The Design and Deployment of Loops: A Web-Based Conversation Space for Work Groups

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

ABSTRACT
We describe the design of Loops, a second-generation CMC 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. Next we discuss how an early working version of the system was 'group tested,' and the changes that lead to. Then we describe its realization in an implemented system, discuss its deployment within our organization and provide some examples of how Loops is 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. Both of these tactics were initially explored in the context the "Babble" system [7].

In this paper we describe the design and deployment of Loops, the web-based successor to Babble. We begin by describing how our experiences with Babble shaped its features. We go on to describe the resulting system, and discuss how we approached the challenge of user testing a system for groups. Next we turn to our experiences in deploying Loops, describing our mixed record of success, and some of the usage patterns observed in successful deployments. Finally we reflect on the implications of our experience for the design of similar systems.

DESIGN RATIONALE

The design of Loops was shaped by our experiences with the Babble system. While Babble's conversation model and social proxy seemed quite successful, there were a number of recurring problems that we wished to address in the next generation system. We will treat each in turn, but to begin we need to provide a little background about Babble.

About Babble

In terms of its functionality, Babble [7] resembles a multi-room chat system, with three differences. First, the conversation in Babble persists across sessions and may be synchronous or asynchronous: that is, remarks may be separated by seconds, minutes, days or months. Second, the structure of Babble's conversation space is user-definable: anyone can create, modify or reorganize rooms. Third, Babble uses visual cues to enhance its users' mutual awareness of one another, and of the presence and location of new information in the environment.

Over the five years of its project's life, Babble has been deployed to about two dozen groups, mostly within the large corporation in which we are based. 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]).

Four Requirements

As we gained experience with Babble, we noticed a number of recurring problems that we used to define four requirements for the next generation system.

Supporting Deployment and Updating

In our five years of work with Babble, we repeatedly struggled with difficulties in deploying and updating it. The Babble client, written in Smalltalk, was 2 to 3 megabytes in size; to do an installation or an update, the users 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 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 time we released a major update, we disrupted the communities of our users 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 aspects of the look and feel of the user interface and to allow our visual designer to directly work in the medium rather than having design prototypes 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 place considerable value on aesthetics, and as Babble had always maintained a stubborn resemblance to a Smalltalk browser, we wanted our next system to provide as much support for iteration in the visual design as it did in the functional design.

Support for 'Publishing' Text

Another problem we observed, 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. Babble users often wanted to create text that would be visible to others for long periods of time. Examples of this were that users often created profiles of themselves (in large Babble communities), descriptions of their projects, personal resources (e.g. lists of URLs), or had announcements that they wanted to make visible to everyone. The problem was that the only mechanism Babble provided for creating text was as a comment in a conversation, and so any text entered was liable to being followed with commnets, and often ended up being lost in a stream of conversation. While users developed a number of conventions for dealing with this problem (e.g. creating parallel conversations, one for talking and one for publishing), it was clear that a way of publishing text outside of conversations would be a welcome addition.

Support Membership in Multiple Communities

Initially, we had envisioned that Babble would be used as an online environment for distributed work groups. But, as time went on, Babble was requested and used by other types of groups: 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 had multiple uses for it, or, to put it another way, people wanted multiple Babbles for use with multiple groups. Babble provided no support for this situation, and so we resolved to address it in the new system.

The Foundations of the Loops System

The requirements described here shaped the design of the Loops system in three ways. First, the requirement for supporting easy deployment and updating pushed us in the direction of creating a web-based application. Obviously, 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 browers. Finally, the last two requirements were addressed in the user interface, which we turn to later.

RELATED WORK

While Babble and Loops are not quite like any other collaborative environment, they blend features from a variety of collaborative environments. In terms of look and feel they resemble multi-channel chat systems (e.g. [W96]), with their transcripts of (potentially) real time conversation and their lightweight conversation model. They also are akin to the instant messaging applications that are becoming widely used in corporate work places [15, 11, 14], particularly in their ability to support both lightweight coordinating talk and in-depth work conversation. And they have similarities to MUDs and MOOs (e.g. [2, 3, 21, 4]) in that they provide a user-extendable set of 'rooms' that persist over time, and through which users move and support both opportunistic interactions 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 include CSILE [20], CaMILE [9] and TeamRooms [19], and is most recently manifesting itself in web boards and blogs. That text-based conversation is a rich source of social information has been well-documented, especially by Cherney [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 systems that support workspace awareness (e.g. [8], and to work on visualizing chat users [22].

THE DEVELOPMENT AND EVALUATION OF LOOPS

As might be expected, Loops went through a period iterative development and evaluation. Initially, early versions were tested among ourselves. This has obvious limitations, and we clearly needed other forms of evaluation. However, as Herbsleb et al. [H02] have noted, a dilemma plagues groupware developers: how do developers go about getting 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 exisiting 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' principal organizer was concerned about fragmenting the part of the community that used Babble, we agreed to do a limited time trial of four days.

Over the four 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 two experienced collaborators, though all four 'members' of Trellis logged in at least once.

In general, both groups made extensive use of Loops. Between them, the Netweavers and Trellis Loop users 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.

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.

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 analysis to identify problems, controversies and suggestions. 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 interval 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.

Space does not permit a detailed description of the results. Over all, the survey (as well as the sheer volume of use during the test drive) indicated that Loops was basically usable: one question (directed only to regular Babble users) showed a preference for Loops (14 agreeing, 2 neutral, 1 disagreeing) provided its performance problems and obvious bugs were addressed (something accomplished in a subsequent move to a new version of Flash). Other probes indicated positive responses to the new UI features for supporting static text. The test drive also provided anformation about small details of the design. Among the things we found out was that people wanted a wider chat pane, smaller or concealable bulletin boards, page-at-a-time scrolling, all of these were implemented.

One other aspect of the survey—which is, to our knowledge, novel—is that we also administered it to our own group. 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. In addition to the standard scale, our self-survey included 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 turned out to be used very rarely!). We did a rough analysis, noting which category the clear majority of the users' responses fell into for each question (or if there was not a clear majority, coding it as "all over the map"), and did the same for the our team's responses. 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 survey results were 'obvious' from the beginning.

THE LOOPS SYSTEM

Loops consists of a set of user-definable rooms, each of which can contain a conversation, URLs, static 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 room to room, 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

We'll begin with an overview of the user interface elements of Loops shown in figure 1:

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

Or, more holistically, figure 1 shows the "Commons" room of the "SCG" Loop. We can see from the social proxy (1), that there are five people in the Commons room, three of whom are active, carrying on the chat shown 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 rooms in the places list (4). The two tabs (3) contain a list of contact numbers for the Loop's members, and a dial-in number and access code for conference calling. The bulletin board (6) contains a reminder of an upcoming meeting, with added 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 room being viewed; the colored dots represent people. A dot shown inside the circle means that that user is in the current room; when users are active (meaning that either they type or click) their dots move to the inner (white) core of the circle

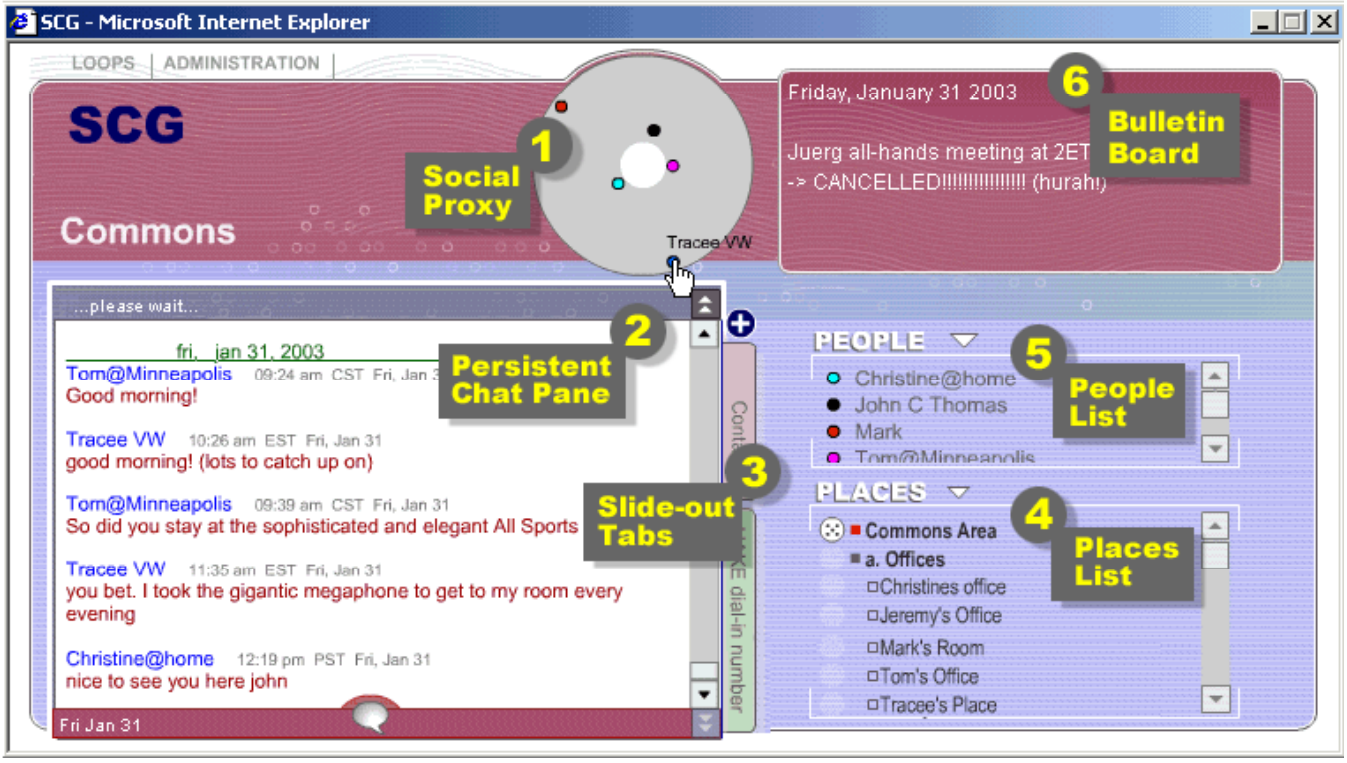

Figure 1. The Loops user interface, including 1) a visualization of the presence and activity of participants, 2) a chat area the supports synchronous or asynchronous conversation, 3) public slide out tabs that can hold editable text and URLs; 4) a list of rooms, 5) a list of people who are present, 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.

(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 social proxy (figure x) that shows who has been present over the last week, and how often they have spoken. In this proxy, each user has a row, they leave a flat line if they are present, and they make a blip when they speak. Thus, the timeline shown in figure 2 shows six people, all of whom have spoken between 12 and 14 hundred hours. 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.

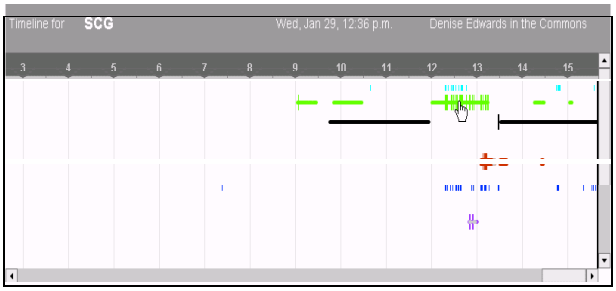

Figure 2. The Loops timeline proxy.

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 room to room 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 rooms, the name of the room turns red to indicate the new content, and when they enter the room, 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 text outside of conversations.

Bulletin boards (Figure 1, callout 6) provide a means for posting text and URLs in a highly visible place. Each room has its own bulletin board, and its text may be edited by anyone in the room. When new or changed text is posted to a bulletin board, the new text is signaled to those in the room by the background color of the bulletin board fading

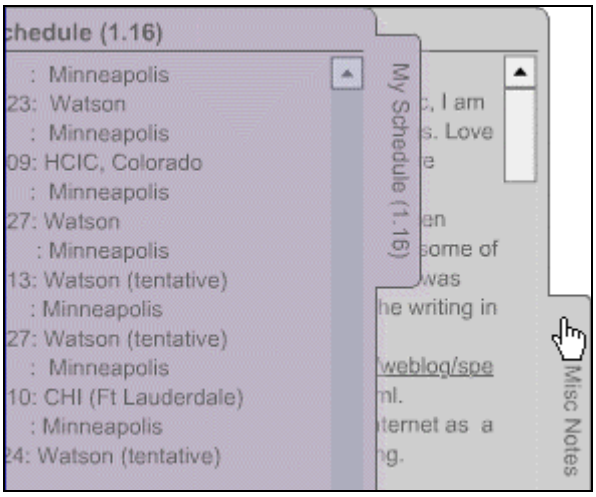

Figure 3 [cropped]. Two tabs: one fully opened ("Misc Notes"), the second sliding out over the first.

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 observations of and comments from Babble users.

The other means of making information readily available is the tab (figure 3). The tabs peek out from behind the conversation pane. Clicking on the tab causes it to slide out, revealing the (editable) information on it. Each room can contain up to three tabs; rooms 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

A final part of the Loops UI is the launching screen, partially shown in figure 4. This was a consequence of the requirement for supporting membership in multiple communities. It provides a single location where users of multiple Loops can sign on once, and have access to all Loops of which they are members. It also provides a place

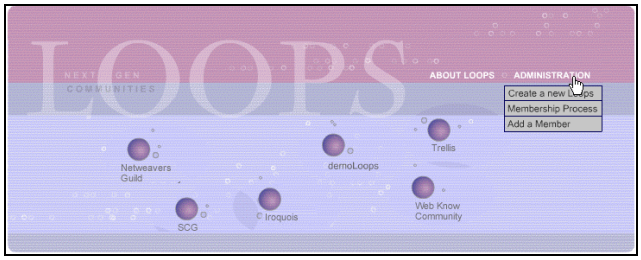

Figure 4 [cropped]. The Loops Launcher provides a view of all Loops communities hosted by a server; it provides single sign-on access to users of multiple communities.

where Loops users can create their own Loop (via the "Administration" menu). The new Loop is automatically set up, although users who are not members of other Loops must be added to the Loops environment by a system administrator. We had hoped to have the individual Loops icons reflect the degree of activity in each Loop, but this was not possible during our implementation time frame.

DEPLOYING LOOPS

In this section we discuss the results of our deployments. We will begin by describing some of the ways in which Loops users have made use of tabs and bulletin boards. Then we will turn to the more general question of the over all 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 in figure 5 are more interesting. The first example shows the bulletin board 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 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 (and indicates) everyone to trust that

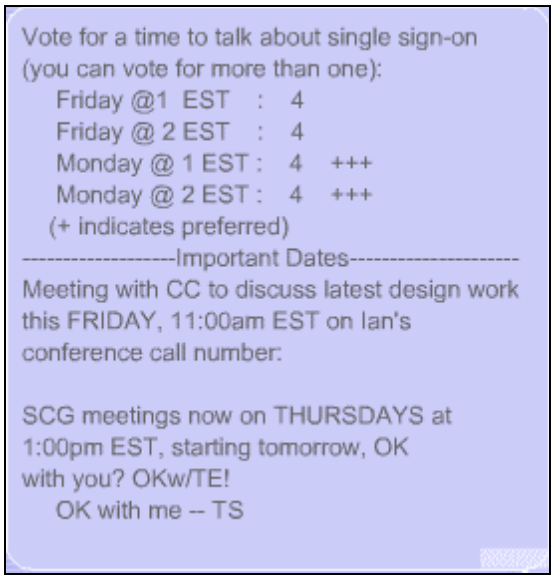

Figure 5. Two forms of bulletin board usage on one bulletin board: voting for a meeting time (top and bottom), and announcements (middle).

their colleagues will not 'cheat' by casting multiple 'votes' or forging initials. While have 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 are used in ways that are similar to bulletin boards, although (obviously) not for announcements. 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 an narrow writing area, this is not really 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 room 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

As of this writing, we have fully deployed Loops to about six groups.[1] 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 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

[1] It is not always clear what to count as a "deployment." Anyone who is a member of any Loop has the power to create a new one 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.

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 (the sixth has not been running long enough to say). In terms of the last definition of permanent use, three of the six are successful (again omitting the sixth, which has been running for about two weeks.

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, not just because it sheds light on how Loops is being used, but because its usage patterns are at odds with what we would expect. (A detailed ethnographic study of Fargo's use of Loops may be found in [10].)

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

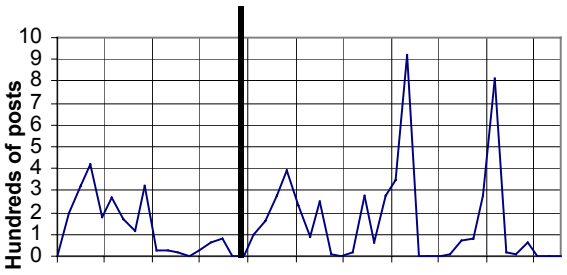

Figure 6. Posts per week (in hundreds of posts) in Fargo over the first 10 months

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.

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 Loops' development via 'group testing,' and the resulting user interface. Finally, we describe our experiences in deploying it to six groups.

While we think that our experience has useful lessons for developers—particularly along the lines of the 'group testing' process—the most significant conclusion we've drawn from our own work is that our initial focus on designing online environments as places for community has led us 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. The Fargo loop is a case in point. 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 channel (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 the Loop functions more as a one time meeting room.

If we relax the notion of Loops as being a space for an online community, a place where people hang out and where they return to 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 have need a much 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 your 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, if Loops becomes more of an occasionally occupied space, as is the case with Fargo, there need to be means for alerting participants when the meeting or event is about to start.

ACKNOWLEDGEMENTS

Thanks to everyone who helped. To be done post- post-anonymization.

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. Erickson, T. and Kellogg, W. Social Translucence: An Approach to Designing Systems that Mesh with Social Processes. *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. M. Ackerman, V. Pipek, V 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. (1996) Workspace Awareness Support with Radar Views. *Ext. Abstracts CHI 1996*, ACM Press (1996) 210-211
9. Guzdial, M. (1997) Information ecology of collaborations in educational settings: Influence of tool. *Proc. CSCL 1997*. Toronto, Ontario, Canada. p. 83-90.
10. Halverson, C., Erickson, T. and Sussman, J. What counts as success? Rhythmic Patterns of use in a persistent chat environment. To appear in *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. (2002) The Character, Functions, and Styles of Instant Messaging in the Workplace. *Proc. CSCW 2002*. ACM Press (2002), 11-20.
15. [NWB00] Nardi, B., Whittaker, S., and Bradner, E. Interaction and outeraction: Instant messaginging 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. (1996). Linguistic and interactional features of Internet Relay Chat. *Computer-mediated communication: Linguistic, social and cross-cultural perspectives* (ed. S. Herring), John Benjamins, 47-63