Structuring and Supporting Persistent Chat Conversations

David Fono, Ron Baecker
University of Toronto
40 St. George St, Toronto, ON, M5S 2E4
fono@dgp.toronto.edu, rmb@kmdi.utoronto.ca

ABSTRACT

Persistence of conversations has been found to be a useful feature in group chat tools. When conversations are stored and made accessible to all members of a group, they can facilitate organizational memory, group awareness, and other beneficial practices. However, the lack of structure in chat conversations makes it difficult for users to read and keep track of lengthy conversation histories. To contend with this problem, we have developed a persistent chat system that incorporates a number of features which facilitate participation in long, ongoing conversations.

Categories and Subject Descriptors
H.5.3 [Information Interfaces and Presentation]: Group and Organization Interfaces.

General Terms
Design, Human Factors

Keywords
Chat, CMC, persistence.

1. INTRODUCTION

Group text chat has been widely found to be a useful CMC medium, capable of supporting a wide variety of helpful collaborative behaviours. The advantage of chat compared to other communicative tools is its relative simplicity, and the typically informal tone which that simplicity engenders.

Several studies have also documented the usefulness of persistent or semi-persistent chat history [1,3,5,6]. In most chat tools, conversations are ephemeral, and only visible to their participants. Conversely, in a chat tool with persistence, conversations are logged and remain visible for an extended or unlimited period of time. Finding an old conversation, even if one did not participate in it, is a simple matter of scrolling through the chat history. Persistence is helpful for a variety of reasons: it creates a record of organizational knowledge for users to refer back to; it allows newcomers to quickly perceive the history and conversational style of a group; it supports a blend of asynchronous and synchronous interaction. In general, chat tools with persistence appear to facilitate longer, more complex discussions than chat tools without persistence.

However, the simplicity and informality of chat creates problems for persistence. Chat conversations typically lack the structure found in conversations that take place over other communicative tools, such as e-mail and message boards. As a result, it can be difficult to read or browse lengthy chat logs, thus limiting effective use of persistent chat.

To contend with this problem, we have created BackTalk, a persistent chat system that incorporates a variety of features that make it easier to browse, search, and keep track of lengthy conversations. One of our design goals for BackTalk has been to avoid eliminating the advantageous simplicity of the traditional chat interface. Thus, the system is intended to combine the advantages of chat with those of more structured tools, creating an environment that is capable of sustaining rich, complex conversations that are also lightweight and approachable.

2. BACKGROUND

2.1 Persistent Chat

Most commercially available chat or instant messaging applications incorporate very limited forms of persistence. The traditional chat interface consists of a textbox for typing in new messages, and a history pane that displays a chronologically sorted list of recent messages, with the newest messages at the bottom. These messages are lost when the user logs out, and each new session starts with a blank history. Some chat tools allow users to save transcripts of chat sessions as local text files, which can then be browsed and searched. Halverson [3] found that this feature facilitates recovery of useful information from old conversations, although it can sometimes be difficult to find the desired information amongst a large collection of transcripts with limited metadata.

A number of research projects have experimented with greater degrees of persistence. Erickson et al. [1] developed Babble, a chat application which stores all conversations that take place within the system, and makes them available to all participants. The authors found that this "conversation as a single document" approach supported group awareness, and helped foster an ongoing narrative of the group as the persistent conversation continuously evolved. Ribak et al. [5] developed ReachOut, a peer support tool that features fully persistent conversations with limited lifespans. In this case, persistence was found to generate additional ideas and dialogue which may not have otherwise emerged, since users were able to observe previous discussion before deciding to contribute their own thoughts. Robbins-Sponaas and Nolan [6] have made similar observations about

MOOs, which have many chat-like properties. In particular, they noted that persistence allows a blend of synchronous and asynchronous interaction, a combination that makes for a dynamic collaborative environment.

These findings indicate that persistence in chat facilitates a variety of useful collaborative practices. However, all of these practices are contingent on the ability of users to read and keep track of lengthy chat histories, which can often be difficult.

2.2 Augmenting Chat

A few research prototypes have been developed with the intention of making conversations more legible by restructuring the conversations themselves.

Vronay et al. [8] developed a “flow client” that displays chat history with information about status and timing. With this interface, users could better understand the temporal flow of the conversation, as well as the relationships between turns.

Several projects have aimed to find a way to help users distinguish between the different topics or threads in a conversation. Smith et al. [7] developed a “threaded chat” interface that closely mimics the threaded structure of interfaces for newsgroups and message boards. Geyer et al. [2] developed Chat Spaces, a persistent chat application that lists messages in two ways simultaneously: temporally, and grouped by topic.

Both [7] and [8] included thorough evaluations of their prototype interfaces. In both cases, the authors found that by radically revising the traditional chat interface, they solved some problems, but also introduced new problems. We agree that the traditional chat interface, which has remained static for the past several decades, is in need of change. At the same time, we acknowledge that over these decades, users have become thoroughly accustomed to the standard interface, and have built a variety of strategies to make the most of its idiosyncrasies [4]. Thus, we believe that any attempt to develop a revised interface should walk a fine line between altering and respecting current practices.

3. PROBLEMS WITH PERSISTENT CHAT

We will now discuss the four key ways in which the traditional chat interface fails to support persistent chat. This discussion reflects similar observations in [7] and [8].

1. *Difficulty of distinguishing between topics*. Traditional chat treats all messages equivalently. The entirety of the chat history is rendered as a simple chronological list of all messages. In the case of a persistent chat system, this can result in an extremely long list going back weeks, months or years. Over such a long period of time, it is very likely that discussion will shift amongst a variety of topics. However, there is no way to find which parts of the history contain discussion on which topics, other then by reading the entire list of messages. This limitation makes it difficult for users to locate or keep track of discussions on a particular topic of interest.

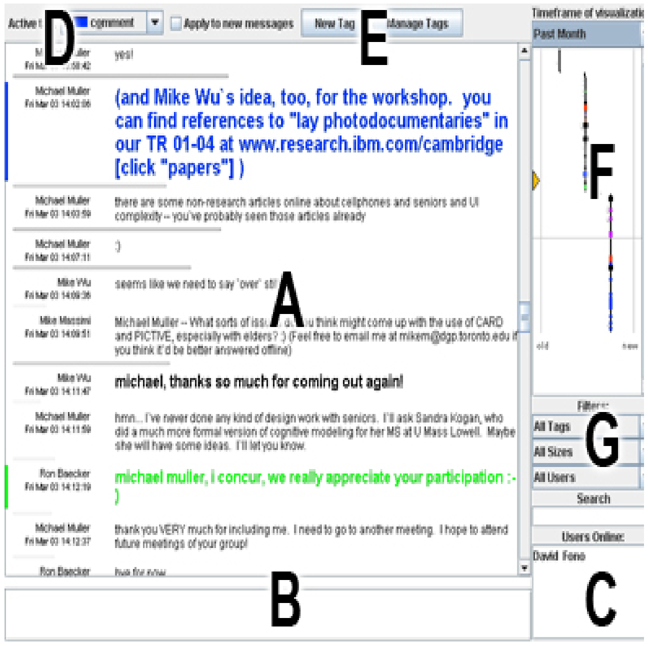
2. *High noise-to-signal ratio*. Due to the informal tone of chat, conversations often contain a great deal of socialization phrases or other messages that are not of much interest later on. However, since all messages are treated equivalently, it is difficult to discern which messages contain important information, without having to read them all. This limitation is particularly problematic for users want to review conversations they did not participate in, since they lack any foreknowledge of which sequences are important, and which can be ignored.

3. *Thread confusion*. Frequently in chat histories, replies to a message do not appear adjacent to that message. This interspersing of threads occurs because chatters often type their messages simultaneously, leading to unpredictability in ultimate ordering of these messages. The out-of-order turns that result often make chat histories difficult to read, because it is unclear which messages are replies to which other messages.

4. *Reviewing vs. chatting*. In the traditional chat interface, there is an inherent tension between reviewing old messages and keeping track of new ones. If the user scrolls the history pane to view messages higher up, then messages added to the bottom will be missed. This tension can be particularly problematic in a persistent chat interface, because asynchronous interaction will typically focus on messages higher up, and synchronous interaction will typically focus on messages at the bottom. In practice, this tension typically results in users avoiding the chat history entirely.

4. BACKTALK

BackTalk is a chat system that was developed with the intention of providing effective support for persistent conversations. The design of the BackTalk interface addresses the first two problems with persistent chat by incorporating a combination of *message annotation and message filtering*. It addresses the third problem by incorporating *threading*. Finally, it addresses the fourth problem by incorporating a *recent message popup*.



M... ...uller
Fri M... ...:58:42
yes!

Michael Muller
Fri Mar 03 14:02:06
(and Mike Wu`s idea, too, for the workshop. you can find references to "lay photodocumentaries" in our TR 01-04 at www.research.ibm.com/cambridge [click "papers"])

Michael Muller
Fri Mar 03 14:03:59
there are some non-research articles online about cellphones and seniors and UI complexity -- you've probably seen those articles already

Michael Muller
Fri Mar 03 14:07:11
:)

Mike Wu
Fri Mar 03 14:09:36
seems like we need to say 'over' sti...

Mike Massimi
Fri Mar 03 14:09:51
Michael Muller -- What sorts of iss... ...u think might come up with the use of CARD and PICTIVE, especially with elders? :) (Feel free to email me at mikem@dgp.toronto.edu if you think it'd be better answered offline)

Mike Wu
Fri Mar 03 14:11:47
michael, thanks so much for coming out again!

Michael Muller
Fri Mar 03 14:11:59
hmn... I've never done any kind of design work with seniors. I'll ask Sandra Kogan, who did a much more formal version of cognitive modeling for her MS at U Mass Lowell. Maybe she will have some ideas. I'll let you know.

Ron Baecker
Fri Mar 03 14:12:19
michael muller, I concur, we really appreciate your participation :-)

Michael Muller
Fri Mar 03 14:12:37
thank you VERY much for including me. I need to go to another meeting. I hope to attend future meetings of your group!

Ron Baecker
bye for now

Figure 1. The BackTalk interface.

4.1 System Overview

The BackTalk interface is shown in Figure 1. Like the traditional chat interface, its primary features are a textbox for message entry (B) and a scrollable history pane that shows the chat history as a chronological list of messages (A). Each message is preceded by a header that contains the timestamp of the message, and the name of the poster. New messages are added to the bottom of the history pane, and older messages can be viewed by scrolling up. All messages are accessible to all users, at all times. A box in the corner of the interface shows which users are currently logged into the system (C).

Since information about the temporal flow of a conversation can be helpful when reviewing that conversation [8], consecutive messages are separated by lines that indicate the amount of time that passed in between the two postings. The absence of a line indicates a separation of a few seconds; a short, translucent line indicates a separation of a few minutes; and a full, opaque line across the entire pane indicates a separation of an hour or more.

The history pane also shows a marker indicating the point in the history at which the user last logged out. When the user logs back in, the history pane automatically scrolls to this point. As a result, the user can easily determine which messages are new.

Clicking on a message brings up a console window, which offers various options for manipulating the selected message (Figure 2). Multiple messages can be selected and manipulated together by clicking and dragging the cursor across several messages before releasing the mouse button.

4.2 Message Annotation

Users can annotate messages in the chat history by *tagging* them or *resizing* them. Any user can annotate any message, and all annotations are visible to all users. Together, these two forms of annotation offer a quick way for users to visually distinguish messages with different kinds of content while browsing the chat history. Different tags can be used to indicate different topics, while different sizes can be used to indicate different levels of importance. Since annotation requires ongoing user effort, we sought to make the process of annotation as quick and unobtrusive as possible.

4.2.1 Tagging Messages

A tag consists of a combination of a colour and a name. A user tags a message or a series of messages by bringing up the message console, then clicking the "Click to tag" box. This action causes the selected messages to be tagged with the active tag, which can be selected from a dropbox either on the console, or above the history pane (D).

When a message in the history pane is tagged, the colour of its text changes to the colour of the corresponding tag, and a mark of the same colour appears to the left of the message. Untagged messages have black text. Once a message is tagged, that tag can be deleted or replaced from the console, or additional tags can be added. When additional tags are added to a message, the colour of its text does not change, but additional coloured marks appear to the left of the message. A variety of messages with different tags can be seen in Figure 1.

Tags can be created, renamed, or deleted by clicking on the appropriate buttons above the history pane (E).

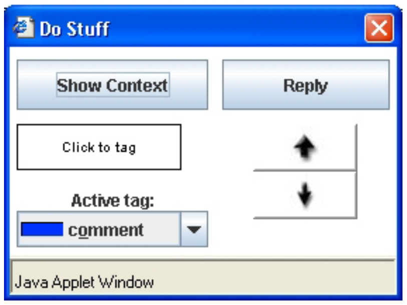


Figure 2. The message console. Clicking on a message or series of messages brings up this console, which contains options for message manipulation.

4.2.2 Resizing Messages

A user resizes a message or a series of messages by bringing up the message console, then clicking the up or down arrows. There are three possible sizes for a message. By default, all messages are created with the smallest size. A variety of messages with different sizes can be seen in Figure 1.

4.2.3 Annotation While Typing

A user can annotate a new message while typing it. To apply a tag, the user holds the Control key, and types the letter that is underlined in the name of the tag. To change the size of the message, the user holds the Control key and presses the up or down key. The colour and size of the text being typed change to reflect these annotations.

4.2.4 Visualization

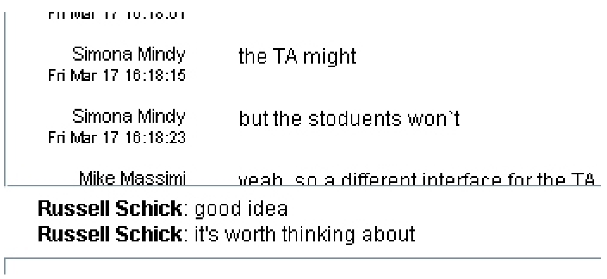
A visualization component to the right of the history pane (F) gives an at-a-glance overview of annotations across a large period of the chat history. Each message is represented in the visualization as a dot, the size and colour of which indicate how that message has been annotated. The messages are ordered along the vertical axis according to their ordering in the history pane. The position of the message along the horizontal axis indicates how old that message is; the scale of this axis can be adjusted by selecting different timeframes from the dropbox above the visualization. The user can click and drag within the visualization to quickly scroll the history pane to specified points in the history.

4.3 Message Filtering

A user can control the kinds of messages that are shown in the chat history by using a set of filters. By using the filter dropboxes to the right of the history pane (G), the user can view only messages that have a certain tag, have a certain size, or are from a certain user. The user can also view messages that contain a specific phrase by entering it into the "Search" box below the dropboxes.

These filters allow users to focus their attention specifically on messages that are of interest to them, without being distracted by the frequent sequences of irrelevant messages that populate a lengthy chat history. In particular, if messages have been properly annotated, a user can easily concentrate only on important messages, or on messages corresponding to a particular topic.

When viewing a filtered list of messages, the user may want to see a particular message within the context of the surrounding, unfiltered conversation. To do so, the user brings up the message



Simona Mindy
Fri Mar 17 16:18:15
the TA might

Simona Mindy
Fri Mar 17 16:18:23
but the stoduents won`t

Mike Massimi
yeah, so a different interface for the TA

Russell Schick: good idea
Russell Schick: it's worth thinking about

Figure 3. The recent message popup. When the user has scrolled away from the bottom of the history pane, new messages appear temporarily in the popup.

console, and clicks the "View Context" button. This action deactivates all filters, and keeps the history pane focused on the selected message.

4.4 Threading

A user can post a threaded reply to a message in the history pane. To do so, the user brings up the message console, clicks the "Reply" button, and types a reply into textbox that appears, pressing Enter when done. The reply appears directly below the parent message, and slightly indented, similarly to other tools which support threaded discussion. Proper use of threaded replies prevents the thread confusion discussed above.

This functionality is similar to the "threaded chat" interface discussed in [7]. However, in the evaluation of that system, the authors found that users frequently found it difficult to follow the discussion. Rather than appearing sequentially at the bottom of the screen, new messages would appear at unpredictable locations all over the screen. BackTalk remedies this problem by displaying new replies *both* in the appropriate thread, and at the bottom of the history pane. The copy of the message that appears at the bottom of the history has an icon alongside it, in order to distinguish it from other new messages that are not replies. Clicking on the icon causes the history pane to scroll to the thread in which the reply was posted.

4.5 Recent Message Popup

When a user has scrolled up in the history pane, and new messages are posted to the bottom, a popup with these messages appears below the pane (Figure 3). Thus, a user can focus on old messages without losing track of new ones. To avoid taking too much screen space away from the history pane, the popup shows only the 5 most recent messages, and messages disappear from the popup after 10 seconds.

5. INITIAL USER FEEDBACK

We have been using BackTalk within our research group for one month. Based on this experience, a few initial observations about the system have emerged.

First, the current design seems to offer a graceful learning curve for users accustomed to the traditional chat interface. Previous iterations of the design placed the widgets for message manipulation within the history pane, which caused confusion amongst users. As discussed, many users are highly accustomed to the traditional interface, and even minor modifications can cause consternation. On the other hand, moving these widgets to a separate console window allowed users to approach BackTalk as they approached their usual chat tools, and allowed them to discover and experiment with BackTalk's advanced functionality at their own pace.

The main problem with our usage of BackTalk has thus far been a lack of consistent annotation. We expected this issue when designing the system, and tried to keep the annotation feature simple and straightforward. However, the problem does not seem to have stemmed from issues with the interface, but rather from a lack of shared understanding amongst the group about proper annotation practices. It is possible that the system should offer some degree of codification of proper practices, sacrificing flexibility for consistency. However, we have been using the system for a relatively short period, and it is likely that more time is necessary before a set of norms around annotation can emerge.

We are currently in the process of evaluating BackTalk more rigorously, using a combination of field studies and laboratory experiments.

6. CONCLUSION

Persistence of conversations has been found to be a useful feature in group chat tools. However, the lack of structure in chat conversations makes it difficult for users to read and keep track of lengthy conversation histories. Our persistent chat system, BackTalk, address this problem by incorporating a number of features which facilitate participation in long, ongoing conversations. At the same time, BackTalk's interface maintains many similarities to the traditional chat interface, and so many of the advantages of the traditional interface are preserved.

7. REFERENCES

[1] Erickson, T, Smith, D.N., Kellogg, W.A., Laff, M., Richards, J.T., & Bradner, E. (1999). Socially translucent conversations: social proxies, persistent conversation, and the design of "Babble". *Proceedings of CHI 1999*, 72-79.

[2] Geyer, W., Witt, A., Wilcox, E., Muller, M., Kerr, B., Brownholtz, B., & Millen, D. (2004). Chat spaces. *Proceedings of DIS 2004*, 333-336

[3] Halverson, C.A. (2004). The value of persistence: a study of the creation, ordering and use of conversation archives by a knowledge worker. *Proceedings of HICSS 2004*, 108-117.

[4] Herring, S. (1999). Interaction coherence in CMC. *Journal of Computer Mediated Communication, 4*, 4 (June 1999).

[5] Ribak, A., Jacovi, M., & Soroka, V. (2002). "Ask before you search": peer support and community building with ReachOut. *Proceedings of CSCW 2002*, 126-135.

[6] Robbins-Sponaas, R.J., & Nolan, J. (2005). MOOs: polysynchronous collaborative virtual environments. *Workplace Internet-Based Communication: Industry and Academic Perspective*, Idea Group, 130-155.

[7] Smith, M., Cadiz, J.J., & Burkhalter, B. (2000). Conversation trees and threaded chats. *Proceedings of CSCW 2000*, 97-105.

[8] Vronay, D., Smith, M., & Drucker, S. (1999). Alternative interfaces for chat. *Proceeds of UIST 1999*. ACM Press, 19-26.