NeoSoft Whiteboard - A framework for Internet-based Collaboration

Karl Lehenbauer, NeoSoft, Inc.
Brad Morrison, Paranet, Inc.
Ellyn Mustard, NeoSoft, Inc.

Abstract

Frequent, informal communication increases the likelihood and longevity of research collaborations. Close physical proximity makes collaborations more likely to be initiated, and even more likely to be sustained.

Considering the importance of physical proximity for successful collaborations, it is a wonder that Internet-based collaborations are even possible. Yet researchers are able to find each other, communicate and collaborate to produce significant results with text-based tools. We propose that the higher frequency of communication associated with close physical proximity can also be enjoyed in a networked environment, where closeness is measured by the available bandwidth for communication.

In this paper, we will show how our observations of the Internet as a groupware environment influenced the design of our own groupware environment. Our inquiries led us to develop a generic groupware framework based on Tcl and Tk, where Tcl programs are transmitted as the fundamental message type of the system. Based on canvas widgets, the framework integrates notions from active email, hypertext, and existing networking extensions available for Tcl. The Interact-4 client is described. Written to run on the groupware framework, it provides an interactive, hypergraphic groupware tool that incorporates many notions found in other groupware tools, while adding interesting new features.

KEYWORDS

real-time groupware, toolkit, development tools, hypertext

INTRODUCTION

Why build a groupware tool for developers?

Groupware is software that supports multi-user interaction and collaboration. We set out to build such a tool because we needed it. Despite the importance of physical proximity when conducting research [1], we have managed to successfully collaborate over the Internet already, and this aspect of our work is growing in ways that make the formation of new tools essential to enable us to provide maximum value for the time spent working with our collaborators.

Project size and scope are increasing, while our collaboration methods are still tied to e-mail, news, and long distance phone calls. Of course, FTP is another way to share data. Finger lets you see if your colleague is on-line; talk allows limited real-time text exchanges; and you can get an account on your colleague's machine and log in remotely to be closer to their files and environment. Sufficient bandwidth might even allow you to remote-mount filesystems using NFS.

Large numbers of successful collaborations have used these tools and little more. They are based on the least common denominator -- text. Higher bandwidth technologies have received a lot more study for groupware, yet text-based tools remain the workhorses of practical groupware on the Internet. Because they are so successful, we think they warrant further development and study.

Text remains, in fact, the most widely used medium for communication of technical information. [2] So we want to be sure to support text effectively within our whiteboards. The hypercanvas system is, in fact, a transport-independent, interactive, multi-user, formatted text and graphics user agent. Hypercanvasses can be delivered over any existing text delivery mechanism, including mail, news, FTP, and interactively, via Internet Relay Chat (IRC) [3].

A hopeful thing for the usability of the tool, at least for the target audience of developers, is that we are the end-users of the application. When developers use the things they build, those things tend to get more attention in terms of user interface, design, and maintenance.

INTERNET COLLABORATIONS-GROUPS AND TYPES

Nunamaker et al describe three kinds of groups of collaborators -- the entire group working together in the same place, groups of people in the same place working with other groups in different places, and individuals working with others in different locations. [4]

The second aspect of remote collaboration is the notion of whether the users are collaborating interactively in real time, or not. [4]

The third constraint, we believe, is content – what are the users doing? The whiteboard, while designed to be a multipurpose tool, is initially targeted to software developers, and is intended to improve project coordination and provide the framework for software design, producing project notes that eventually form the basis for project deliverable documentation.

Aside from the group organization, Internet collaborations can be differentiated by project type. A *formal collaboration* is one in which the areas of responsibility and interaction are well defined. In formal efforts one or more organizations agree on what is to be built, who is to build it, how long it should take, and any other decisions that need to be made in support of defining a manageable project and then implementing it. Formal collaborations typically include written agreements between the parties involved assigning areas of responsibility, defining milestones and project plans.

Informal collaborations, on the other hand, are done on a volunteer basis by a self-selecting group of interested individuals. Characteristically, people may join informal collaborations at any time, and they may leave at any time, even abruptly, when other time commitments or factors, such as a loss of net access, occur. Thus these self-selecting groups have a very amorphous nature. Examples of informal efforts include the netnews software, the 386BSD and Linux operating systems, and the Xfree386 X-windows for PC efforts.

Likewise there are *hybrid collaborations*, for example Tcl and Tk, where substantive, funded work is performed by some people (e.g. John Ousterhout), while others produce and contribute extensions such as the XF interface builder, expect, the photo, plot, hypertext and tree widgets, Extended Tcl, and so forth.

REQUIREMENTS FOR A SUCCESSFUL INTERNET GROUPWARE TOOL

The groupware tools need to support the lowest common denominator environment – the informal, geographically dispersed development group.

- The software must be highly portable.
- The software should be inexpensive or free.

- The software must be evolvable by its users. [5]
- The software should employ open interfaces.
- The software must be able to exploit emerging Internet groupware technologies, such as audio, video and the World-Wide Web.

PLANNING THE WHITEBOARD

Some things we decided from reading the current literature were that a pure What You See Is What I See (WYSIWIS) system would be too limited [6] -- we would make use of scrollbars to access canvasses larger than a screen, and have multiple canvasses.

Another notion from other groupware that we were quite taken with was the see-every-action approach taken by Roseman and Greenberg [7]. In their system, GroupKit, all of the users' cursors were visible and could be seen as they moved about. Because we wanted to support people with fairly low bandwidth connections (people sharing a SLIP link, for example), we would make this capability selectable, as well as providing a way to set a minimum movement per update.

Choosing Minimum Hardware Requirements

A minimum set of hardware requirements must be defined. To reach the absolutely broadest possible audience, tools would have to be text only. We felt, however, that a graphics system using the Windows-Icons-Mouse- Pointer model set a more realistic, although more elite, floor.

X-windows was selected for the display standard, since X is fairly widely available on the Internet, plus that's what Tk currently supports.

Our next consideration was the network connection. What sort of bandwidth would one need to have to use this tool? (Although Greenberg et al found bandwidth to not be the limiting factor in groupware performance [8], they were delivering over a LAN, not the Internet, where bandwidth can vary by three orders of magnitude from one site to another.)

We decided we couldn't require audio or video channels, although they are incredibly useful and to be recommended. We wanted to find an approach that would work both for people who have interactive Internet access, and for those who don't, because in our experience, there are motivated people out there with useful skills and only Email links. Successful informal collaborations have been, and are being, completed with the simplest low-bandwidth network tools.

The System Must Be Evolvable

This whiteboard system has to not just be configurable within some programmer-anticipated range of possibilities. Rather it would be user- extendible, both through customization and programming. Only then is it broadly adaptable to different organizations' approaches. Further, it permits experimentation -- a good framework should be able to accommodate broad notions of how groupware should work and should facilitate efforts to test new ideas.

Using An Embedded Programming Language

There's nothing like an embedded programming language to enable end-user customization, including by configuration and programming. When a sufficiently powerful programming language is available at runtime, uncommonly powerful environments often result. Examples noted by Ousterhout [9] include EMACS and the UNIX shell.

One of the most successful and widely available embedded programming languages is Postscript. Postscript gives laser printers and other display and/or printing technologies unprecedented flexibility in laying out pages. Postscript is routinely used in ways not anticipated by its developers. [10] Its ability to do so is its fundamental strength.

FUNCTIONAL DEFINITION

The Whiteboard is a multi-user tool that allows communication within groups of software developers who use the Internet as a groupware environment. It encompasses client/server technology and X-windows user interfaces to build a project organization and design control tool.

To maximize extensibility and force the fewest design decisions on all users and groupware developers, *basic groupware services* are provided on the client and server sides. Client and server Tcl programs that implement the actual groupware application are defined within the whiteboards themselves, and are executed on the client and server machines in the process of selecting and using a whiteboard. These programs make use of the basic groupware services, which include basic Tcl execution, communications between client and server, and the ability to create and manipulate Tk toolkit widgets. All decisions relating to user identification, cursor updates and differentiation, floor control, cutting and pasting, display slaving, and the like, are left up to the canvas-supplied client and server Tcl programs.

Basic services are provided on the client side by Tcl [9], the Tk toolkit [11], and Extended Tcl [12], in the form of an Extended Wish (wishx) interpreter, supplemented by Lehenbauer and Diekhans' open_socket code [13], Lehenbauer's secure interpreter code [14], and Tcl code written for the purpose. Tcl procedures to perform groupware services are exported into the secure interpreter by the service code.

Basic services are provided on the server side by an Extended Tcl (tcl) interpreter, supplemented by Pekka Nikander's server-side TCP/IP Tcl extensions [15], Lehenbauer's secure interpreter code, along with Tcl support code. As above, Tcl procedures to provide the basic services are exported into the secure interpreter by the service code.

Note that even the basic services can be extended by exporting new services into the secure interpreters.

THEORY OF OPERATION

The NeoSoft whiteboard system uses a central server. The data for in-use canvasses is replicated among all clients that have that canvas open. By criteria defined by Greenberg and Roseman, this is a "hybrid" system, because there is replication and a server. [8]

Client-side startup

Basic client services start up when you start the customized version of Extended Wish (wishx) and it loads in the basic client service code.

When the client is started up, the *home canvas* is automatically opened. The home canvas is specified by the HOMECANVAS environment variable or compiled-in default, and canvas ID's include the server address, service number and name of the canvas. (One possible home canvas is *hypercanvas.neosoft.com 2222 home*.)

This is essentially a standard Tcl startup, including loading the client side framework Tcl code. The "open_socket" client-side socket interface is used to handle the user's side of the TCP connection. Next a connection is established with the server, and the first whiteboard (the *home whiteboard*) is selected.

The following things occur when a whiteboard is selected:

- A secure interpreter is created in the client with its own top-level window.
- The client notifies the server, which creates its own secure interpreter and loads in the Tcl code specified for the server-side of that canvas.
- The server downloads the Tcl code making up the client side of the canvas to the client, who executes it in the secure interpreter.

From this point on, the client and server programs take over, using the basic groupware services for storage, user interface, and communications services. All authentication, permissions, notions of interface style, floor control, etc., are handled by the Tcl programs associated with the canvas, independently of the basic groupware service code.

Server-side startup

The server connects to its service number, and begins accepting connections from clients. When a client requests a connection, the server accepts it. It then awaits a message identifying the selected canvas.

When a canvas name is received, the server creates a secure interpreter and sources in the server side of the canvas, which does any startup validation and the like, then transmits a program to the client, which the client sources into a secure interpreter to get its display and user interface.

Example of server functions include:

- Authentication.
- Access lists and list management.
- Update displays to reflect arrivals and departures.
- Bring new user's display up to date.
- Implement caching with the client side.

Developers with Email Connections Only

For developers that can only access the Internet through Email connections, running the whiteboard is still possible, but the user must run the server and the client on his/her machine.

Canvas Control

We define the *canvas control portion* as the area in the interface where you choose colors, actions, etc.

Canvasses are manipulated using user interface elements defined on the canvasses themselves, and by elements defined on the control canvas.

In our first few hypercanvas iterations, we had a fixed set of controls for drawing, entering text, etc., but found that to be cumbersome -- it didn't allow for people in different roles to have unique interfaces, which is desirable [16], and it made the tool less useful as a framework for developing groupware interfaces due to its hard-coded nature.

By separating the drawing interface from the delivery mechanism and making it be an arbitrary Tcl program, all of those problems are eliminated, and individuals and organizations are free to evolve the canvas system at either end to better support their goals and methods.

There isn't necessarily a need to limit ourselves to canvasses, though. The reason to do so is consistency and that we can set up the canvas bindings in such a way that the cursors can always been seen. This way, it's even possible to draw on the control canvas, which could be quite useful for providing documentation layers, feedback during the evaluation phase, etc.

Latecomer Updates

When someone enters an active canvas, the state of the canvas is downloaded from one of the current participants, using the server.

All actions, including cursor movements by all participants, are seen on all displays that are on the same canvas and showing the same part of the canvas.

Methods Of Delivering Whiteboard Updates

You can deliver whiteboards as compound files using regular Email, news and even interactively using IRC.

It's useful to deliver interactive whiteboard changes over IRC because IRC already has a multichannel realtime text distribution system in place that has scaled to support a network of interconnected servers and thousands of interactive users. Since we're going to send these commands as text, it is no problem to transmit Tcl commands instead of user-typed in text on specifically named channels. If we decide not to deliver over it, it is still worthy of study -- the IRC developers have solved some interesting problems.

Floor Control

Floor control is a system of deciding who speaks next. While floor control is important for large groups, it is not usually considered very important for small ones [15]. Since the client and server sides of a canvas are user-definable, different notions of floor control can be experimented with, implemented and deployed, typically without requiring changes in the system framework.

Attaching Files

There are all sorts of files, important to your activity, already of a specific format that was not defined by the groupware tool. Any development effort is going to have those files, and they are going to be a topic of conversation within the whiteboard. So we need a way to point to those files that is meaningful in terms of user interaction, in other words, we want not only the name of the file, but to be able to read it, listen to it, view it, format it, compile it, etc., according to what it is. Again we are saved by the client program, because it's user-defined.

GROUPWARE CLIENTS AND SERVERS

A number of groupware client/server programs have been written for the groupware framework. They range from simple game interfaces to a working developer collaboration tool that addresses many important groupware issues.

Interact-4 Groupware Tool

The Interact-4 groupware tool is the fourth in a series of in-house attempts to create a groupware tool that operates within our framework and addresses many important groupware issues.

Greenberg et al showed the user's name along with their cursor as it moved around. We support this notion, and others, since cursor management is totally user-configurable with Tk event bindings and creating and manipulating graphical elements within the canvas. Cursor differentiation could also be done on the basis of color, or a pixmap can be displayed, perhaps the facesaver image of the participant or a Monopoly piece, to name two.

We also implemented the idea of gesturing. Roseman et al had the ability to emphasize by temporarily switching in a big cursor [7]. Other possibilities would be to substitute a different bitmap, change the color and flash the cursor, and possibly also doing something with the person's facesaver bitmap, such as flashing it.

Drawing Tool

Graphical objects in the control canvas can be selected to choose colors, draw line segments, arcs, rectangles, polygons and text in the display canvas.

Cursors Of Arrow And Name

All the active users of the canvas have their cursor position displayed, and updated in real time, on all other users' canvasses. Cursors are arrows with the user name underneath.

Conversation Window

When you don't have a phone connection, a conversation window is available where you can type messages and have them scroll in a window. Of course it's just another hypercanvas.

Local Save

Any user can save a private copy of the drawing canvas at any time.

Server Save

Authorized users can save a public copy of the drawing canvas at any time.

Hyperlinks To Other Canvasses

Hyperlink buttons can be created on any canvas that causes a jump to another canvas.

File Attachments

Files can be transmitted intact and "attached" to a canvas. Displaying or processing of the file is performed by outboard tools, such as *xv*, *ghostview*, etc.

Floor Control

Since this tool is targeted towards small groups, thus far we have gone with an open floor policy (no floor control) .

Cutting and Pasting

Cutting, copying and pasting are implemented using Tk's excellent facilities for locating, obtaining and modifying the definitions of tagged canvas items. In this code, Tk's ability to return the ID's of all of the tagged items within a bounding box is put to good use to implement cut and copy.

Scaling

Once a piece has been copied from a canvas, it can be scaled with facilities built into Tk's canvas widget, before being pasted into a different canvas.

Display Slaving

Another useful concept from existing groupware is *view slaving* [17], where view control is granted to another user, or shared with them. There the enslaved display is scrolled to the same view on whichever canvas. We could have a button that says "Jump to whatever Brad is doing." Since we have multiple windows, one canvas can be slaved while another is not.

Update Tagging

Update tagging is where you want to see what was done since some point in time. Each display item is tagged with the integer time (seconds since 1970) it was created. Items can be selected over a range of time and their attributes manipulated in some desired way to show which ones they are.

User Identification

Determining who created what items is easily done by tagging each item with its author. There are times and places where anonymous editing should be allowed, however. Examples include brainstorming sessions, where it is desirable that ideas be considered by themselves without the weight of their author being attached to them [4], and cases where the content of the canvas may demand anonymity.

Most times you need to know who wrote what, for example when working on a schedule for a software project. On non-anonymous canvases, our client and server tag each item with the *user@machinname* ID of the author. Tk can then be used to determine who an author is. We could flash one person's widgets, or perform some other attribute modification on them, as with layering.

In normal use, we show the names and facesaver images of the people in the canvas on the control canvas. How this works is up to the developer of the client and server programs.

Layers

Many CAD systems support the notion of layers. These are display elements grouped by some criteria, that can be included or excluded according to user selection. Tk's tag mechanism is perfect for this task; the client code has all it needs to provide support for separate layers.

Jumping Out to Other Applications

Since the client side is Tcl code, bindings and Tk user interface elements can cause anything to happen -- an application to be launched, local calculations to be performed, ad infinitum, subject to security constraints imposed by what services the client's trusted interpreter exports to the secure interpreter.

Data that is created outside of the hypergraphic system still needs to be accessible by that system. For example, news articles, Email, postscript and GIF files, and so forth, may need to be referenced by, and be accessible from, the hypergraphics system.

We added an optional number-of-bytes-to-copy parameter to copyfile, to facilitate in-line, in-stream file transmission. We want to transfer bitmaps for certain, ultimately the GIF and postscript files as well.

When transmitting over a text link, binary data needs to be uuencoded, or equivalent. Examples would include transmitting attachments over IRC and mail.

CONCLUSION

Groupware makes it possible for people separated by distance or time to effectively collaborate on software development.

The NeoSoft Whiteboard System, while drawing on existing groupware concepts, provides several unique capabilities that make it attractive as a framework for building hypergraphic groupware systems.

AVAILABILITY

The groupware code will be available by conference time from *ftp.neosoft.com* in the /pub/groupware directory.

ACKNOWLEDGMENTS

Mark Roseman helped us with a lot of background on groupware. Robert Cottingham was invaluable for late-night formatting. Thanks to Nick Handel for his part of the all-night, marathon programming session that launched these efforts. And thanks to Mark Diekhans for critical reading, constructive engagement, and for being our most prolific remote collaborator.

REFERENCES

[1] R. Kraut and C. Egido, *Patterns of Contact and Communication in Scientific Research Collaboration*, 1988, Association for Computing Machinery reprint, ACM 0-89791-282-9 / 88 / 0001

[2] P. Saffo, *Hot New Medium: Text*, Wired, Vol. 1, No. 2, May/June 1993, Page 48.

[3] Michael Sandrof, *Internet Relay Chat II*, FTP, from ftp.std.com, /src/network/irc/ircII2.2.1.2

[4] J. F. Nunamaker, A. R. Dennis, J. S. Valacich, D. R. Vogel and J. F. George, *Electronic Meeting Systems to Support Group Work*, Communications Of The ACM, July 1991/Vol. 34, No. 7

[5] Jonathan Grudin, *Why Groupware Applications Fail: Problems in Design and Evaluation*, Office: Technology and People, Vol. 4, No. 3 (1989), pages 245-264.

[6] M. Stefik, D. G. Bobrow, G. Foster, S. Lanning and D. Tatar, *WYSIWIS Revised: Early Experiences with Multi-user Interfaces*, ACM Transactions on Office Information Systems, Vol. 5, No. 2, April 1987, Pages 147-167.

[7] M. Roseman and S. Greenberg, GroupKit - A Groupware Toolkit for Building Real-Time Conferencing Applications

[8] Saul Greenberg, Mark Roseman, Dave Webster and Ralph Bohnet, *Human and Technical Factors of Distributed Group Drawing Tools*, Interacting With Computers, Vol. 4, No. 3 (1992), pages 364-392.

[9] John Ousterhout, *Tcl: An Embeddable Command Language*, Proceedings of the Winter 1990 USENIX Conference, January, 1989

[10] Adobe Systems, PostScript language reference manual, ISBN 0-201-18127-4

[11] John Ousterhout, *Tk: An X11 Toolkit Based on the Tcl Language*, Proceedings of the Winter 1991 USENIX Conference, January, 1990

[12] Karl Lehenbauer & Mark Diekhans, *Extended Tcl Extended command set for Tcl*, unpublished manual page, January, 1992.

[13] Karl Lehenbauer and Mark Diekhans, *open_socket: Tcl interface to TCP/IP socket library*, Usenet News, Message-ID: <1992Feb11.143549.8314@NeoSoft.com>

[14] Karl Lehenbauer, *Re: Using Tcl for active messages (source code included)*, Usenet News, Message-ID: <C5KszH.Csw@sugar.neosoft.com>

[15] Pekka Nikander, *A Simple TCP Connect for TCL/TK (source included)*, Usenet News, Message-ID: <PNR.92Mar22213722@innopoli.ajk.tele.fi>

[16] Stephen Viller, *The Group Facilitator: A CSCW Perspective*, Proceedings of the Second European Conference on Computer-Supported Cooperative Work, September 1991, pages 81-95.

[17] M. O. Pendergast, S. C. Hayne, *Assisting groups during the merging process*, Decision Sciences Conference, 1991.