space; it is also the case that the members of a group may have multiple, and even differing, goals. Finally, if we define success as permanent adoption by the group, we rule out legitimate uses for limited duration activities; we also have the difficulty of deciding when to declare adoption permanent, and how long to wait until declaring that a deployment has failed (which, as we shall see, is a non-trivial decision).

For the purposes of this paper, we will consider the first and last definitions. For the first, sustained usage, we will count a deployment as successful if it has continued activity for eight weeks or longer. This admittedly arbitrary metric is based on our experience with Babble deployments, where we found that most Babbles would experience usage activity for the first several weeks, and that at about the six week mark we would see either a drop off in activity leading to the demise of the deployment, or a continuation of activity for a much longer period of time [1]. In terms of this metric, five of the six Loops we've fully deployed have been successful. In terms of the last definition of permanent use, three of the six are successful.

Is this good or bad? It's not clear. Rather surprisingly, we know of no studies that report adoption rates for groupware applications (by any measure of success). We do know that adoption of even proven applications is a non trivial process affected by variables ranging from individual factors (e.g., 17] to social and organizational factors [16]. Certainly, in our own experience it is not uncommon for attempts to make use of shared databases (within our organization) or mailing lists (outside of our organization), to begin with a burst of activity only to quickly subside into non-use. Clearly, more investigation is called for.

A Close Look at a Successful Deployment

In this section we take a close look at a successful deployment to a group we will call Fargo. Fargo is interesting for two reasons: first, although successful, it's usage patterns deviated from our expectations; and second, it provides a good example of a sophisticated use of tabs and bulletin boards to do project management tasks that would have been difficult to carry out in Babble.

Fargo is a team of about 28 people distributed over 5 sites: New York; North Carolina; Japan, India and Zurich. The team is involved in a software development project, and includes managers, programmers and testers. Fargo does major code releases every six months, and incremental build releases every one to three months.

We used a combination of methods to study Fargo. First, we surveyed the Fargo team before Loops was deployed to understand their relationships with each other, and their knowledge of other team members. Second, we analyzed the conversation in Loops, and (drawing from log files) the various contributions and edits to the tabs and bulletin board, as well as more general activity (logins, idle times, etc.) Third, we examined the contents of two databases the Fargo used to manage its project. Finally, we conducted semi-

[2] It is not always clear what to count as a deployment. Anyone who is a member of any Loop has the ability to create a new Loop from the Loops Launcher page; thus, it is possible to quickly generate a Loop for a person or group interested in a demo, although they have no intention of using it for a long period. Here we use the term deployment for cases in which we went through a dialog with the prospective users, identified a facilitator, and made sure that the facilitator circulated a welcome message with usage instructions and advice on running a community.

structured interviews with three Fargo members who happened to be in our geographic area. (A more detailed report of Fargo's use of Loops may be found in [10].)

Fargo is an interesting case to look at because it is an example of a successful Loop that, by our first definition, had failed. This can be seen in figure 6, which shows a ten month segment of Fargo's posting patterns. What we see is that at the end of its first two months, Fargo's posts had dropped to nearly zero, and continued at a low level for the next two months. At the end of the fourth month we had concluded that Fargo had died, and were therefore quite surprised, a few weeks later, to receive an urgent call from the Fargo facilitator during a server outage.

As it turned out, the Fargo team was using Loops quite vigorously, but only during the weeks when they were approaching a code release and needed to communicate as quickly and widely as possible. At other times, the members of Fargo, especially the programmers, abandoned Loops and used more asynchronous means of communication. And at still other times, Loops was used primarily for reading rather than talking. For a fuller account, see [10].

Fargo is also interesting because of the team's use of tabs and bulletin boards. For example, the Fargo Commons room bulletin board was heavily used during their first build cycle. Over a one week period (during which we were monitoring its use) the bulletin board was used for announcements of build dates and "burning issues," with updates 2 to 3 times a day (all but two of these posts were made by Fargo's manager). During this period, the tabs were not used for any substantive purpose. However, in the next build cycle, the group developed a more sophisticated use of tabs and bulletin boards. As in the first phase, the Fargo manager used the bulletin board to keep track of the build dates. However the other information relative to the build was moved to the tabs. This included the burning issues (now titled "key issues", and soon to be renamed "action items"), and also lists of known problems, and information for the documentation. It is also interesting to note that during the lulls shown in Figure 6, activity did not entirely stop: people continued to log in, and in fact there was a significant amount of activity involving opening (and presumably reading) tabs. It turns out that as the project approached completion, new members – responsible for product management tasks such as

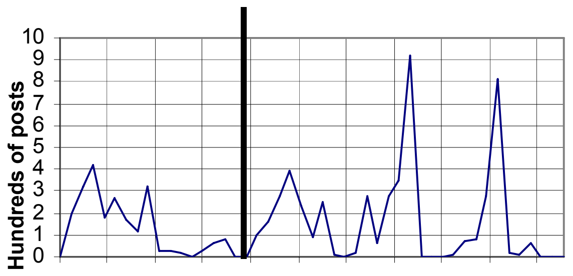


Figure 6. Posts per week (in hundreds of posts) in Fargo over the first 10 months

documentation – were joining the Loop, and accessing the tabs to learn about the state of the project.

CONCLUSIONS

In this paper we describe the design of Loops, a second-generation web-based conversation environment designed for corporate work groups. Beginning with the design rationale derived from our experience with the first generation system, we describe the resulting system and the ways in which it addressed the rationale. We take a number of lessons away from our work on Loops. First, we believe that effort to create niches for 'publishing' non-conversational text in our conversation environment is a clear success. All deployments have made some sort of use of these textual niches. We are also struck by the degree to which simple editable text, and the ability to have conversations about its use, can support sophisticated use, such as the group scheduling example. Of course, this sort of use is dependent on agreement (and trust) amongst those so using it, but the fact that this can substitute for technical mechanisms (e.g., user authentication, controls to keep one person from voting more than once) is interesting. Looking at the Fargo example, we note that the ability to place text in a prominent place, where it is not buried in the conversational stream, is useful not only for those who are currently using the system, but for those who come by later.

More generally, our experience with Loops deployments suggests that even amidst the proliferation of computer mediated communication tools in corporate environments – email, instant messaging, and now blogs and wikis – there is still an important role for group conversational environments. However, our experience with Fargo suggests that our initial focus on designing online environments as places for community may have led us somewhat astray. While providing a permanent online space for a group is certainly a valuable end, it is becoming increasingly clear that this is not the only usage model. Fargo uses its Loop as a war room; it moves in for one phase of its development cycle, and then abandons it for other communication channels (which often means little communication among the more disparate parts of the team). We can point to other uses of Loops and Babble that have similar characteristics, although there Loops functions more as a one time meeting place for ad hoc groups.

If we relax the notion of Loops being a space for an online community, a place where people hang out and to which they return day after day, it suggests a number of directions for future work. First, it should be as easy to create and enter a Loop as it is to grab an unoccupied meeting room. While we have made some strides in this direction, we need a lighter weight way of adding new members to the Loops environment. Second, it should be easy to bring material into a Loop, work with it there, and take it away afterwards. As Loops is now, cut and paste is the primary import and export mechanism; this does not seem adequate. Third, Loops' simple membership model – you're either in the community or you're not – needs to become considerably more sophisticated. If a Loop becomes more like a meeting room, there is a greater need for roles – and their accompanying privileges and responsibilities – than there is in a tight knit community. Finally, to the extent that Loops is to function as an occasionally occupied space, as is the case with Fargo, it needs mechanisms for alerting participants when activity resumes after a lull.

ACKNOWLEDGEMENTS

Thanks to Cal Swart who maintains our servers, anonymous reviewers of a previous version of this paper, and our users.

REFERENCES

1. Bradner, E., Kellogg, W, & Erickson, T. Bradner, E., Kellogg, W, & Erickson, T. The adoption and use of Babble: A field study of chat in the workplace. *Proc. ECSCW 1999.* Kluwer, 1999, 139-158.
2. Bruckman, A. & Resnick, M. The MEDIAMOO project: Constructionism and professional community. *Convergence*, 1(1) 1995.
3. Cherny, L. *Conversation and community: Chat in a virtual world.* CSLI Publications, 1999.
4. Churchill, E. F. and Bly, S. Virtual environments at work: Ongoing use of MUDs in the workplace. *Proceedings of the International Joint Conference on Work Activities Coordination and Collaboration*, 1999, 99-108.
5. Erickson, T. and Kellogg, W.A. Social Translucence: An approach to designing systems that mesh with social processes. *ACM Transactions on Computer-Human Interaction*. **7(1)**, 2000, 59-83.
6. Erickson, T. and Kellogg, W.A. Knowledge communities: Online environments for supporting knowledge management and its social context. *Sharing expertise: Beyond knowledge management* (eds. Ackerman, Pipek, Wulf). MIT Press, 2003, 299-325.
7. Erickson, T. Smith, D. N., Kellogg, W. A., Laff, M. R., Richards, J. T., and Bradner, E. Socially translucent systems: Social proxies, persistent conversation, and the design of 'Babble.' *Proc. CHI 1999*. ACM Press, 1999, 72-79.
8. Gutwin, C., Greenberg, S., and Roseman, M. Workspace awareness support with radar views. *Ext. Abstracts CHI 1996*, ACM Press, 1996, 210-211
9. Guzdial, M. Information ecology of collaborations in educational settings: Influence of tool. *Proc. CSCL 1997, 83-90.* Toronto, Ontario, Canada.
10. Halverson, C., Erickson, T. and Sussman, J. What counts as success? Punctuated patterns of use in a persistent chat environment. *Proc. GROUP 2003.* ACM Press, 2003.
11. Herbsleb, J.D., Atkins, D.L., Boyer, D.G., Handel, M. and Finholt, T.A. Introducing instant messaging and chat in the workplace. *Proc. CHI 2002.* ACM Press, 2002, 171-178.
12. Hiltz, S.R. and Turoff, M. *The network nation: Human communication via computer.* Revised edition, MIT Press, 1993.
13. Houde, S. and Sellman, R. In search of design principles for programming environments. *Proc. CHI 1994*, ACM Press, 1994, 424-430.
14. Issacs, E., Walendowski, A., Whittaker, S., Schiano, D. and Kamm, C. The character, functions, and styles of instant messaging in the workplace. *Proc. CSCW 2002.* ACM Press, 2002, 11-20.
15. Nardi, B., Whittaker, S., and Bradner, E. Interaction and outeraction: Instant messaging in action. *Proc. CSCW* 2000. ACM Press, 2000, 98-88.
16. Orlikowski, W. Learning from Notes: Organizational issues in groupware implementation. *Proc. CSCW '92*: ACM Press, 1992, 362-369.
17. Orlikowski, W.J., Yates, J., Okamura, K., Fujimoto, M. Shaping electronic communication: The meta-structuring of technology. *Organization Science*, **6(4)**, July-August 1995
18. Rheingold, H. *The virtual community*. Addison Wesley, 1993.
19. Roseman, M. and Greenberg, S. TeamRooms: Network places for collaboration. *Proc. CSCW 1996*, ACM Press, 1996, 325-333.
20. Scardamalia, M., Bereiter, C., McLean, R.S. Swallow, J. & Woodruff, E. Computer-supported intentional learning environments. *Journal of Educational Computing Research*, **5(1)**, 1989, 51-68.
21. Schlager, M., Fusco, J., & Schank, P. Cornerstones for an on-line community of education professionals. *IEEE Technology and Society Magazine*, **17(4)**, 1998, 15-21.
22. Viegas, F.B. and Donath, J. Chat circles. *Proc. CHI 1999,* ACM Press (1999), 9-16.
23. Werry, C.C. Linguistic and interactional features of Internet Relay Chat. In S. Herring (Ed.), *Computer-mediated communication: Linguistic, social and cross-cultural perspectives*, 1996, John Benjamins, 47-63.
24. Wolf, T.V., Rode, J.A. and Kellogg, W.A. Dispelling design as the 'black art' of CHI. *Proc. CHI 2006*, ACM Press, 2006, 521-530.