The Design of Loops: A Web-Based Environment for Persistent Conversation and Community

Thomas Erickson, Christine Halverson, Wendy A. Kellogg,
Mark Laff, Jeremy Sussman, Tracee Wolf
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598, USA
+1 914 784-6659
{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 groups in a corporate environment. We discuss the rationale behind the system, its goals, their realization in an implemented architecture and user interface, and our initial experiences with deployments of the system to two groups. In the course of this discussion we discuss two secondary themes: our design-oriented approach to research that focuses on system implementation and deployment; and our observations of and designs for the use of simple text for a wide variety of collaborative purposes.

Keywords
Awareness, Chat, Computer Mediated Communication, CMC, Conversation, Community, Design, Design Research, Structure, Text, Social Proxies, Visualization.

INTRODUCTION
Our chief goal is to design "socially translucent" systems — systems that convey social information and context by providing visual cues about the presence and activity of participants. We argue that such systems can, by taking advantage of the human ability to make inferences from traces of activity, provide an environment that supports a wide range of social processes (e.g. imitation; peer pressure) which permit groups to function effectively.

Our approach to making social information visible employs two tactics: social proxies and persistent conversation and social proxies. Social proxies are minimalist graphical representation of the presence and activity of participants; their aim is to provide a sense of both the synchronous and asynchronous activity in an online system. Persistent conversation refers to text-based computer mediated communication that persists over time. In particular, we find that chat — synchronous or near synchronous text-based CMC — is a rich source of social information due to both the expressiveness inherent in written language, and the variety of expressive mechanisms (e.g. emoticons) that CMC users have developed (see [3] for many examples).

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 [6], 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 upon this previous work, but opens up some new areas of discussion. Its primary aim is to describe the design of a second generation system called Loops. Thus we discuss the goals of the Loops design, the rationale behind these goals, their realization in an implemented architecture and user interface, and our experiences with initial usage and deployments. In this paper we will focus 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 use. In the process of telling this story, we will also discuss our development process, and the way in which our design-oriented approach to research has shaped the system we've developed.

This paper begins by providing background on the design context in general, and the Babble system in particular. In the next section we lay out the factors that shaped the design of Loops; we devote particular attention to examining how users constructed structures within the weakly-structured Babble environment. In the third section we describe the architecture and user interface of Loops. Finally we discuss our experiences with the working systems, include its deployment and use by thirty people..

BACKGROUND
Before turning to the factors which specifically drove the design of Loops, we'll say a bit about the context in which development occurred. Here we describe the position and role of the development group, the general situations for which we were designing, and the nature and use of the first generation system to which Loops was a response.

Design Research

In understanding the various forces that shaped the design of Loops, a good place to begin is with the context in which the design takes place. Our group is situated in the research division of a large corporation, and our charter is to carry out "adventurous research," that is, research that does not necessarily play into company's product development strategies.

Our work is best described by the rubric "design research." That is, conduct our research by embodying a variety of claims or conjectures in design prototypes, and then seeing how they play out that context. While some of this work may involve various forms of reflection and critique familiar to designers, or the genre of 'user studies' most familiar to HCI audiences, most of our work proceeds via the creation, implementation, and deployment of working systems to actual workgroups.

This implementation and deployment centered approach has a number of advantages. First, because our research agenda has to do with how to support *social* processes, it is in our best interest to be able to deploy robust, working systems to real work groups. Group-based activity takes weeks to months to coalesce, and the only effective way we see of studying it is to observe it 'in the wild.' Second, even though we are chartered to do "adventurous research," it is nevertheless very useful to be able to point to working systems being used by internal groups as an additional way in which we are 'returning value' to the company. This is particularly valuable at a time when industry-based research does not seem to be faring well. Third, since part of our mission is to influence product development groups to adopt our ideas, having working systems deployed to dozens to hundreds of people is a useful way of distancing ourselves from the stereotype of 'blue sky' researchers sometimes found in more development oriented groups. Finally, having an existing set of users — or, more particularly, an existing set of groups-as-users — provides us with a readily available means of testing everything from vague ideas to new systems.

We emphasize the advantages of our implement-and-deploy approach because it has a lot of costs. These stem, of course, from the need to construct systems which are sufficiently robust that they can be deployed to other workgroups. Ideally, as has been the case with Babble, the system will not only be robust, but will be sufficiently attractive to entice a number of groups into using the system. And this, in its turn, means that the ability to easily deploy and update the system is critical—something which became an important factor in the design of Loops.

The Babble System

Our first generation system, a Smalltalk-based client and server system, was called Babble. Here we provide a very brief description of it; interested readers should see [6] for more details of its design.

Babble was designed to serve the needs of small to medium sized corporate work groups. It was intended to provide a semi-private online conversation area where members of groups such as workgroups, committees, and special purpose task forces could have text-based synchronous or asynchronous conversations. Among the assumptions embodied in Babble were that the groups would be relatively small, that most participants would know one another or *of* one another, and that, because participants were identified and situated in an organizational context, there would be considerable pressure to behave responsibly.

Figure 1 shows a screenshot of the Babble user interface. From the upper left its basic components are: the list of users who are currently logged on; a visualization of the presence and degree of activity of users called a social proxy; a hierarchical topic list of user-definable conversations; and, in the lower half of the window, the text of the current conversation. There are three things to note. First, although Babble resembles a chat system, the conversation in Babble may be synchronous or asynchronous: that is, remarks may be separated by seconds or by minutes, days or months. Second, Babble provides a number of cues about who is present and what is new. The social proxy in the upper middle pane shows not only who is in the current conversation (the dots within the circle), but how recently the participants have 'spoken', i.e. typed, or 'listened', i.e. clicked or scrolled (via the proximity of the marbles to the circle's center). The topic list in the upper right pane has 'mini-proxies' next to a conversation name if anyone is in it (e.g. the third topic), and displays the topic name in red if it contains material the user hasn't seen. The third point to note is that the topic list is user definable: that is, users can create new conversation topics, rename them, and organize them hierarchically in nested categories — we will examine examples of such organization when we turn to the issue of structure.

Babble Deployments

Over the past four and a half years we have deployed Babble to about two dozen groups. Most, though not all, deployments have been within our company to groups such as official (i.e. organizationally defined) work groups, ad hoc task forces, and special interest groups. Most deployments have been intended to provide environments for long term collaboration, although some have been

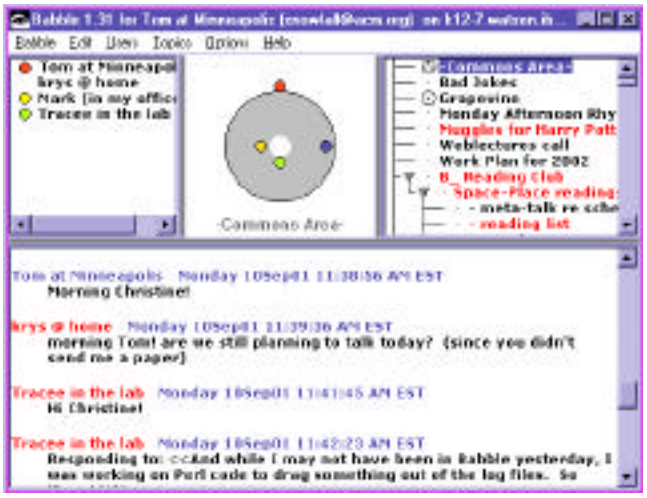

Figure 1. The Babble user interface.

deployed to support shorter term online events (ranging from three days to a month or so).

Our success in deploying Babble has been mixed. Initially, we had limited success in getting groups to use Babble for lengthy periods of time. Although almost all groups initially used Babble enthusiastically, usage often fell off after about six weeks (see [Brader1999]). Over time, based on our observations of successful deployments, we developed a screening process, advice for moderators, and suggestions for new users, that appear to have increased the rate of successful adoption. In the last year, the primary diffusion of new Babbles has been via word of mouth: that is, a member of an existing Babble requests a new deployment to support some other group in which he or she participates and takes charge of launching and moderating it; in these cases the presence of one or more experienced 'Babblers' seems very beneficial to adoption.

FACTORS THAT SHAPED LOOPS

The design of Loops was shaped by three factors: deployment logistics; the desire to retain the most successful features of Babble; and the desire to support the creation structure within the online environment.

Deployment Logistics

As already noted, the ability to successfully create and deploy working systems to groups is critical to the success of our design research approach. This is particularly true if the system turns out to be popular, as has Babble.

Babble was not easy to deploy. The client, written in Smalltalk, was 2 to 3 megabytes in size. This meant that installation was non-trivial: users had to run an installation package that they usually downloaded from the network. The drawback is that this leaves the timing of the install up to individuals, and thus installation in most groups was staggered across several days. This made it more likely that those who installed Babble right away would log on and, finding no one to talk to, be uninclined to return.

Because Babble was a research prototype, it changed over time. This lead to the problem of trying to maintain compatibility across versions. Considerable efforts — in both programming and testing — were made to make sure that changes to the server or client did not break older versions of the client. However, incompatibilities did arise, sometimes by accident, and sometimes due to the desire to add new functionality that required fundamental changes. When this happened, all users need to reinstall more or less simultaneously, or lose access to the Babble environment when the server changed or incompatible clients came on line. The first case was difficult to manage for the reasons already mentioned; the second was disruptive to group use.

A third problem arose when trying to deploy to tightly administered environments. For example, one deployment of Babble was to a University class, which meant that Babble needed to be available in the public University computer labs. However, the lab administrators were wary of installing 'experimental' software on their lab machines, fearing that it might crash, corrupt disks, or have infelicitous interactions with other software. This particular problem was solved by creating a version of Babble that ran off a CD-ROM and kept its user-specific files on a diskette — however, this was not really a viable long term solution, and it exacerbated the update problem by requiring the burning of new CDs. While deploying to a public lab is a special case, note that similar concerns arise in most mission critical environments, such as help desks, sales support operations, etc., as well as any environment where hardware and software support is centrally controlled.

In considering the next generation design, each of these problems pushed us in the direction of developing a web-based client that could be run within a standard browser. Such a solution addresses the deployment problems that are inherent in the design research approach we're pursuing, as well as making it easier to support multiple hardware and OS platforms, and people using multiple machines. However, we were rather reluctant to pursue this approach, because of the issues we discuss in the next section.

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 signs of activity over time.

One of our concerns was that our emphasis on synchrony, awareness, presence, and inhabitation might prove difficult in an environment that was fundamentally asynchronous and lacked any concept of state. That is, whereas the Babble clients maintain a continuous connection to the server, web browsers do not: they connect with the server only to send or receive information. It was not immediately clear to us how to maintain the feeling of presence (i.e. that users are continuously connected to the environment), and we were concerned that we might end up trying to maintain an illusion with no infrastructural means of support.

In addition to these infrastructure concerns, we were also interested in creating an engaging user experience. We had two goals. First, we wanted freedom to use a range of subtle visual and auditory effects in designing successors to the Babble social proxies. Second, we wanted to allow our interaction designer to directly work in the medium, rather than having design prototypes reinterpreted by a programmer. As noted by Houde and Sellman [7], most development environments do a good job of supporting design or programming, but not both. Although we explored alternatives, these user experience goals eventually lead us to build the Loops client in Macromedia's Flash, an environment well-known for its ability to support the creation of interactive animations that can play in browsers.

Supporting Structure

As already described, we have considerable experience in deploying Babble. While we have discussed many aspects of our studies elsewhere, we have said little about how users have actually turned Babble to their own ends. In this

section, we examine some of the ways in which Babble users have structured information within Babble to support their needs. We consider this structuring quite significant because Babble provides only rudimentary support for it.

To begin with, we need to say a bit more about how structure is created in Babble. Basically, Babble allows any user to create or rename the topics and categories shown in the hierarchical topics list (Figure 3, upper right). Topics are simply names for single conversations (analogous to documents in a GUI-sense), and categories are a means of grouping conversations (analogous to folders) which can contain topics or sub-categories. The only way to create structure in Babble is by creating named hierarchies of categories and topics.

For the purpose of this paper, we will look at data drawn from the five active Babble deployments (see Table 1 for a summary). It is important to note that the explicit analysis we present here did not drive the design; it is really to persuade the reader. Having spent years watching the users of dozens of Babbles structure and re-structure their environments, the conclusions drawn from this data were already evident to us.

Perhaps the most obvious feature of these Babble deployments are the number of categories and topics and categories in evidence. The number of user-created topics and categories per Babble ranges from 62 to 170, with the average being 129 (these counts exclude automatic archives of conversations generated by the system, and topics and categories that users deleted). When one considers that there are a relatively small number of regular, active users who contribute to conversations (typically 10 to 20, but around 30 for B1), this is quite a lot. Why is this happening?

What we see when we look at the lists of categories and topics is that quite a bit of the structure which has been generated is for presenting and organizing static information. That is, conversation topics are often not used for conversation. Instead, what we see is that users are trying to create meaningful structures which, while they often contain niches for conversation, are larger in scope.

Table 1 summarizes the most common types of structure observed across Babbles. These include personal places (places 'owned' by a particular user), structures intended to support events (e.g. an upcoming conference) or projects, and places for announcements. Note that the counts shown in Table 1 are quite conservative; they reflect only existing structure (not structure created and later deleted by users), and only structures that are named so that an outsider can recognize their purpose.

The most prevalent form of structure is what we will refer to as an 'office'. Offices are topics, or, most often, hierarchies of categories and topics, that 'belong to' and are named after a user. The third row of Table 1 shows the number of distinct offices in each Babble (the top number), the total number of topics and categories devoted to offices (the second number), and the percentage of structure in that deployment that is devoted to offices. As Table 1 shows, offices comprise from one to three quarters of the structure (i.e. number of category names and topic names) in these Babble deployments. A common form for an office is:

Pat's Place
    About Me
    Talk with Me

the first item being a category, and the next two items being topics contained within it. The first topic is typically intended to contain a profile of the person, and the second as a place for conversation.

The event and project structures shown in row 4 are similar offices, in that they distinguish between topics intended for conversation and topics for information, although they typically contain much more structure. 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 specially for conversation, occurs across all Babble deployments for a variety of different types of structures.

	B1	B2	B3	B4	B5
Months of use	10	6	6	6	5
Total structure in terms of # of topics + categories	141	147	170	62	126
# of distinct 'offices'	28	47	20	15	12
Amount of structure (t+c)	90	109	46	38	33
% of total structure	64%	74%	27%	62%	26%
# distinct events/projects	3	1	2	1	3
Amount of structure (t+c)	14	9	19	1	53
# announcement topics	1	0	2	2	3
Amount of structure (t+c)	1	0	2	2	3

Table 1. Summary of the use of structure — topics (t) and categories (c) —in five active Babble deployments.

Another structural feature found in most Babble deployments is the attempt to explicitly or implicitly create one or more topics or categories for announcements (row 5 of Table 1). Most often these are named "Announcements;" other examples are "Heads Up!," "News," and special purpose announcements like "Kittens Free to a Good Home!" (However, most announcement structures do not seem successful, judging by their degree of use; one of the moderators reports that people wouldn't come to the announcements topic quickly enough, and so he shifted to posting announcements in a topic where most participants in that Babble 'hung out.')

To summarize, throughout the various deployments it is clear that users are doing more than creating places to talk. First, they are trying to display static information in a readily accessible way. Thus, they resort to using topic names to signal whether the topic is supposed to be a place for conversation (e.g. "Talk to me!" "Questions," "Chit-chat," "Discuss <Project Name> Here"), or whether it is primarily informational ("About Me," "<Project Name> Status," "About"). Second, users are trying to make certain types of information visible. The most obvious example of this is the "Announcements" topic, which, by virtue of its name, appears at the top of the alphabetically sorted list. Third, and more generally, all five Babbles exhibit attempts to structure topics within particular categories by

using numbers or punctuation characters as prefixes to control their sorting order (most often trying to put informational topics first in the list as with "-Where to Start"). Both the desire to display structured static information, and the desire to control the visibility of information, are taken up in the design of Loops.

LOOPS: THE WORKING SYSTEM

Thus far we've described our overall goals, the design context, and the factors which shaped the design of Loops. In this section we describe the resulting system. Currently, all the major functionality is implemented, and we have, in the last week, upgraded our client code base from Flash 5 to Flash 6, which has resulted in an enormous improvement in performance and in the elimination of some of the most vexing bugs.

Conceptually, Loops consists of a set of user-definable rooms, each of which can contain a conversation, tools, documents, static text, and people. The basic user experience is that people connect to Loops server, and move from room to room, reading conversations that have changed in their absence, contributing new comments to synchronous or asynchronous conversations, and encountering other users as they do so. As with Babble, the ultimate goal is that Loops feel like an inhabited place in which 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.

The Architecture

Loops uses a client-server architecture, with the client and server communicating via TCP/IP, with content coded in XML. A server written in Java uses IBM's DB2 database as a persistent store; the client is implemented in Macromedia's Flash 6 and runs in any standard web browser that supports the Flash 6 plug-in. The server maintains a list of currently connected clients, the contents of the Loop's conversations, and a log of user activities used to drive the Loop's social proxy and other elements of the interface that show traces or results of user activity. This architecture allows us to track user location and activity within the conversation space (e.g., which conversation a user is 'in', how long since the user visited or posted to a conversation, etc.), thus supporting the user experience described in the next section.

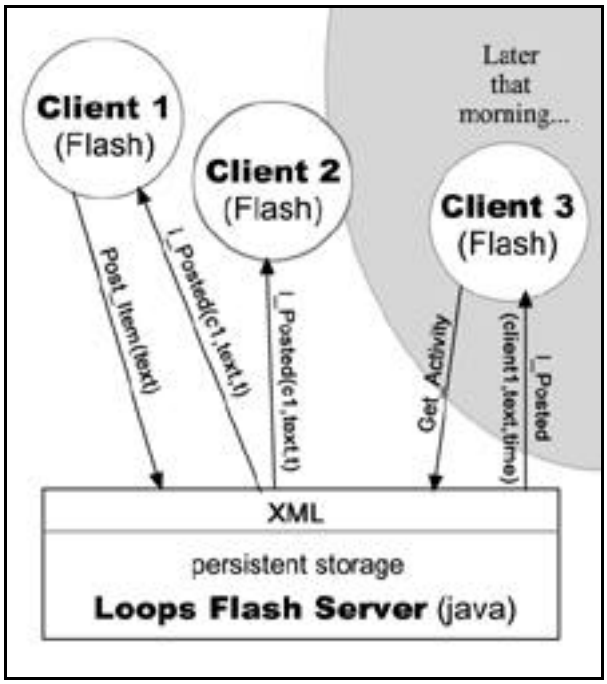

Figure 2. The Architecture

Figure 2 illustrates how the architecture works. The story begins with the user of client 1 entering a remark. The client wraps the comment in XML (along with meta information regarding the conversation it is being posted to) and sends it to the server (e.g. the "Post_Item(text)" request). Upon receipt, the server processes the request: it verifies that the client has appropriate permissions, creates a command by adding meta-information such as the client_ID and time, and stores the command in its activity log. Then, the server broadcasts the result to all connected clients (e.g. the "I_Posted(client1,text,time)" command), including the originating client. When the clients receive the command from the server they act appropriately (in this case, displaying the new comment in the conversation window, and updating the social proxy to reflect client 1's activity).

This story is complicated by two factors. First, not all users may be connected simultaneously. As shown in Figure 2 (gray area), when Client 3 connects at a later time, it issues a "Get_Activity" request to the server. The server responds by sending it the activity and content for the conversation that Client 3 is viewing ("I_Posted (client1,text,time")), which, in this example, is the same conversation to which Client 1 originally posted. The second complicating factor is the (common) case in which all connected users are not viewing the same conversation. Thus, suppose that Client 4 (not shown) was present when Client 1's comment was posted; but was viewing a different conversation. In that case, Client 4 would receive the same I_Posted command that the server broadcast to Clients 1 and 2, but would use it differently: Client 4 would update elements of its social proxy (e.g. those that show the time of Client 1's recent activity) and may cache Client 1's comment. (A client caches conversations it is not displaying only when it has previously visited that conversation; otherwise, it does nothing and, if and when it switched into Client 1's conversation, it would use Get_Activity to obtain the contents).

The User Interface

We'll begin with an overview of the general interface elements of Loops. The basic features shown on the screen layout (Figure 3) are from left to right:

- the social proxy (upper left), which provides cues about the awareness and presence of others, with a list of users logged on below it as well as icons the provide access to user-related functionality
- the pane where the text-based conversation occurs
- a row of slide-out tabs (peeking out from under the conversation pane)
- three bulletin boards for editable static text

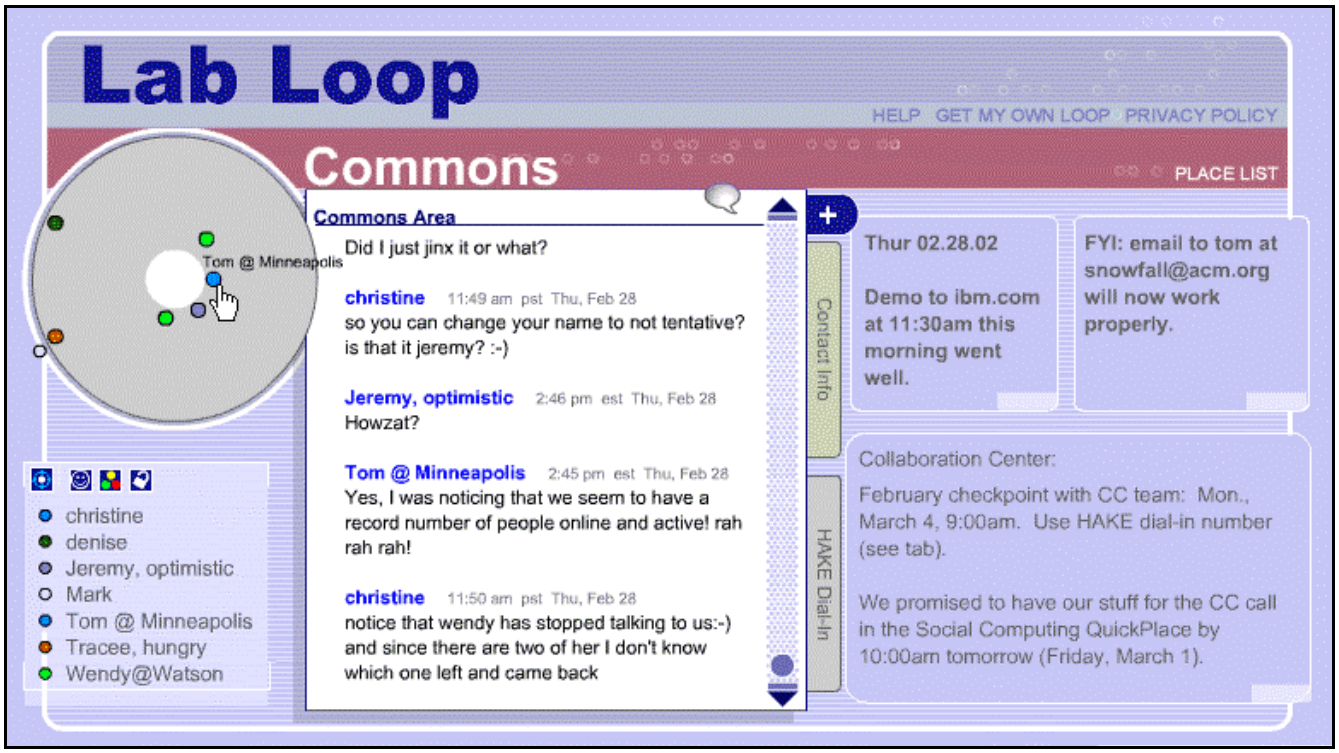

Figure 3. The Loops User Interface (screen height reduced by a third).

· and, in the upper right portion of the screen are several menus, most importantly the "Place List" which enable users to move to (and create) other places in the environment.

In addition, functionality is attached to the screen elements: thus, the row of icons along the top of the user list (lower left) enables management of user accounts and preferences, whereas mousing down

Figure 3 shows a view of the Commons, the central gathering place of this Loop. The particular contents of the Commons are the conversation pane (showing a segment of not-quite-synchronous banter), the social proxy (showing that six people are in the room, and a seventh is logged on but elsewhere in the Loop), the bulletin boards (containing various announcements and reminders), and two tabs (one containing a list of group phone numbers, and the other information about a group number for conference calls).

Awareness and Conversation

Because the awareness and conversation models were transposed from Babble, we'll cover them briefly.

The chief awareness interface element is the social proxy, the circle in the upper left corner of Figure 3. The circle represents the conversation being viewed, and the colored dots (referred to as marbles) represent people. A marble inside the circle means that a user is in the current conversation; when users are active (meaning that either they type or click) their marbles move to the inner ring of the circle (as in Figure 3), and then, over the course of 15 minutes, they drift to the edge of the circle. Marbles shown outside the circle represent users who are connected to Loops, but are viewing different conversations.

Loops implements the lightweight type-and-post, conversation-in-one-window model that Babble supported. A user adds to a conversation by clicking on the speech bubble icon in the lower right corner of the conversation pane and typing into a moveable dialog box. This arrangement, which differs from instant messaging and chat, allows users to refer to the either the existing conversation, or to move to and copy from other conversations, thus making it easier for them to create responses that draw together different threads of conversation. Once the user posts a comment, it immediately appears in the conversation. For users who are in other rooms (or who log on later), the name of the room in the drop down places list turns red to indicate the new content.

Static Structure: Bulletin Boards and Tabs

As noted earlier, there was considerable evidence that Babble users devoted a lot of effort to making information accessible. In response to this, Loops provides two ways of structuring static information: bulletin boards and tabs.

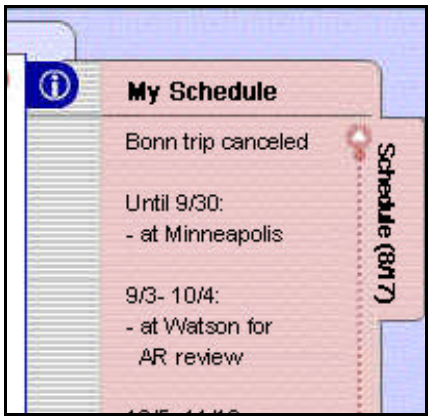

Figure 4. An opened tab.

Bulletin boards (Figure 3, right) provide a means

for posting text in a highly visible place. Each room has its own bulletin board, and its text may be edited by anyone who has write permission for 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 out and then fading back in with the new text displayed. We anticipated that bulletin boards would be used for purposes ranging from announcements and reminders, 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. Each room can contain up to three tabs. The tabs peek out from behind the conversation pane. Clicking on the tab causes it to slide out (Figure 4), revealing the (editable) information on it. The lower portion of the tab provides access to controls for setting its background color, and clearing and deleting it. We expected that tabs would be used for activities such as sharing lists of URLs, schedules, and keeping to do lists.

One role that we expected both bulletin boards and tabs to play is to visually distinguish rooms from one another. In Babble, rooms looked almost identical; in Loops, the text in the bulletin board, and the number, and colors and names of tabs should make rooms feel more distinct.

EXPERIENCE WITH LOOPS

As of this writing, Loops has been implemented, mostly debugged, and performance problems that were delaying deployment have been solved with the release of Flash 6. (Note that the test deployments reported here occurred before the change to Flash 6.)

Now we turn to our experiences in using and deploying Loops, and the lessons derived from them. We describe our user groups, our methods for studying them, and report on their opinions, feedback, and the usage practices we have observed.

User Groups

We have three Loops deployments which are guiding our evaluation and refinement of system.

The Lab Loop: Reflecting on Our Own Use

Our first source of experience is our own use of Loops. It is our practice to live in the software we are developing. This has obvious limitations — as its developers we are obviously motivated to see the system succeed. Nevertheless it is a valuable way to detect problems (if it's a problem for us, it will probably be a problem for anyone), and through ongoing use to support our daily activity, we gain experience of some of the possible ways in which it may be turned to various ends.

The lab consists of six core participants who are members of an officially defined corporate workgroup; there are also a smaller number of participants from outside the offical group, primarily associates and collaborators of lab members. It is important to note that our lab is distributed — two of our six members are remote workers and a third is frequently on the road — and is thus we are, in fact, an example of the sort of group for which the software is intended. Furthermore we have used Babble as one of our primary collaboration mechanisms for over four years, and thus have a substantial repertoire of collaborative behaviors that we wish to continue in the Loops environment.

Several attempts were made to shift our daily collaborative activity from the Lab Babble to the Lab Loop; the first few failed when our enthusiasm proved insufficient to compensate for the performance limitations of early versions and our very real need for an effective collaborative environment to support our daily activity. Performance had improved sufficiently by early January that all activity shifted to the Loop; as of this writing, we have about three months of group experience. For the purposes of this paper, we will primarily draw upon our own usage of Loops as a source of examples of how the system can be turned to particular ends. For indications of its usability, opinions of its value, etc., we turn to two other groups.

The Awareness Loop: A New Group

The Awareness Group is currently comprised of four people (two others were invited but have not appeared); they are not an official corporate group, but are instead collaborators from different parts of the company who wish to be in closer touch. The two core members of the group are close collaborators and typically make heavy use of SameTime, (an instant messaging application) to communicate with one another. The members of the Awareness Loop expressed interest in being beta-testers, and so, when performance reached what we felt was an acceptable level, we created a Loop for them, and invited them to 'move in.' As of this writing, they've been using Loops for about two weeks. Except for requesting that they give us feedback, there were no requirements or restrictions on their use.

The Net Weavers: A Test Drive by an Existing Community

The Net Weavers is a pre-existing cross-organizational group whose goal is to foster the creation and maintenance of communities within our global company. Net Weavers, as a whole, consists of over 100 people; they use a variety of mechanisms to interact, but in particular about thirty of the members inhabit one of the most active Babble deployments. We thought that their collective experience in using Babble would make them an interesting test case for Loops; similarly, the prospect of Loops ultimately replacing Babble seemed as though it would motivate them to be properly critical.

Because the Net Weavers' principal organizer was concerned about fragmenting the part of the community that used Babble, we agreed to do a limited time trial, which we called the "test drive." We used a broadcast email to the entire Netweaver's mailing list, and created a guest account to provide all comers with access to the Loop. Over the four days of the test drive, 30 people accessed the Net Weavers 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 i the Net Weavers' Babble.

Methods

Participant Observation

Our primary method of gathering information at this stage of the design process is to observe and participate in the Loops, 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 on the system, critiques often took on a dialectic character. Sometimes agreement about problems was easily arrived at; other times, disagreements arose and led to discussions which revealed differences in assumptions, prior experience, cultural values, previous experience, etc.

A Survey of the Awareness and Net Weavers Loops

A potential drawback of the collective nature of this process is that it may result in artificial emphases (or under emphases) of certain problems. That is, a particular problem may get a lot of discussion, not because it is the most serious, but simply because it has become a momentary focus of attention. Alternatively, people may refrain from commenting on a problem, because they see that it has already been discussed.

To try to get a clearer picture of users' preferences and priorities, we printed out transcripts of all discussion (about 42,000 words of text), and did a rough analysis to identify problems, controversies and suggestions. From this we developed a structured survey was designed to be circulated by email, and could be completed in no more than 10 minutes. It was emailed to the thirty participants in the Awareness and Net Weavers Loops, shortly after the end of the Net Weavers Test Drive, and twenty one Loops users completed the survey.

The main portion of the survey made four to five statements about each element of the interface, and used a 7 interval scale (strongly disagree to strongly agree) to quantify results. Statements ranged from probes about particular details of the interface ("I liked the use of local time stamps for each user (e.g. EST, GMT)" to elicitations of general preferences ("I think bulletin boards are useful and should be kept..."). The second part of the survey asked users who were experienced with Babble to compare the two systems, and asked more open ended questions, including queries about which interface elements merited the most screen space. The final part of the survey asked users about their degree of experience with relevant applications (e.g. Babble, Sametime), and the characteristics of their hardware and software platforms.

General Usage and Preferences

In general, both groups made extensive use of Loops. Between them, the Awareness and Net Weavers Loops produced approximately 42,000 words (or 5,000 and 3,300 words per day of use, respectively). Individuals varied considerably in their usage patterns, but the median user reported logging onto Loops 3 times, and spending about 3 hours on line. We take the number of users, frequency of use (including return visits), and amount of content produced as a sign that the system is basically usable.

Although participants made frequent positive comments about features of Loops in their online talk, we tend to discount these, given that people often like to be 'nice' in public situations and that positive comments may be viewed as a way of taking the sting out of the more critical remarks being solicited. We place more confidence in the results of the surveys, which were filled out privately and outside the excitement of 'test driving' a new system. In particular, probes which asked Babble users (all but 5 of the respondents) about their relative preferences for Babble versus Loops gave encouraging results. One probe showed a preference for Loops (14 agreeing, 2 neutral, 1 disagreeing) provided its performance problems and obvious bugs were fixed (something accomplished with the move to Flash 6). Another showed that only a minority actually was against moving to Loops with no fixes whatsoever (5 in favor, 3 neutral, and 6 against). The definitive test will come, of course, when we give existing Babble communities a chance to 'vote with their feet.'

Feedback on Tabs and Bulletin Boards

The primary goal of our deployments in general, and of the survey, in particular, was to get feedback on the relative success of the interface elements of Loops. We were particularly interested in the reception of the two brand new elements, the tabs and bulletin boards. We were also interested — based on our own experiences — in how new users would react to a variety of changes (including inability to implement certain details of Babble interactivity) in the model for displaying and adding to conversation, however we focus on tabs and bulletin boards here.

With respect to the tabs and bulletin boards, we were disconcerted to observe that there was considerable confusion about whether the tabs and bulletin boards were public areas or private ones. To us, a fundamental aspect of Babble (and hence of Loops) was that everything in the main window was public, and we expected most users (i.e. those who were experienced Babble users) to share a similar conceptual model. However, in our observations of the two deployments, this did not appear to be the case: "Yow. So when I thought I was typing a note to myself, I was really broadcasting to everyone?" This is an example of something we translated into a survey probe. According to the survey, a substantial number of users reported initially believing that tabs were private spaces (8 private, 12 public, 1 don't know), but only a few believed the same of bulletin boards (3 private; 14 public; 4 neutral or don't know). Of course, we have to be cautious at taking these statements at face value, since they are retrospective reports and some users may have been cued into the pubic nature of each elements by discussion that had taken place by the time they had arrived. The main takeaway point here is that the public nature of tabs and bulletin boards needs more design focus.

Other discussions raised issues about access control, usage models, and the purposes of tabs and bulletin boards. For example, strong opinions were expressed about limiting the ability to delete material from tabs ("Dang Dang Dang Dang Dang Dang.... This is now the third time I'm re-

writing the content for the [Net Weavers] Tab... Others keep deleting this content!!!!!!!"), were mirrored in the survey results. On the other hand, users were less concerned about protecting the contents of bulletin boards, perhaps because the were viewed as more public spaces. Over all, although most users weren't entirely sure how they would use bulletin boards or tabs, by the end of the test period most users felt both elements were useful and should be retained.

Emergent Usage Practices

The week to two weeks during which the Awareness and Net Weavers Loops have been in use is too brief to see the formation of any usage norms. However users conducted some interesting experiments which we describe here, along with usage practices that are evolving in our own use.

Figure 5 shows two uses of tabs, one from the Net Weavers' Loop (foreground), and the other from the Lab Loop. In the Net Weavers example, one user ('Bob') created a tab and reflected out loud on what it was for. Another user adds material (speaking in the third person to signal that this is his addition) and goes on (not shown) to propose a collaborative writing effort, composing group poetry (a game played —typically with limericks — in the Net Weavers' Babble. The example in the background is from the Lab Loop, and illustrates an interesting attempt at doing collective composition. Here, the text being composed is written in the tab (where it can be edited by anyone), and the conversation pane is used as a question and answer area. Whether this use will catch on remains to be seen, but it is a nice example of a mode of use unexpected when the element was designed. In the Lab

Figure 6. Bulletin Board uses: (a) character graphics art and public comments (the Net Weavers Loop, right and front); (b) voting and announcements (the Lab Loop, left and behind);

Loop tabs are also used in ways that were anticipated (for example, figures 3 and 4 show uses of tabs for holding common reference information such as a conference call number and group member's travel schedule).

Figure 6 shows two different uses of bulletin boards. The superimposed picture shows the first uses of bulletin boards in the Net Weavers Test Drive. The bulletin board on the left shows a character graphics sketch created by one user, and the one on the right applauds the feature as a "Wiki Area," although users soon discovered that the bulletin board lacked the undo features of Wikis). The bulletin board in the background picture is from the Lab Loop, and shows an example of an unexpected use that has developed into a norm: the use of the bulletin board to

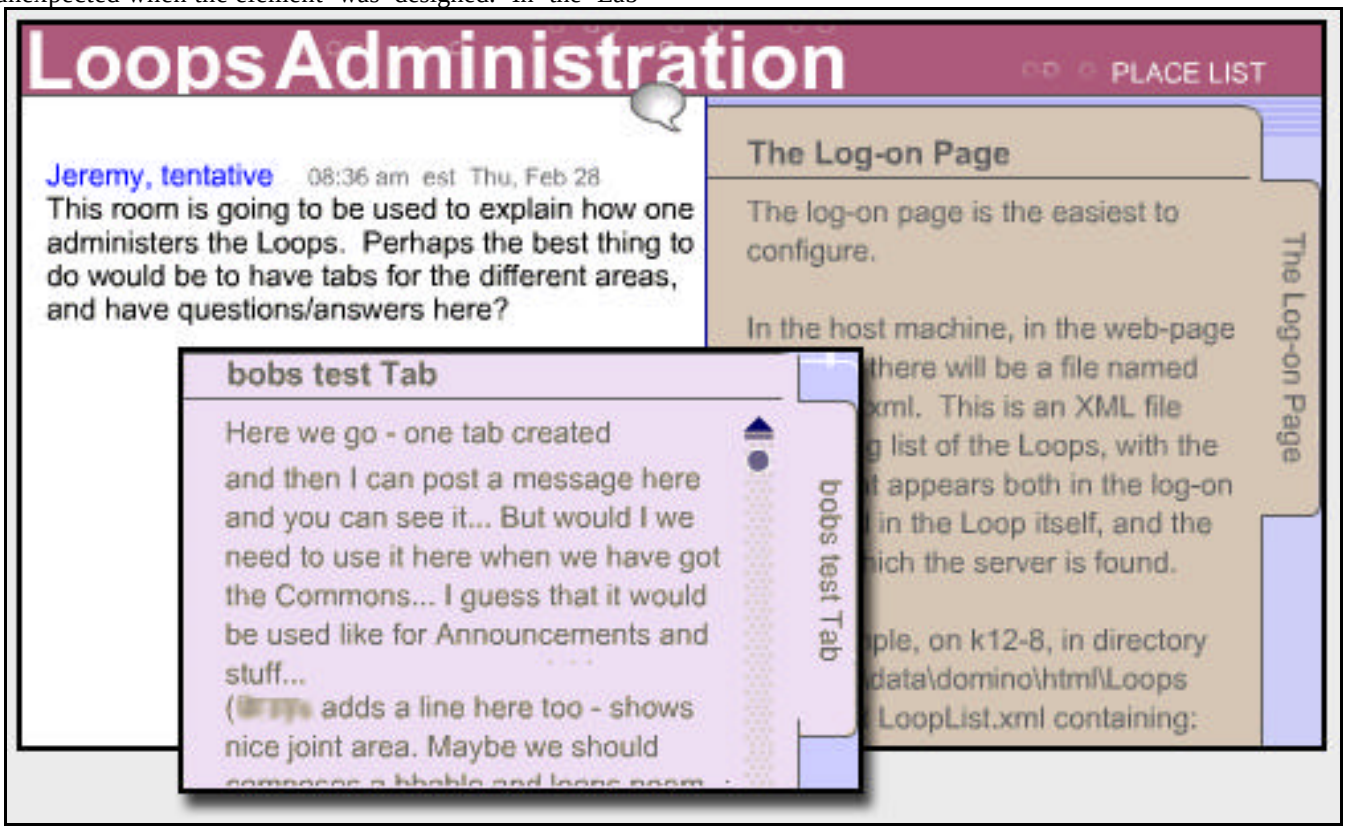

Figure 5. Two uses of tabs: (a) collaborative writing from the Net Weavers Test Drive (below and front)
(b) text development in a tab, using the conversation as a question and answer are, from the Lap Lop (above and behind).

propose and vote on possible meeting times; this particular instance started out with just numbers, and then another user began using pluses to indicate preference. In other instances of this sort of behavior, users use their initials to vote (shown in the lower portion of the bulletin board), or otherwise indicate preferences. Since bulletin boards do not track who enters or edits which text, this sort of use requires a significant amount of trust among group members. A use of the bulletin board more in line with what was originally intended is shown in the middle portion of the board, which contains a reminder about a conference call (and a tab — not shown — just to the left of the bulletin board contains the conference call number and access code).

Other Experience

We'd like to conclude this section by noting that user input was not limited to complaints about problems, expressions of preference, or even novel use of design elements. Users of both Loops discussed a number of issues that will contribute to future design work. These included debates about how to support 'threads,' the desirability (or not) of being able to participate in multiple conversations simultaneously, and the ways in which the Loops conversation model felt different from that of other applications. Similarly, users not only critiqued the existing design, but generated new ideas, one of the more promising of which is shown in figure 7, a design showing one way of merging the Social Proxy, the User List and the Places List.

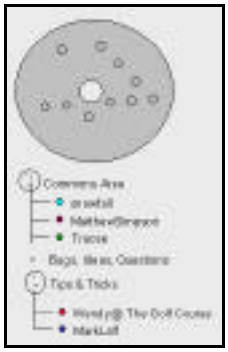

Figure 7. A design idea for merging the social proxy, user and places lists from a Net Weavers participant.

CLOSING REMARKS

Babble and Loops have similarities to a several existing types of systems. On the surface, they bear a strong resemblance to multi-channel chat systems (e.g. [12]), with their transcript of (potentially) real time conversation, and their lightweight conversation model (just start typing). However, below the surface, the conversation model bears a stronger resemblance to traditional bulletin board or asynchronous discussion systems, in that the conversation persists over sessions, and may be carried on asynchronously. And, on top of all this, Babble and Loops provide visual cues about the social context of user interactions, an approach which has been explored by Donath and her colleagues (e.g. [4, 11]), Sack [8], Smith [10] and many others.

In yet another sense (and in our opinion this is the closest fit), Babble and Loops resemble MUDs and MOOs (e.g. [2, 9]) in that they provide a set of 'rooms' that persist over time, and through which users move. Babble was, from this point view, almost the inverse of a MOO: whereas MOOs provide a persistent frame of (textual) rooms and artifacts within which ephemeral chat occurs, Babble provides a minimal frame of rooms (category and topic names only) within which persistent chat occurs. From this perspective, our design trajectory from Babble to Loops is essentially making Loops more MOO-like by providing artifacts like tabs and bulletin boards which permit the creation of persistent textual features for rooms. Given that this direction proves valuable — and we are encouraged by our initial experiences — we intend to continue in this direction, developing Loops rooms that have more internal structure to them, and expanding the range of artifacts within rooms, and integrating their presence and use with our visual awareness mechanisms.

REFERENCES

1. Bradner, E., Kellogg, W, & Erickson, T. (1999) The Adoption and Use of Babble: A Field Study of Chat in the Workplace. *The Proceedings of ECSCW*.
2. Bruckman, A. & Resnick, M. (1995). The MEDIAMOO project: Constructionism and professional community. In *Convergence*, 1:1, 1995.
3. Cherny, L. *Conversation and Community: Chat in a Virtual World*. Palo Alto: CSLI Publications, 1999.
4. Donath, J., Karahalios, K., & Viegas, F. (1999). Visualizing conversation. *Proceedings of HICSS-32*, Maui, HI, January 5-8, 1999.
5. Erickson, T. and Kellogg, W. (2000) Social Translucence: An Approach to Designing Systems that Mesh with Social Processes. *Transactions on Computer-Human Interaction.* Vol. 7, No. 1, pp 59-83. New York: ACM Press.
6. Erickson, T. Smith, D. N., Kellogg, W. A., Laff, M. R., Richards, J. T., and Bradner, E. (1999) Socially Translucent Systems: Social Proxies, Persistent Conversation, and the Design of 'Babble.' *The Proceedings of CHI '99*. ACM Press.
7. Houde , S. and Sellman, R. (1994) In search of design principles for programming environments. *Human Factors in Computing Systems: The Proceedings of CHI '94*, pp. 424-430. New York: ACM Press.
8. Sack W. (2000) Discourse Diagrams: Interface Design for Very Large Scale Conversations. *Proceedings of HICSS33s*, January 4-7, 2000.
9. Schlager, M., Fusco, J., & Schank, P. (1998). Cornerstones for an on-line community of education professionals. *IEEE Technology and Society Magazine*, 17(4), 15-21/40.
10. Smith, M and Fiore, A. (2001) "Visualization Components for Persistent Conversations." *Proceedings of CHI 2001*. ACM Press.
11. Viegas, F.B. and Donath, J. Chat Circles. (1999) *Human Factors in Computing Systemss: The Proceedings of CHI 99*, pp 9-16, ACM Press.
12. Werry, C. C. (1996). Linguistic and interactional features of Internet Relay Chat. In S. Herring (Ed.), *Computer-mediated communication: Linguistic, social and cross-cultural perspectives.* Amsterdam: John Bemjamins, pp. 47-63