SYMMETRIC SYNCHRONOUS COLLABORATIVE NAVIGATION

Luca Gerosa, Alessandra Giordani, Marco Ronchetti
Dipartimento di Informatica e Telecomunicazioni, Università di Trento
Via Sommarive 14, 38050 Povo di Trento, Italy
{lgerosa,agiordani,marco.ronchetti}@dit.unitn.it

Amy Soller
Irst-ITC
Via Sommarive 18, 38050 Povo di Trento, Italy
soller@itc.it

Ron Stevens
UCLA/IMMEX Lab,
560ı W. Slauson Avenue #255, Culver City, CA, 90230
immex_ron@hotmail.com

ABSTRACT

Synchronous collaborative navigation is a form of social navigation where users virtually share a web browser. In this paper, we present a symmetric, proxy-based architecture where each user can take the lead and guide others in visiting web sites, without the need for a special browser or other software. We show how we have applied this scheme to a problem-solving-oriented e-learning system.

KEYWORDS

Collaborative navigation, social navigation, e-learning

1. INTRODUCTION

Computers can be used to support both individual and collective experiences. Sometimes, even the same task can be performed individually or cooperatively. For instance, using a word processor to write a letter is individual task, while using it to write a paper with other authors, and making use of tools like versioning and annotations, is certainly a collaborative action. Even computer games can have individual and networked versions. For example, Doom can be used as a solitary game, or as a social one when playing over the network with friends, cooperating to defeat the (virtual) enemy. Solitary games can also have a social component, e.g. when trying to outperform other users. Simply keeping track of the high scores introduces a (indirect) social component in an otherwise individual game. We therefore have a whole spectrum of possibilities, where activities can be classified as *individual, indirect social,* and *social*. Moreover, social activities can be *asynchronous* or *synchronous*. They can also be *symmetric* or *asymmetric*: in the first case all the users play the same role (or at least have equal opportunity to do so), while in the second, a leader controls the system. These possibilities are determined by the natural affordances of the technology – those characteristics of the technology that define the ways in which users can carry out activities and interact with other users in a given context, and the ways in which the software mediates these activities (Jermann, et al., 2004). In this introduction, we take a look at a few different examples.

Navigating a hypertext or the web appears at first glance to be an individual experience, not unlike reading a book. However, even web navigation can be seen as a social activity (Erickson 1996, Dieberger 1997). For instance, *indirect social navigation* is possible in an on-line grocery store if people visiting the site are given indications about what other people have bought (Munro et al. 1999). Amazon.com actively uses

this concept for suggesting books to buy according to the behavior of other users having similar interests. Along this idea, recommendation systems have been employed and studied by many authors in the last few years. Recommendation can be explicit (like when users are allowed to express their opinion, for instance by voting on a page or writing a comment) or implicit (when the recommendation is obtained by analyzing indicators of the visitor's behavior, such as the choices taken by the users, or the time spent on a page and the followed links).

Direct social (and interactive) navigation occurs when the navigators can become active authors, like in the case of Blogs or Wiki's. In this case the navigator can actively modify portions of a Web site, mirroring the kind of interaction experienced during the cooperative activity of writing of a paper. A Wiki (WikiWikiWeb, a concept invented by Cunningham in 1995) enables documents to be authored collectively in a simple markup language using a web browser (Leuf and Cunningham 2001). Similarly, a weblog, also known as a blog, is a website which contains periodic, reverse chronologically ordered posts on a common webpage (see e.g. Stauffer 2002). Individual posts either share a particular theme, or a single or small group of authors. Because of their ability to support the notion of direct social navigation, Blogs and Wikis are growing in popularity. Interactions through blogs and wikis are in most cases *asynchronous*.

A case in which interactions among navigators are *synchronous* has been proposed and implemented in the EDUCO system (Nokelainen et al, 2002). EDUCO appears to the users as a visual collection of web sites; the users can navigate the documents and see when other users are navigating those same documents. Users can contact and chat with each other by clicking their respective dots in the EDUCO view. Furthermore, users are able to set "alarms" which are triggered when someone (e.g. anyone, or a particular person) arrives to the systems or to a certain document. (Hoppe and Ploezner 1999). The interactivity here is given by the ability to observe, in real time, the population of visitors of the environment, and to contact them explicitly. The main goal of the system was to let the user break the feeling of "loneliness" when navigating.

An even stronger form of *synchronous* interaction during navigation is *co-navigation*, i.e. allowing multiple users to share a navigation experience by synchronizing their browsers. This idea has been explored in the early days of the web. Yeh et al. suggested in 1996 that a web client (in master mode) could take control of other web clients (in consentient slave mode) to guide them through an internet tour (synchronous navigation). They argued that such an activity could be valuable in an e-learning environment, where a tutor might show domain material or learning artifacts to pupils. In the following years, similar systems have been proposed (we discuss these later).

In this paper, we realize the idea of co-navigation in not only a *synchronous*, but also *symmetric* setting. We describe the prototype we implemented, in which users can join a group, and navigate in a *collaborative*, *synchronous* and *symmetric* way. This means that all users of a collaborative learning group view the same screen on different browser, and the actions taken by any group member has a direct effect on the browsers of all the other users in the group. In our prototype, we also provide tools such as a textual chat, and a system to obtain interface control. The aim of this research was to enhance individuals' learning experiences in an online, initially single-user, problem-based learning environment by providing them the opportunity to collaborate synchronously while navigating together. In our environment, the students typically solve problems by exchanging ideas while exploring a simulated mini-world through co-navigation. In section 2 we describe the environment and our goals in more detail. In section 3 we present the overall architecture. In section 4 we discuss related work, and draw conclusions.

2. MOTIVATION OF THE WORK

Our work was motivated by the desire to extend the IMMEX™ system to support collaboration among students. The IMMEX™ (Interactive Multi-Media EXercises: www.immex.ucla.edu) software, which was developed at the University of California, Los Angeles, has been used in science classes across middle and high schools, universities, and medical schools in the U.S. over the past 12 years, and has logged over 140,000 student problem solving performances (Stevens and Palacio-Cayetano, 2003). Through the IMMEX web-based interface, students learn how to elaborate hypotheses and analyze laboratory tests while solving real-world problems. The system presents problem sets as scientific case-studies and realistic multimedia domain-specific simulations. For instance, in one problem set, students perform physical and chemical chemistry tests to determine, as quickly as possible, whether or not some chemicals that spilled during an

earthquake are dangerous. The system does not allow students to perform an exhaustive exploration of the problem space; they need to be selective and use scientific inquiry to solve the problem. A rich portfolio of over 100 problem sets in various disciplines has been developed, and is now available online. Statistics generated by the system have been used to identify the common types of strategies high school chemistry students use to solve qualitative chemistry problems (Vendlinski and Stevens, 2002), and we have now begun to study how collaboration influences students' problem solving strategies. It was therefore necessary to enhance the IMMEX system with collaboration facilities to allow students to cooperatively solve problems.

Because IMMEX is a web-based environment in which we wanted to support collaborative learners who might not be co-located, it was necessary to develop tools to support their synchronous, symmetric cooperation through the web. Important tools included a chat that allows them to discuss the problem, and their proposed actions, and facilities that enable students to cooperatively use the IMMEX system as if they were seated in front of the same computer. The advantage of mediating the students' interaction through the computer was twofold: this enables remote students to work together over the network, adding a layer of cooperation to distance learning, and it also allows the system to keep track of the actions and discussions. Such logs can be analyzed, for example, to investigate how the collaboration influences the student's learning styles.

The concept of synchronous, symmetric co-navigation satisfies these goals. We designed and implemented an architecture based on an HTTP proxy, Java and Javascript, to allow the co-navigation *on a peer basis*. We do not use the concept of a (static) master and slaves. In our implementation, any user can take control of the navigation, and the effect of each action is visible to all other associated users. Our architecture is scalable, and can be used both in the context of the IMMEX system, and, with minor adaptations, to support other environments where co-navigation may be useful.

3. THE ARCHITECTURE

In order to add extra functions on top of the web paradigm there are three possibilities: to enhance the server, to modify the client, or to interpose an additional actor (typically a proxy).

Enhancing the server is, in general, the simplest solution. Several technologies are available to achieve the goal: CGI programs, scripting languages (like ASP, PHP etc.) and Java Servlets (or their dual: Java Server Pages). Such solution can be applied whenever the desired functionality fully depends on a single Web server, and when one has full control of the web application that has to be modified.

Some degree of customization of the behavior of a client (i.e. of the web browser) is possible by using client-side scripting languages (Javascript being the main player in this field) or by using applets. Such approaches, however, require modification of the original pages coming from the server, so that ultimately, one does have to act on the server side. Modifying the server functionality or the content it delivers may not be an acceptable solution in many cases. For instance, one might not have access to the server and its content, or one might rather not touch it.

More radical browser behavior modifications can be obtained by building an ad-hoc browser that incorporates the desired functionalities. The modified client can then talk to the standard "content providers" (i.e. normal web servers) via HTTP, and to a special purpose server providing the needed functionality using an arbitrary protocol. Such an approach is viable, e.g. in Java some APIs provide the basic building blocks (like the class JEditorPane) to build a rudimentary basic browser with only a few lines of code. The main drawbacks are, however, that building a full fledged browser is a gigantic task, and moreover having to utilize a special browser in some context, and a standard browser in others, can be very inconvenient for the user.

The last possibility is to encapsulate the desired business logic in an active element interposed between client and server. The browser must address its requests to the middle layer, that in turn, pushes them to the server, gets the response, and before delivering it to the browser, performs whatever modifications are needed to add the desired functionalities. The simplest example of such an architecture are the widely employed proxy servers, where the middle layer stores a local copy of any page that is requested so that the successive requests (of the same or of other users) can be fulfilled quickly without downloading the content again from the server. From now on we call "proxies" such middle layers, although they can do much more than just caching pages. For instance, a proxy can actively modify the served pages. Such modifications may include,

for instance, URL rewriting, content filtering, triggering of events, and addition of Javascript code to allow a limited control of the browser behavior. Such schemas can be added to implement a number of new functionalities (e.g. to allow adding personal annotations on a HTML page, see Ronchetti 2002). Moreover, it is possible to employ a cascade of proxies, each of them providing a specialized service. In some cases a proxy adds a (typically small) delay to the delivery of an HTTP request and response, while in others (like in the case of caching) they can actually improve overall performance.

In our case we wanted our general architecture to scale beyond the IMMEX environment. Additionally, we needed the IMMEX server and its content to be unaffected by the collaborative features, and to continue to operate for individual users in the standard way. Both of these requirements ruled out the first of the above discussed options. The option of using a custom browser was never taken into consideration for the reasons already mentioned. The obvious choice was therefore a proxy-based solution, augmented by browser control, embedded in applets and Javascript that are dynamically added from the proxy to the pages obtained from the web server.

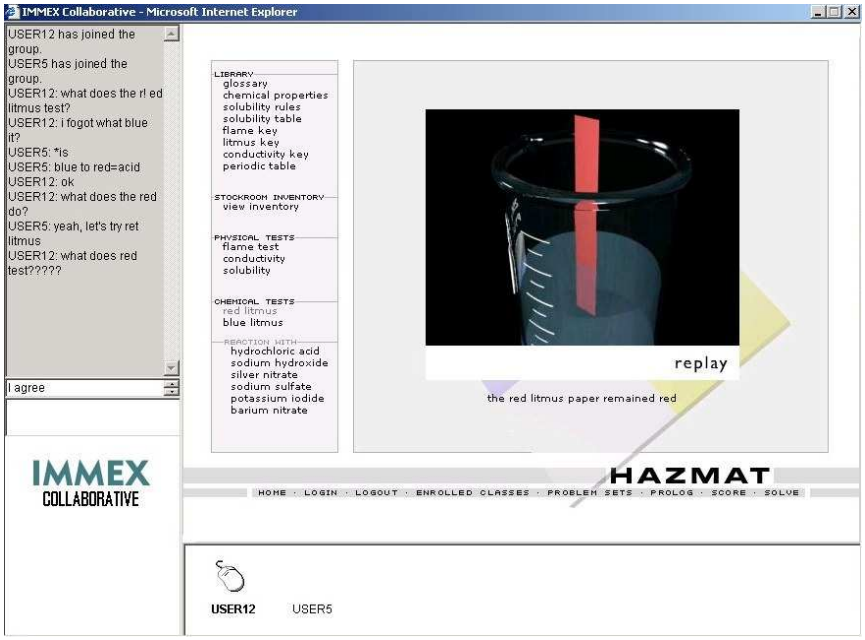

Figure 1. The IMMEX Collaborative user interface

In order to describe our architecture, it is useful to follow a use-case scenario. Therefore we shall now describe step-by-step what the user perceives, and the architectural details that make each action possible. The first scenario is group formation. Because co-navigation means dealing with more than one actor, we define a group as two or more actors that intend to co-navigate. A user begins forming a group by requesting a login web page from the proxy (that in this case acts as a web server). Other users may join this group using the same page. We consider several different logics for group formation: either the composition of a group is predefined, or groups can be dynamically formed "on the fly". Also, the process of group formation can be considered complete when the group size has reached a given dimension, when all the predefined users have joined, or when the user who started the group decides that the process is over. At present we use groups composed of two persons, but the reasons for such limitation are given by the goals of the collaboration experiments we intend to perform, and not by technical reasons. During the group formation, the proxy acts as a normal web server, interacting with a database in which it logs the group information. The proxy completes the user login procedure by delivering a page with frames: two of the frames contain applets. The first applet implements a textual chat. The chat server is forked on the proxy. From this point on, and for the entire time of the co-navigation, users can interact using the chat (see Figure 1). A second applet, that we will call the control applet, is not visible to the user. It has the important duty of controlling the browser. It opens

a socket connection to the proxy, on which a second daemon is forked. This daemon is the core of the co-browsing process – we call it the control engine. Note that this socket connection is open on port 80, so that the system can also work through firewalls.

Once the group formation is complete, the proxy impersonates a browser, and starts a session with IMMEX (in the general case, it accesses the web page that the first user of the group defines as the navigation starting point.). From the server's point of view, the proxy is just a generic client. The server delivers a page to the proxy as the HTTP response. When the proxy receives the page, it sends a message to each of the group member's control applets. The message contains the (local) reference for the page. The control applets talk to the browser using their AppletContext class (a standard Java class), requesting that the browser load the referenced page in the main frame. The browser then contacts the proxy and requests the page, which the proxy delivers. At this point all browsers in the group will show the same initial page.

Figure 1 shows what the user sees at the end of this process: the largest frame contains the page that was originally obtained from the Web server (in the figure the "HAZMAT" chemistry problem). The gray frame on the left contains the chat. The static frame in the lower left-hand corner with the title, "IMMEX Collaborative" hides the control applet. The frame along the bottom of the window showing a mouse contains another applet, "the mouse applet", which enables users to manage the synchronization of the application (described in more detail later).

The second scenario takes off from where the first scenario ends. In this scenario, all the users in the group see the same page, and can interact through the chat. At this point a user might want to navigate by following one of the links that are present on the page. When s/he clicks, a new request is sent to the server from which the page originated. For the browser, this server is the proxy (there also exists a caveat regarding absolute links, which we discuss later). The proxy gets the request, and can process it similarly to the case of the first page: it forwards the request to the web server and gets in return a page. This time, however, not all the group members' browsers are in the same state. One of them (the one which originated the request) is waiting for an HTTP response, while the others are idle. The proxy then delivers the page to the waiting browser, and contacts the control applets of the remaining browsers to solicit a request for the fresh page to be delivered. At this point all the browsers are once more in the same state, and the scenario repeats.

We experienced a potential problem stemming from absolute links, in that when the browser saw an absolute link, it bypassed the proxy, and directly accessed the specified server. Such problems can be avoided in two ways. The first is to ask the user to modify the browser preferences; every browser allows the address of the Internet proxy to be defined. If that address points to the proxy, then all requests (including the absolute links) can be intercepted. The other possibility is to let the proxy parse all HTML pages that it serves, and have it re-write all the URLs. Although this is possible, it introduces a small additional delay in serving the pages. In the case of IMMEX we have the guarantee that all the needed links are relative, so we can avoid dealing explicitly with the problem. For the more general case, we apply URL-rewriting because it is a safer approach, since it does not rely on user's preferences.

Until now, we have deliberately ignored a typical problem of parallel code execution: concurrency control. What happens if a second user clicks on a link right after the first user, before the serving of the new page is complete? Clearly, such events must be avoided. Other approaches to co-navigation (see the discussion in section 4) use a master-slave model, meaning that they assign different roles to different users. In particular, there is one (and only one) privileged user who holds the bar, while the others just "look". Because we wanted to provide students with a synchronous, symmetric environment, approaches like this were unsatisfactory. We solved the problem by using the mouse applet and introducing the concept of a token. Only when a user has a token, his/her clicks are effective. The presence of the token is pictorially shown by representing the mouse above the user name in the mouse applet. A user can ask for a token by clicking on the mouse: in this case the mouse applet talks to a daemon on the server, and puts the user on a queue. When a user gets to the top of the queue, all the mouse applets are informed that the token has been given to the user and update their view so that everybody is aware of who is in control at any given time. To prevent the user from clicking on links when s/he is not the mouse owner, we use the Javascript onClick function, which

in our implementation checks the ownership of the token. If the token is not available, the actions do not have any effect.

4. RELATED WORK

As we mentioned in the introduction, attempts to provide synchronous co-navigation date back to the early days of the web. One approach was to build a special browser that provided ad-hoc functionalities, such as peer-to-peer interaction, or that provided special views or interacted with dedicated server through a protocol. Among such systems are Albatross (Yeh et al. 1996), GroupWeb (Greenberg and Roseman, 1996), Nestor (Zeiliger 1998) and Co-Vitesse (Laurillau 1999). Building a special purpose browser, however, has strong drawbacks. For example, users may desire to use their own preferred browser, and using different browsers for different tasks may be rather unnatural. Also, implementing a browser that can replace a commercial browser, providing support for Java, Javascript, and plug-ins of various kinds, can be a daunting task. Such an approach should therefore be restricted to ad-hoc tools for demonstration purposes.

Support for synchronous navigation using a standard browser was illustrated by Cabri et al. (1997), using an architecture similar to the one presented here. However, Cabri and colleagues' work is mostly aimed at cache optimization for workgroups. Their application enables members of the workgroup to be aware of each other's activities, hence avoiding duplicate searches. It also facilitates communication directly through the browser, and benefits from the typical proxy advantages such as saving time when getting pages already retrieved by others. Cabri and colleagues only briefly mention their synchronous navigation, which employs a master-slave model.

Esenther (2002) describes a *symmetric* implementation of co-browsing. Although the system architecture is not described in great detail, it seems as if the implementation is based on pure Javascript, uses a dedicated server, requires a Microsoft browser, and is mostly suited in an intranet. Another symmetric implementation, JASBER, is reported by de Oliveira et al. (2003). In this case, only Netscape browsers are supported, whereas our implementation does not rely on a particular browser.

The work by Aneros et al (2003) contains an interesting discussion of various co-browsing models, although their work is concerned with providing a persistent shared unified history object that allows cooperative browsing in a broader sense (not as co-navigation).

5. CONCLUSION

In this paper, we presented a symmetric implementation of co-navigation, in which collaborative learners can join groups and collectively visit web sites, while their views are synchronized. A simple concurrency control system allows any learner to take the lead of the group navigation at any time. Users can also interact through a chat. No special tools are needed on the user side – only a standard web browser. All of the controlling logic is kept on a proxy server, so that the navigation is not restricted to specially enabled web servers.

The idea of co-navigation is not new. It is the symmetric role of users coupled with the use of a standard browser, and the independence on the server-side that makes our approach novel. The idea of co-navigation has also been previously proposed as a technological enhancement of the web paradigm, without a strongly motivated application or precise user need. In our case, the IMMEX collaborative problem solving environment provides a context in which our synchronous, symmetric co-navigation solution can be used to naturally support and enhance students' problem solving activities.

ACKNOWLEDGEMENT

This work is supported by NSF under grant NSF ROLE 0231995.

REFERENCES

Aneiros, M., Estivill-Castro, V., Sun C., 2003, Group Unified Histories an Instrument for Productive Unconstrained Co-Browsing in *GROUP'03*, November 9–12, 2003, Sanibel Island, Florida, USA.

Cabri G., Leonardi L., Zambonelli F,. 1999, Supporting Cooperative WWW Browsing: a Proxy-based approach. *7thEuromicro Workshop on Parallel and Distributed Processing*, Madeira, Portugal, pp 138-145

de Oliveira, J.C., Hosseini, M., Shirmohammadi, S., El Saddik, A., Malric, F., Nourian, S., Georganas, N. D., 2003, Java Multimedia Telecollaboration, *IEEE Multimedia*, Volume 10, Number 3, pp. 18-29

Dieberger, Andreas, 1997, Supporting Social Navigation on the World-Wide Web, in *International Journal of Human Computer Studies*, Vol. 46, pp 805 – 825

Erickson, T. (1996). "The World Wide Web as Social Hypertext." *Communications of the ACM* 39(1): 15-17. .

Esenther, A. W., 2002, Instant co-browsing: Lightweight real-time collaborative Web browsing. In *Proc. Of the 11th Int. WWW Conference*, May 2002, pages 107–114, Honolulu, Hawaii, USA

Greenberg, S. and Roseman, M. 1996. Groupweb: A WWW browser as real time groupware. In *Human Factors in Computing Systems, CHI Companion Proc.*, ACM Press, pp. 271–272..

Hoppe, U. & Ploezner, R. 1999. Can Analytic Models Support Learning in Groups? in P. Dillenbourg (Ed.), *Collaborative-learning: Cognitive and Computational Approaches*, pp.147-168. Elsevier, Oxford, UK.

Jermann, P., Soller, A., & Lesgold, A. (2004). Computer software support for CSCL. In P. Dillenbourg (Series Ed.) & J. W. Strijbos, P. A. Kirschner & R. L. Martens (Vol. Eds.), *Computer-supported collaborative learning: Vol 3. What we know about CSCL ... and implementing it in higher education*, pp. 141-166. Boston, MA: Kluwer Academic Publishers

Laurillau. Y., 1999, Synchronous collaborative navigation on the WWW. In *Proc. of CHI Conference 1999*, 15–20 May 1999, Pittsburgh, PA, USA, pp. 107–114

Leuf B., Cunningham W., 2001, *The Wiki Way: Quick collaboration on the Web*, Addison-Wesley Longmann

Munro, A., Höök, K. & Benyon, D. 1999. Footprints in the Snow. in A. Munro, K. Höök & D. Benyon (Eds.), *Social Navigation of Information Space*, pp.1-14. Springer Verlag, London, UK

Nokelainen, P., Miettinen, M., Tirri, H. 2002, EDUCO - A Tool for Real Time On-Line Collaboration in Web-Based Learning, in *Proceedings of the ED-MEDIA 2002 Conference*, Denver, USA, pp. 1448-1453

Ronchetti M., 2002, "Why Web pages annotation tools are not killer applications? A new approach to an old problem". *Proceedings of the "IADIS International Conference WWW/Internet 2002"*, Lisboa, Portugal, pp. 735-738

Stauffer, T., 2002, *Blog On: Building Online Communities with Web Logs*, McGraw-Hill Osborne Media

Stevens, R., & Palacio-Cayetano, J. (2003). Design and Performance Frameworks for Constructing Problem-Solving Simulations. *Cell Biology Education*, Vol. 2, pp. 162–179.

Vendlinski, T. and Stevens, R. 2002, Assessing Student Problem-Solving Skills With Complex Computer-Based Task. In *The Journal of Technology, Learning, and Assessment*, Vol.1, No 3, pp. 1-21 Available from http://www.jtla.org.

Yeh, P., Chen, B., Lai, M. and Yuan., S., 1996, Synchronous Navigation Control for Distance Learning on the Web, *Computer Networks and ISDN Systems*, Volume 28, issues 7–11, p. 1207-1218

Zeiliger, R. 1998. Supporting constructive navigation of Web space. In *Workshop on Personalized and Social Navigation in Information Space*, K. Hook, D. Benyon, and A. Munro, editors, 16th-17th March, Stockholm, Sweden.