ANCHORED CONVERSATIONS: CHATTING IN THE CONTEXT OF A DOCUMENT

Elizabeth F. Churchill, Jonathan Trevor, Sara Bly, Les Nelson, Davor Cubranic[†]

FX Palo Alto Laboratory Inc.
3400 Hillview Avenue
Palo Alto, CA 94304, USA
{churchill, nelson, trevor}@pal.xerox.com

Sara Bly Consulting
24511 NW Moreland Road
North Plains, OR 97133, USA
sara_bly@acm.org

ABSTRACT
This paper describes an application-independent tool called Anchored Conversations that brings together text-based conversations and documents. The design of Anchored Conversations is based on our observations of the use of documents and text chats in collaborative settings. We observed that chat spaces support work conversations, but they do not allow the close integration of conversations with work documents that can be seen when people are working together face-to-face. Anchored Conversations directly addresses this problem by allowing text chats to be anchored into documents. Anchored Conversations also facilitates document sharing; accepting an invitation to an anchored conversation results in the document being automatically uploaded. In addition, Anchored Conversations provides support for review, catch-up and asynchronous communications through a database. In this paper we describe motivating fieldwork, the design of Anchored Conversations, a scenario of use, and some preliminary results from a user study.

Keywords
Text-based chat, sticky chats, collaboration, conversations, CSCW, shared documents, synchronous communication, asynchronous communication

INTRODUCTION
The past few years have seen the development of a number of computationally lightweight text-chat systems that support synchronous and asynchronous communications between individuals and groups. Examples include Internet Relay Chat (IRC) and AOL Instant Messenger [1]. Once chat software of this kind is installed, initiating contact with others is easy, taking only a few keystrokes. By creating 'buddy lists' of regular contacts, starting a chat is even simpler – selecting a user-name initiates contact. Therefore, ongoing and frequent informal interactions are easily supported. It is also possible to have multiple simultaneous conversations with different individuals and groups by opening more than one chat window [3,5].

In many chat spaces, conversations are ephemeral, lasting only while the current conversation or session is alive. However, some applications and systems allow text-based conversations to be recorded so that they have persistence. Such text logs enable asynchronous messaging between collaborators who are not online at the same time, and can also be used for catch-up and review [2, 3, 5, 6, 16]. Much recent work has also focused on providing more social activity cues in the interfaces to such chat environments [6, 16].

The success of these chat tools is not entirely surprising – people like to talk and maintain contact with others and chat systems support a form of quick, lightweight, informal communication. These are not just important for recreational activities - in the workplace, informal quick conversations and exchanges are important for maintaining the social fabric as well as for specific information exchange. Undoubtedly, the success of messenger applications and chat spaces is also due to their being computationally and cognitively lightweight: they take little time and effort to set up, little processing power to run, and little maintenance to keep going. Yet they support rich, informal, ongoing contact whenever users require. Most people don't even have to stray from their offices – everything happens on their desktops.

Problems With Chats At Work
Although chat tools are proving very useful in the workplace for maintaining contacts and exchanging information, they do not support the close integration of documents with ongoing conversations. Many work collaborations involve the sharing of documents: Web pages, presentations, spreadsheets and word processed documents. However, text conversations tend to occur in windows that are separate from, and have no relationship to, work-related resources. Figure 1 illustrates this point. In

[†]Current address: Dept. of Computer Science, University of British Columbia, 20-2366 Main Mall, Vancouver, BC, Canada. Email: cubranic@cs.ubc.ca

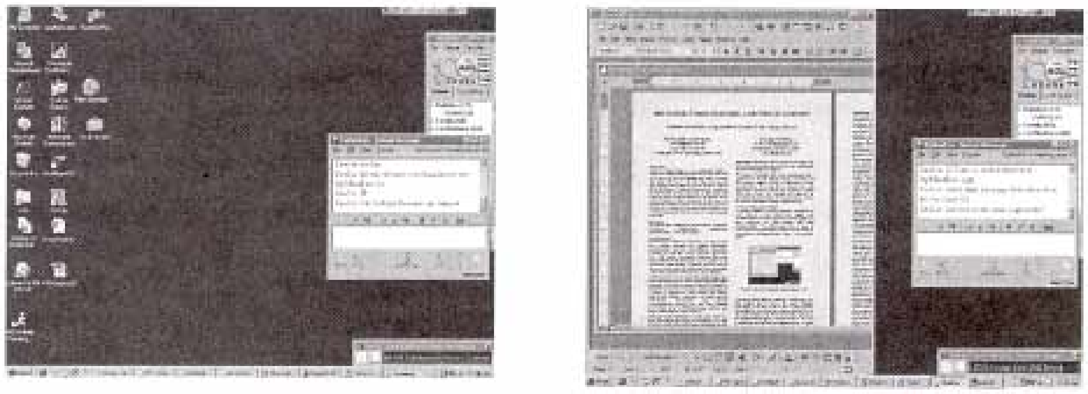


Figure 1. Chatting in messenger windows.
On the left a text chat is ongoing; on the right the word processor document that is being discussed has been opened on the desktop. Conversations and documents have no relationship

the first image a chat window is open on the desktop. In the second image, the chat window is shown alongside a word processor document that constitutes the subject of the conversation.

There are several problems with this arrangement. First sharing or copying the document that is to be discussed requires that users carry out extra steps using other applications (e.g. ftp the file, email an attachment, browse a shared repository, set up a shared window, etc). Secondly, when documents are sent or retrieved, issues arise around ensuring that all collaborators have the *same* document. Thirdly, once all collaborators have copies of the same document, establishing shared reference by navigating to the same portion of the document can be time consuming. Discussions about the content of the document require that navigation statements be typed into the chat window ("Look at section three, paragraph 2") or that the relevant materials from the document be copied and pasted into the text-chat window. In the case of shared windows, navigation cues in the form of shared pointers may be available, but shared windows tend to result in restrictions on what users can do with the content of the file itself and do not support asynchronous communications.

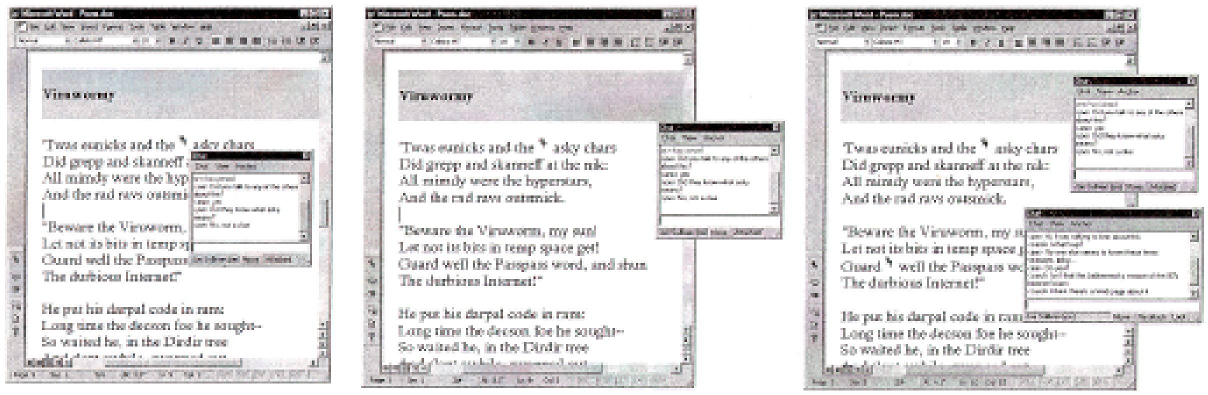
What we know about many collaboration situations is at odds with the technology affordances in this instance. Collaborators who are collocated and working in a "tightly coupled" way over those documents often converse about the details of documents: lines of text, figures or cells in a spreadsheet. Conversations in such collaborations have been characterized as "object laden" [7]; there is a high level of focus on the object or artifact that is under discussion and/or that is being co-constructed. Discussions are facilitated by (1) knowing which section is currently under discussion, (2) seeing the broader context in which the section is located, and (3) negotiating a shared visions of what is required and who will carry it out. Such conversations, therefore, crucially depend on shared context, i.e. that collaborators all have visual access to the artifact or object under construction or discussion. Here, "the object leads and language follows" [7].

FIELD WORK: CONVERSATIONS OVER WORK DOCUMENTS

Our intuitions about work collaborations were further developed in a series of field observations and interviews [3]. Our analyses focused on tightly-coupled collaborative work between non-collocated colleagues in a number of domains including software research, nuclear fusion experiments, geology field studies and after-sales software support. In each of these domains we noted issues arising around the sharing of artifacts.

In the first domain, software research, collaborators were using a text-based MUD to keep in touch and work together. Here, the tendency was to share files through email or by consulting shared file repositories. Specific details were shared by pasting text into the chat window and giving navigation cues like "Section 3, line 4" [5]. As well as being somewhat unwieldy and involving many steps, this copy-and-paste practice had the side-effect of taking the pasted-in material out of its local context. In the second domain, nuclear fusion experiments, problems arose around the sharing of experimental results in graphical and textual form. In the third domain, geology field experiments, we noted the need to support discussion around numerical data in a spreadsheet and graphs. In the final domain, after-sales software support, we noted the need to support collaborative problem solving over on-line software manuals.

In each of these cases problems arose around the establishment of shared context for conversations. The communication media (e.g. email, telephone and text-chat) were separate from documents under discussion and explicit navigation cues were required to literally "get everyone on the same page".



Viruswormy

'Twas eunicks and the asky chars
Did grepp and skanneff at the nik:
All mimdy were the hyperstars,
And the rad ravs outmnick.

"Beware the Viruswormy, my sun!
Let not its bits in temp space get!
Guard well the Passpass word, and shun
The durbious Internet!"

He put his darpal code in ram:
Long time the decoon foe he sought--
So waited he, in the Dirdir tree

Figure 2: Chatting over a document using Anchored Conversations
Chat windows are anchored into word processor documents. Anchor points are represented as pushpins. There are no restrictions on the number of chat windows that can be anchored into a document.

ANCHORED CONVERSATIONS

The Anchored Conversations tool directly addresses the issue of coupling conversations and work artifacts when working remotely. Like chat spaces, the Anchored Conversations tool is computationally and cognitively lightweight, supporting synchronous and asynchronous communications. However, Anchored Conversations also allows text-based conversations to be coupled with the documents that provide the context for work discussions (e.g. word processor documents, presentation files, spreadsheet files, graphical simulations, etc). The user model is that of "sticky chats" - adhesive chat windows that can be stuck to documents for in-context conversations, much as a sticky note can be affixed to a printed document.

Although our examples in the following sections are all grounded in word processor documents, Anchored Conversations is application independent. Sticky chat windows can be placed into any application document, and can be moved between different application documents, just as a sticky note may be moved from one type of printed document to another.

Below we detail the features of Anchored Conversations. First we describe the "sticky chat" windows in which conversations take place. Then we describe how the Anchored Conversations tool supports document transport and sharing. Here we also describe how Anchored Conversations solves the problem of establishing and maintaining shared context.

Chat windows in documents

In Figure 2 we show the use of Anchored Conversations to support synchronous discussions over a text document. In the first image at the left of Figure 2, a text editor is shown with a document displayed. In the document is an anchored chat space, a "sticky chat" window. This is a standard chat space having a space for typing text, a window for viewing the ongoing conversation and a scroll bar at the right for viewing things that have already been said. The window can be resized easily. Again, as is standard, people's names are shown in the angled brackets at the left of the chat window.

The small pushpin icon to the top left of the chat window before the word "asky" indicates the location at which the chat is anchored - or, to put it another way, the context for the conversation. The chat space is literally *anchored* to the point where the pushpin is inserted. If the user scrolls the document, the chat window scrolls with it. It may even go off screen. On scrolling back, the sticky chat window will reappear, still attached to its anchor point. Similarly, if the user drags the application window around the desktop, the sticky chat window will stick to the application window at that location and move around with it.

Sticky chat windows may be stuck to the anchor as described above, stuck near the anchor or detached from the anchor. Thus, a sticky chat window's spatial location in relation to the anchor can be altered to prevent occlusion of the material in the body of the document.

In the second image in the center of Figure 2, the sticky chat window is shown having been moved to the right of its anchor point in the document in order to uncover the previously occluded text. The sticky chat window has been "locked" into this new location. The chat window is still attached to its anchor and will remain in this spatial relation to the anchor unless moved and again "locked" to a new location. So, if the user scrolls the document or moves the application window, the sticky chat will move as before.

Sticky chat windows can also be detached; in the third image at the right of Figure 2, two sticky chat windows are shown. The lower window in the image at the right of Figure 2 is detached. The toggle at the bottom of the chat window indicates this, as the user has the option to "reattach". When a window is detached, if the user scrolls the document or moves the application window, the chat window will not move with it. However, the sticky chat does not "forget" its "home" or context anchor point. By simply clicking on the 'Reattach' toggle at the bottom of the sticky chat window, the chat window reattaches to its "home" anchor location.

As shown at the right of Figure 2, users can have as many sticky chat windows anchored in the document as they wish. These sticky chats could be separate conversations between different groups of people or could be conversations that are separated due to theme or topic.

It is also possible to move a sticky chat window's anchor location by placing the cursor, and then selecting and clicking on the "Move" toggle at the bottom of the chat window. The new anchor location is set and the anchor is moved to a new place in the document. The sticky chat window is now anchored at this new location. When sticky chats are moved in one document, this change propagates: if I move a sticky chat to a new location, you will see the change and your chat will move too. Sticky chats can also be moved *between* documents. This is discussed below.

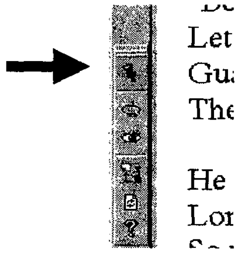


Figure 3 Pushpin Anchor Menu Item The Anchored Conversations tool adds a menu down the left side of applications. The pushpin menu item allows sticky chats to be attached to documents.

Starting conversations and sharing documents
The Anchored Conversations tool user interface consists of four parts: (1) the sticky chat window described above; (2) the pushpin that represents the point at which a sticky chat is anchored; (3) a menu bar that is added to applications and that appears down the left side of application windows (shown in Figure 3); and (4) a conversation coordinator window which interfaces to the Anchored Conversations database (shown in Figure 4).

Starting an anchored conversation requires few steps. By selecting the push-pin from the menu (shown in Figure 3) and clicking, an anchor is placed into the document at the current cursor point. Clicking on the push-pin in the document results in a sticky chat window appearing. Once the window is open, names can be selected from a "buddy" list in the menu bar. Selections result in invitations being sent automatically to invitees' desktops. This model is the same as for most messenger services. Also in accord with messenger services, once invitations are accepted, the invitees join the chat.

However, Anchored Conversations differs from other chat applications in that, on accepting an invitation to converse over a document, that document is *automatically* transported to the desktop of the invitee. The document itself opens automatically and is scrolled to the location where the sticky chat window is anchored. The sticky chat window is open and IRC available. Each person has his/her own copy of the document at this point; the sticky chat window is shared.

Anchored Conversations can also act as a simple chat application. If a chat window is not anchored within a document when created using the Conversation Coordinator, invitations result in a conversation client being opened on the desktop much as a standard chat client would. However, a chat can then be attached to a document and that document will be sent to others in the conversation.

Logging conversations and contexts
All conversations are logged in the sticky chat database (called the Conversation Database), and are available for review at any time – irrespective of whether the associated document is open. As all conversations are logged as text, searching and displaying by theme or person is trivial.

Conversation contexts are also logged. Sticky chats are not simply embedded chat objects. The "anchors" store information about the local context for the conversation. This context may be words that are nearest to the anchor, or cells in a spreadsheet – the specifics of context depend on the application in which the sticky chat is anchored. Context information of this kind is stored in the database along with information about the creation time and date of the anchor. All previous context locations in which the anchor has been inserted are also still available.

Users may query the database to see, for example, if an anchored chat has ever been located in a different place. All actions and conversations that take place within an anchored chat window are retained in the database, and are available for search, review and reuse. Even if the document in which the sticky chat was located has been deleted, it is still possible to review the conversations that were associated with the file – the filename and local context that the anchor was originally in are also preserved. Thus users can recreate conversation contexts if decisions need to be reconsidered. There is evidence that people review in this way occasionally [3].

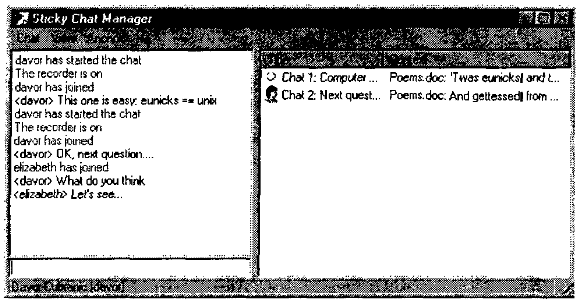
In Figure 4 we show our current interface to the database. Information about different active chats is shown (2 chats and their contexts are shown on the right of the window) as well as a log of all the current chats is shown on the left. By selecting the chats on the right, the user can navigate to a chat window in a document. We thus support our fourth goal, to provide review facilities for recreating conversations in context.

Scenario of use

In this section we present a short scenario to illustrate the use of Anchored Conversations in a distributed, collaborative work context.

Carol, Ann and Joe are colleagues working closely together on the collection and analysis of field data. They are currently preparing an executive summary of their recent work to distribute to the company management. Ann and

Joe work in the same geographic location but Carol works 4000 miles away with a 6-hour time difference.



```
davor has started the chat
The recorder is on
davor has joined
<davor> This one is easy: eunicks == unix
davor has started the chat
The recorder is on
davor has joined
<davor> OK, next question....
elizabeth has joined
<davor> What do you think
<elizabeth> Let's see...
```

```
Chat 1: Computer ...   Poems.doc: 'Twas eunicks and t...
Chat 2: Next quest...  Poems.doc: And gettessed from ...
```

Figure 4. Interface to database. The area on the left displays the text chat and the area on the right lists the sticky chats that are currently open, their names and their current contexts. The icon beside the sticky chat name shows which chats are currently active.

Last week they had a telephone conference call so that they would all have a chance to agree on the general direction of their executive summary. In addition, they have been emailing messages back and forth daily. Today, Carol is working to incorporate her analysis into the draft document that Joe sent yesterday. She likes the graph showing the data points that they've chosen to use but thinks there is an additional interpretation that should be included based on the original data. However, it will require a substantial change to the summary graph and she's not sure how to best to represent her concern. She is anxious to interact with Ann and Joe about the data and the analysis. She has already made several modifications to the document itself. She has tried to summarize her concerns about the data in an email message. However, she wants to have a conversation about it. Although she knows she could call, Joe typically works at his home in the mornings while Ann will be in the company office. They would have to arrange for a conference phone call and still might have trouble knowing exactly what point in the original data was troubling her. She decides to use the Anchored Conversations tool. In the paragraph explaining the data, she inserts a sticky chat window and invites Ann and Joe. She knows they'll be getting into work soon and they'll see the invitation to chat with her. She then continues to edit other sections of the document, inserting and carefully positioning a second anchored conversation to explain why she reorganized the Related Work section.

As soon as Carol sees activity (she can see this both in the conversation coordinator window and in the sticky chat window itself), she returns to that section of the document. Joe is there and asks why she isn't happy with the data as presented. Carol immediately tells Joe she'll show him the issue with the original data and moves the chat window to the spreadsheet with all the original data. She anchors the conversation to the graph of data points that she thinks contradict the summary data representation. As she moves the sticky chat window, Joe's document automatically scrolls to the location Carol has selected and the sticky chat window in his document reanchored.

At this point, Ann joins in as well. She reviews the conversations that have taken place – seeing what was said and where it was said. She agrees with Carol's point and starts another sticky chat in a document where she had been trying various data representations. Neither Joe nor Carol has seen this document but copies open up for each of them on their respective desktops, and all the chats automatically follow. They continue to move around the data, the report document and their conversation, arguing and discussing the possibilities for their summary.

When Carol leaves for the day, she knows that the next morning she will have another go at the document -- and that Joe and Ann will have left conversations in critical places in the document to share their thinking with her as they continued to edit the report.

Anchored Conversations allows Joe, Ann, and Carol to work together on a shared document despite their differences in time and place. Carol was able to express her concerns about the data with a clear reference to the particular place in the data at issue. They were able to converse at the same time and to leave notes for each other at different times. Placing multiple conversations in the documents allowed different conversations to point directly to different topics. Also a conversation could move as the group discussion continued; it was not held to document boundaries or type. The anchored conversation serves as a means of annotation, of chat, of pointer in both real-time and at different times.

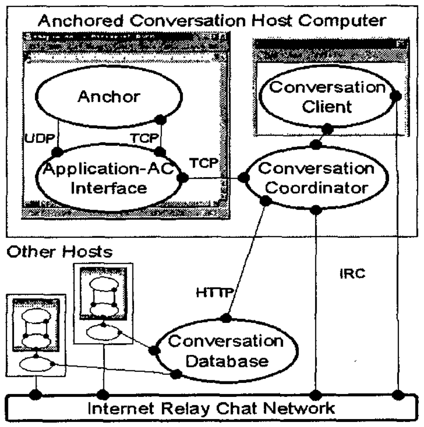


Figure 5 Architecture for Anchored Conversations

Implementation

The Anchored Conversations software design (Figure 5) consists of several components: (1) the anchor that locates a conversation within a document context; (2) the conversation client that provides the interface for one conversation; (3) the conversation coordinator that handles all conversations occurring on a host computer; (4) the

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.