Foundations of Unconstrained Collaborative Web Browsing with Awareness

Maria Aneiros Vladimir Estivill-Castro
School of Computing and Information Technology
Griffith University
Brisbane 4111, QLD Australia

Abstract

*Research has focused significantly in enabling the Web (WWW) for Computer Supported Cooperative Work (*CSCW*). However, surfing, the most common use in the WEB, remains an individual, rather than group activity. Previous attempts to provide collaborative browsing capability constrain some users to the command of a selected user who controls the browsers of others. We adapt the technology of unconstrained distributed collaborative editors to develop unconstrained collaborative Web browsing. However, the effective collaboration is dependent on the awareness of context and group activity. We develop the history mechanisms for our solution to provide 4 types of awareness commonly discussed in the literature of* CSCW.

1. Introduction

The World Wide Web was labeled as a technology to revolutionize human interaction and communication, very much as when the TV and the telephone were invented. Interestingly enough, the World Wide Web has also been named as a ubiquitous and powerful enabler for Computer Supported Cooperative Work (CSCW). Nevertheless, the most fundamental activity in the Web, surfing (and exploring) its contents remains an individual activity, and not a collaborative, shared and partnered activity. Those systems that offer Web surfing for groups essentially deliver the possibility of a master and a group of followers [28]. The master controls and drives the path of visitation while the others watch. Of course, the control can be passed to another in the team, and the old master becomes an observer. As a family watching TV with one remote control, the holder of the remote control switches channels at will, forcing the others to follow a navigation path. We propose here *Unconstrained* navigation in a Collaborative Surfing Session. For the analogy of a family watching TV, this is equivalent to everybody having a remote control. It may seem impossible to achieve collaboration if all members of a surfing session have equal power and control. The purpose of this paper is to describe our solution to this challenge.

The avenue to provide a solution is the distributed nature of collaborative Web surfing. Distribution provides other resources, since participants are not facing the same monitor (as in the family TV analogy), but spatially distributed (although our solution will work if several or one user are operating one or more browsers on the same monitor). Any participant using its own browser faces the delay of the Internet infrastructure for the WEB. That is, visiting pages at remote sites imposes a delay to retrieve such content (of course, caches, proxies and other alternatives shortcut this, but new site visits have delay). Thus, even in the current model of collaborative surfing (with a master surfer and its followers), the slave followers must (1) wait for signals from the master and then download the page designated by the master, or (2) a central system downloads the page designated by the master and supplies it to all followers. This constrains the surfing by the followers even more. We aim for a system where each participant can visit a new site and expect to obtain the content with equivalent (no more) delay as if operating independently from the group.

The technology we propose for our solution is based on the technology for Unconstrained Distributed Cooperative Editors (UDCE) [24, 25, 29]. Here, several authors are concurrently editing a common document over a distributed computer network in real time. Unconstrained collaboration and good responsiveness demand a replicated architecture. Local edition commands are applied to the local copy immediately, and no participant exercises control or privileges over the others. Thus, Section 2 will show that an Unconstrained Collaborative WEB Surfing Session (UCWSS) reduces to an Unconstrained Distributed Cooperative Editing Session (UDCES). This reduction is to demonstrate the feasibility of the solution and follows many common reductions in computability. For example, multiplication can be performed by repeated addition, or NP-hard problem reductions show that one problem could be solved if another problem's solution was used as a subroutine. Thus, we show that we can achieve UCWSS if we use UDCE.

Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI'03)
0-7695-1932-6/03 $17.00 © 2003 **IEEE**

We do not advocate that a UDCE should be under each collaborative surfing session. If fact, we believe that collaborative surfing demands awareness [18]. In simple terms, for the collaboration to be effective, participants must be aware of the involvement and participation of others. We will detail this more in Section 3. While awareness has been the focus of work on UDCE [3, 7] not so much emphasis is placed on historical awareness.

One of the basic abstract structures constructed during Web surfing is the history of the path of visitation (the stack that allows to back-step). It is fundamental for a sense of status of the session and the surfer's awareness of its current and past visits through the WEB [6, 16]. Thus, we argue that UCWSS must construct a collectively built history of visitation. Moreover, this abstract structure will not only provide awareness to the group, but also the base for other smart computer tools to synthesize useful information for the group (what are common interests could be extracted from such a structure).

In Section 4, we will present the foundations that show that UCWSS with history translates to the problem of Collaborative Unconstrained Distributed Edition of a stack. We will then show that the technology for UDCE can be simplified to obtain feasible and efficient Collaborative Unconstrained Web Surfers.

Section 5 discusses our implementation of these ideas in a simple prototype. Section 6 contrasts our work with previous research. Finally, Section 7 discusses our work with respect to previous work and highlights the avenues open by our results.

2. Unconstrained Collaborative Web Surfing

Unconstrained means concurrent but free access at any time and it has been argued that is fundamental to allow natural information flow among multiple users [25, and references]. Consider the following definition for *Unconstrained* Collaborative Web Surfing Sessions (or UCWSS).

Definition 1 *Two or more users are surfing the Web collaboratively through two or more Web clients (browsers). The surfing is unconstrained, in that each action on a browser operates at least as if this was as a browser operating independently of the others. Independently in the sense that the other browsers do not prevent this browser from obtaining the requested contents as it would do if operating alone. Naturally, if all browsers are accessing the same proxy or are running on the same machine the action of one may inflict performance degradation on the others, but we consider this effect of one browser on the performance of another as part of the de facto effects of the Web technology even for non-collaborative, independent browsers. Thus, this effect will be ignored in what follows.*

Unconstrained collaborative surfing does not prevent any participating browser from obtaining content at any time.

Thus, each participant browser can replace at any time the contents it is displaying to its user in order to fulfill a command by its user (to go back, to go to a specific site, or to follow a link). But with collaborative browsing, another browser may attempt to push into the display some other content. We refer to the other browser as remote. Note that our aim is not to push content, but to preserve the work/surfing of others for further reference. How shall the local browser synchronize the content obtained following local user commands while allowing remote browsers suggestions for content?

Consider now the technology offered by UDCE. This allows a group of distributed users to manipulate independently and in unconstrained fashion a collection of objects in a document. That is, the editor enables each user to issue commands to insert, update and delete objects in the document. Insert adds an object in a designated position, delete removes the object while update modifies properties of a designated object. Unconstrained implies that whenever a user is in a position to meaningfully apply a command to its local copy, the user can do so (is not restricted). For example, a user can update an object still present in its local copy although the object has been deleted by another remote user but network latency has not forwarded the delete operation. The UDCE technology solves three problems: Causality violation, Divergence and Intention Violation.

Perhaps convergence is the most natural to expect. If the editors are working on the same document, after all commands from all users have been collected, the document should reach a state that is consistent and in agreement with all users. In fact, this can be achieved if we place the objects in a document under a central repository or database and we serialize the editing commands from distributed users in the very same fashion as databases ensure consistency of their information. The side effect that we want to avoid is the additional latency that this solution provides because the operation on an object may need to wait for another operation to finish. Unconstrained means that operations take effect in local copies immediately and independently of other editing commands on the same object by remote editors. Convergence under unconstrained editing can not be achieved by central server solutions, since they introduce lag attributable to the central server and remote clients.

Causality respects the logical sequence of operations. For editing, if a user inserts an object into the document and then updates one of it properties, such modifications to the document can not be reflected in a remote site by first updating the property and later inserting the object. In the contents of browsing Web sites, if a user visits site A and then selects a link on A to go to site B, then remote users in the session can not be shown site B before site A.

Intention preservation implies that the intention of a command should survive the effects of commands at remote editors. For example, let the object be the string 'ABCDE' and the local user request INSERT '1' AT POSITION 2 expecting to obtain 'A1BCDE'. But another user's command deletes the first two characters expecting to remove 'A' and 'B' to get 'CDE'. Then, perhaps the result should be '1CDE' to preserve the second editor's wish to remove 'A' and 'B' and the first editor's wish that the inserted character should be before 'C'. Note that applying the local command followed by the remote results in 'BCDE' while applying the remote followed by the local results in 'C1DE'. This may be unsatisfactory to the intentions of both. Unconstrained Collaborative Editing makes no attempt to preserve and manage semantic intentions [25].

We use this illustration of collaborative editing for describing our reduction. In fact, this reduction is, in itself, a specification to what UCWSS are required to support.

The reduction is conceptually simple. Each unconstrained collaborative browser is mounted on a collaborative unconstrained editor. For simplicity, we say that the browser is required to display different content when its user has requested to go back or to follow a link (by using a URL from history, favorite bookmarks, following a link in the current display or typing a target URL). The request to display different content is translated in the browser into the following sequence of operations.

1. Retrieve the desired content as a standard browser.
2. Issue the operation 'Select All' to the local UDCE.
3. Issue the operation 'Delete' to the local UDCE.
4. Issue the operation 'Insert' to the local UDCE and provide the contents just retrieved.

UDCE handles Convergence, Causality and Intention preservation. With this reduction we observe the following. Clearly, the availability of UDCE makes available Unconstrained Collaborative Web Surfing Sessions. As a specification, an UCWSS is an environment that offers at least the functionality obtained by this conceptual reduction. It provides unconstrained surfing with Convergence, Causality and Intention Preservation. We are not suggesting here that actual implementation of UCWSS should be constructed in this way. We are just demonstrating the feasibility of their existence.

3. Awareness

Awareness of one's participation is central to collaboration with others. Naturally, awareness of the presence of others facilitates cooperation since one can assist others or request assistance from others. Awareness is also central to guarantee the context and the environment of collaboration with respect of individual interests. In particular, privacy may be severely eroded if one's surfing session is observed by others without consent. Thus, awareness is necessary for effective Collaborative Web surfing. Note that awareness in the context of CSCW has been defined as "an understanding of the activities of others, which provides a context for your own activity" [7]. Many tools have been proposed to facilitate the coordination among people [18, and references] and to help communication and collaboration. In particular, the trade-off between the requirements of individuals (emphasis on control) and the requirements of groups (emphasis on awareness) has been explored in the context of workspace navigation [12]. This *group awareness* [18] is catered for in our UCWSS by an application that will inform users when their session is part of a collaborative session, allowing them to disconnect or join at will. Even within a collaborative surfing session, users may chose to veto their visitation to a page from view by the rest of the group.

Notably, information about what has happened before and who else is around is needed in shared workspaces. These issues are fundamental aspects of what has been called *workspace awareness* [18]. We suggest that the history of the navigation path in UCWSS provides what has happened before. We will also suggest how this structure provides the awareness about others.

The third type of awareness is *contextual awareness* which relates to the application domain, rather than the users [18]. Here, we want to identify what content is interesting for the community. For our UCWSS, we propose to synthesize the history of navigation paths into editorial pages and into summaries of collective interests.

Finally, we deal with the fourth type of awareness, *peripheral awareness* [18] by presenting our collective history as an optional display in the GUI that constitutes a browsing session and placing it away from the display of Web pages (like a tool-bar).

4. The Group Unified History

We have seen is Section 2 that it is feasible to have UCWSS. However, browsers with only this capability would provide no awareness and thus fail short of being useful tools for CSCW. We emphasize here that in order to satisfy the four aspects of awareness, the browsers in a UCWSS should at a minimum compose a Group Unified History (GUH) [1]. This is analogous to current browsers history mechanisms that allow users to return to previous pages under the model of a stack (last in - first out storage structure). Empirical studies carried out in 1997 found that 58% of navigation is backtracking [27]; later, in 2001 it was found to be much higher, 81% [5]. We suggest that the GUH

is built automatically while registered users are collectively browsing the WWW and it should reflect the recency of visitation. It can be thought of as an editable document, with limited functionality, where new sites are added at the end of the document.

An advanced sense of awareness is provided when this list is analyzed and useful contents made public. This public GUHs could be compared to WWW sites that show entire collections of bookmarks for specific topics. GUHs could become valuable Knowledge Management assets within an organization, i.e. a Knowledge map that could be useful, for example, for training new employees.

We now show that such GUH can also be realized with the technology of UDCE. The GUH would be modeled as a shared document that can be edited in an unconstrained cooperative distributed fashion. But we will take advantage of the fact that the applicable operations are restricted to a stack. In particular, when a user in a UCWSS moves to a new site, this corresponds to inserting at the end of the history. The item inserted is an identifier of the page visited, like an URL and is considered atomic.

Thus, we review the consistency model of UDCE applied to GUH. The model has three properties: convergence, causality preservation, and intention preservation. Convergence requires that all copies of the GUH be identical after the same collection of operations. Thus, we require that eventually all local copies of GUH converge to the group's GUH, even if different users have locally followed different paths. Essentially, the collective visitation path should became the same to all involved.

We propose here that to maintain a true unconstrained character of collaborative surfing, the divergence of paths among synchronous users does not impose in any of them the refreshing of their browser's contents. The system will peripherally update the local GUH.

4.1 The Consistency Model

Consider collective unconstrained editing of the GUH by distributed browsers. First, we define the operations. When a user manipulates a browser to display content by following a URL(using URL from favorite bookmarks, following a link in the current display, typing a target URL or using a URL from history), we consider the browser is issuing a PUSH(URL) into the collective GUH. When the user presses the BACK button, this issues also a PUSH(URL). Only when a designated UNDO button is used, the user instructs a POP(URL) to the GUH and the resulting content on display is as with the BACK button (the previously visited page by the user). We also allow for a PREVIOUS button. This does not affect the GUH but refreshes the local display with the previous WEB-page to the current top of the GUH(note that the previous page to the top may be in the GUH because of a remote visit by another user and the local browser may have not yet displayed this content). The NEXT button refreshes the display with the next WEB-page indicated by the GUH unless already at the top. NEXT is the reverse of PREVIOUS and neither of them change (edit) the GUH.

A PUSH(URL) can be applied by the user at any time. A POP(URL) can only be applied if the current URL on display is exactly the same as the last on the local GUH. A situation where UNDO is not applicable is if a user Alice visits URL A, then A is reflected on Bob's browser who uses a link in A to retrieve URL B. Then, B is reflected (pushed) into Alice's browser. Alice will be able to UNDO A only if Bob undoes B first.

In contrast to the nomenclature for UDCE, we distinguish between a remote operation reflected on a local browser, and a remote operation executed (happening) at the local site. By *reflected* (pushed) we mean the content visited by the remote user is displayed in the local browser after the OK of the local user. By executed (happening) we mean tat the local copy of the GUH is affected (i.e. updated) by the remote operation. Happening is managed without user interaction because it does not affect the user's main view, but only the peripheral (and optionally displayed) GUH. Note that a local operation always happens and is reflected locally (thus, it can never be the case that it happens but is not reflected). This is because local operations happen and are reflected by immediate local execution.

We are now in a position to formulate a consistency model for the GUH similar to UDCE [25]. We now define a casual (partial) ordering relation on operations in terms of their generation and execution sequences.

Definition 2 (*Causal ordering relation "$\rightarrow$"*) *Given two operations O_a and O_b generated at sites i and j respectively, then $O_a \rightarrow O_b$, if and only if*

LOCAL CASE. *$i = j$ and the generation of O_a happened before the generation of O_b, or*

REMOTE CASE. *$i \neq j$ an the execution (happening) of O_a at site j happened before the generation of O_b, or*

TRANSITIVE CLOSURE. *There exists an operation O_x such that $O_a \rightarrow O_x$ and $O_x \rightarrow O_b$.*

Note the following observation.

Proposition 1 *If O_a, O_b and O_c are all local, O_a happened before O_b and O_b happened before O_c, then O_a happened before O_c.*

So, locally, the relation HAPPENS_BEFORE is transitive. A distributed system that maintains the causal ordering relation "$\rightarrow$" is the extension of the natural local expectation to unconstrained collaborative surfing. That is, the causal

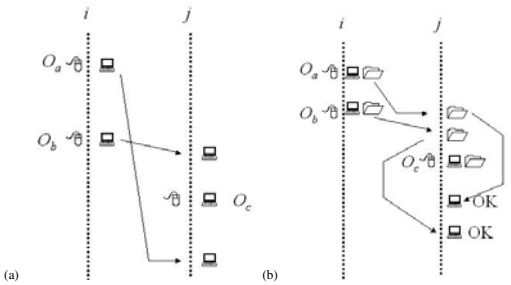

Figure 1. (a) A sequence diagram of a non acceptable situation. The mice icons mean the generation of an operation. The display icons mean their reflection (and thus, their execution) at a site. We have $O_b \rightarrow O_c$, because O_c happens after O_b is site j. Note that the browser at j not only has executed (updated its GUH with O_b) but also reflected (updated its display). We have $O_a \rightarrow O_b$ because O_a happens before O_b in site i. Thus, O_a should not happen after O_c is site j. (b) A sequence diagram of an independent operation. The file icons mean the happening (updating) of a GUH. The operation O_c is a direct typing of an URL without reference to an updated GUH.

ordering relation defines formally one aspect of what is required. It is required that the local transitivity extends remotely. A situation like Figure 1 (a) must never occur; that is, we have $O_a \rightarrow O_b$ and $O_b \rightarrow O_c$ but $O_a \rightarrow O_c$ is false. Because computer networks introduce random delays, the relay of O_a from site i to site j may get delayed so much that O_b arrives first at j. In our implementation, operation O_b will not happen at j because O_b has an operation id (through a State Vector) [26] indicating dependency on a previous operation O_a.

Now we can introduce the notion of independence.

Definition 3 (*Dependent and independent operations)*

Dependent operations. *Given two operations O_a and O_b, if $O_a \rightarrow O_b$, then O_b is dependent on O_a.*

Independent operations. *Given two operations O_a and O_b, they are independent if neither $O_a \rightarrow O_b$ nor $O_b \rightarrow O_a$.*

Figure 1 (b) illustrates an independent operation O_c. At site j we assume that O_c is a fetch of a WEB page by typing in the URL. Although the GUH at site j has been updated by the arrival of O_a and O_b, these have not been reflected in the display. O_c is independent on both O_a and O_b. Any order of display for O_c would maintain causal ordering. In fact, site i may reflect O_c after O_b, while site j reflects O_a by the OK. Note that, in the example of Figure 1 (b) site i may end up with a list of visited pages that look like O_a, O_b, O_c while site j may end up with a list that looks like O_c, O_a, O_b. Eventually, this must be a commonly edited object. That is, this must be the unconstrained editable global GUH. Thus, the following definition.

Definition 4 (Consistency model) *A collective surfing system that collectively and unrestrictedly edits a* GUH *is* consistent *if it always maintains the following three properties:*

Convergence. *Whenever the same set of operations have happened at a set of sites, all copies of the* GUHs *at those sites are identical.*

Causality preservation. *For any pair of operations O_a and O_b, if $O_a \rightarrow O_b$, then O_a happens before O_b.*

Intention preservation. *For any operation O, the effects of executing O at all sites is the same as the intention of O at its generation, and the effects of executing O does not change the effects of independent operations to O.*

We do not demand that $O_a \rightarrow O_b$ implies O_a reflected before O_b. This gives the freedom to users for which both O_a and O_b were remotely generated to skip reflecting O_a if they are more interested in O_b. But, if they just click OK repeatedly, those pages in the GUH that have not been reflected at their site will be reflected in the order of the GUH. For instance, in Figure 1 (b), successive OK clicks reflect O_a and O_b at site j (both have already happened). But the user at j could have used the GUH to visit O_b directly.

Our definition of consistent model above mirrors the definition for collaborative unconstrained editing systems. It specifies what UCWSS are. All terms, like causal preservation, happens before, $\rightarrow$ and so on appear in the definitions given before, except for effect and intention. Our claim is that, in the case of GUHs, all possible definitions one may wish to make are such that they are covered by the first two properties. That is, we only need algorithms for convergence and causality preservation and we will obtain algorithms for intention preservation. UCWSScan be implemented with less machinery than UDCE.

Theorem 1 *If a system for* GUHs *is convergent and preserves "$\rightarrow$", then the system is consistent.*

Proof: Implementation of the operations on a GUH consists of implementation of PUSH and POP operations on a stack. For this, we present an editing system that works on strings of symbols from an alphabet Σ. This system has two operations.

1. INSERT[S,p](C) means insert string $S \in \Sigma^*$ at position $p > 0$ into C.
2. DELETE[n,p](C) means delete from C as many as $n > 0$ symbols starting from position $p > 0$.

Figure 2. GUHs after a few operations by three partners and some network delay.

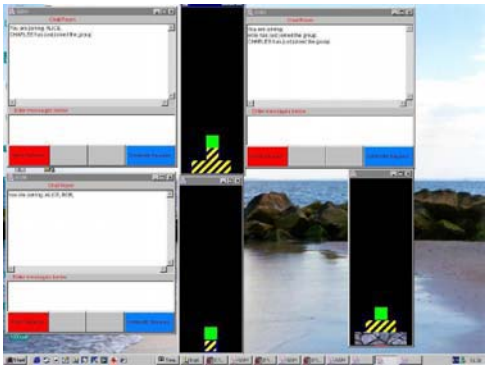

Figure 3. After Charles' operation arrives everywhere.

Here Σ^* means the language of all strings of 0 or more symbols in Σ (as commonly used for regular expressions) and a position in a string $C = c_1, \ldots, c_t$ means the gap between c_{i-1} and c_i (with the understanding that, for $p = 1$, this is the position before all symbols in C and for $p = t + 1$ is the position after all symbols in C). Insertion into an empty string C gives the insertion string S as a result.

A system based on these two simple operations under a text data model is powerful enough to emulate real editing systems as `vi` and `emacs` [19, 20] and has been used in the literature of UDCE before.

Now we show that we can implement a GUH on such system. Simply, make Σ the set of URLs. Then

1. PUSH(URL) of GUH=$[URL_1, \ldots, URL_t]$ is conceptually implemented by INSERT$[URL, t + 1]([URL_1, \ldots, URL_t])$.
2. POP(URL) on GUH=$[URL_1, \ldots, URL_t]$ is conceptually implemented by DELETE[1,t]$([URL_1, \ldots, URL_t])$.

Now, the effect of an operation depends only and can be correctly interpreted on its own context. But then, the execution effects of independent INSERT/DELETE operations in a text document model will not interfere with each other because all operations in this conceptual implementation of GUHs are strings of size 1. The conditions for an INSERT operation not to interfere are fulfilled because we only insert a string of size one. DELETE/UNDO is only allowed at the site of generation, only removing one item and in a context that has nothing after it. $Q.E.D$

The proof emphasizes that intention can only be expressed in terms of a context of where the operation is to occur. For example, delete the string of symbols '12' after the second 'A' in 'BAC1123CA112'. However, in a GUH with only PUSH and POP there is no facility to express context. The only reference to position is implicit, the top of the stack, but the operation can not be qualified by what is already in the stack. Thus, the position of the operation cannot be qualified by what is already in the stack. In fact, absolute positions are not reflected in a stack and relative positions, like 'keep URL on top of the stack' would constrain all other browsers to slaves.

5. Prototype

We have implemented the algorithms in a prototype using JAVA applets. The techniques are essentially the State Vectors, timestamps and the algorithm known as the "undo/do/redo" scheme [8, 26]. Our code has been instrumented to verify scenarios with network delays. For example, when a browser fetches a page, we can hold the message out of this participant to the other participants, as if the network was delaying it. Simulations of reversal of order of messages on the network are to be instrumented. For illustration here we consider an session with 3 participants. We call them Alice, Bob and Charles and they join the UCWSS in that order (so on concurrent independent operations, Alice's take preference over Bob's and these over Charles'). Alice visits page A but network delay prevents this happening (and thus reflected) at Bob's and at Charles respective browsers. Then, Bob visits page B_1 independently and we assume that the network relays immediately this to remote participants, and thus happens in the others GUH. Then, Charles visits page C after B_1 but this does not happen in Alice's nor in Bob's browser because of network delay. Concurrently and independently to Charles visit to C, Bob visits page B_2. Figure 2 is a screen shot of the 3 applets for the respective GUHs of the participants and applets of respective chat boxes. The right most chat-box and GUH is Alice's while the top left is Bob's and the bottom left is Charles'. Items in the GUHs are identified by a color per

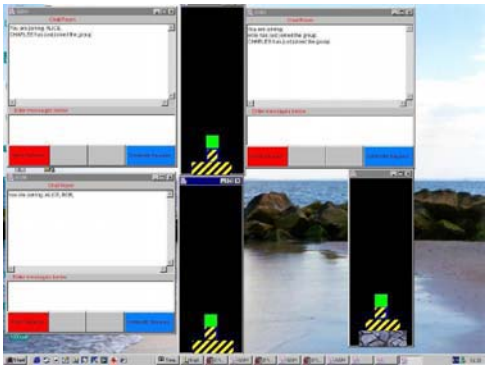

Figure 4. After Bob's B_2 happens at remote GUHs.

Figure 5. GUHs are all the same after all operations are received by all sites.

participant (as is common in CSCW [3, 8]). Alice's URLs are red, while Bob's are blue and Charles' are green. We see that Alice's does not have a green URL while Bob has only blues (his local operations) and Charles has a blue (B_1) and his local green.

Now, Figure 3 shows the GUHs after packets communicating Charles' operation arrive at remote sites forcing the update in GUHs. The green URL is at the top on all GUHs.

When Bob's second operation visiting B_2 happens at remote sites (Figure 4) we see that Bob's URL for B_2 modifies Alice's and Charles GUHs by placing itself under Charles operation.

Finally, we let the first operation overall, Alice's visit to A to be communicated to remote sites. The red URL now updates the GUHs and goes at the bottom of everybody's stack (refer to Figure 5). We see that all GUHs converge. Note that the chat boxes and the GUHs display as a stack can be placed on the background or on a peripheral region of the screen and are updated automatically.

6. Previous and Future Work

Several examples of collaborative Web browsers have been described in the literature. GroupWeb [21] and GroupScale [10] focus on one user controlling the browser of others, or a user tracing what others are doing [11]. Because of their architecture and specialized browsing requirement standard browsers can not joining the sessions. So others have focused on other WEB technologies [2], like replacing HTML elements with Java applets that establish a connection to a central server [13], and using JAVA and proxies [4]. We emphasize on the unconstrained nature of our approach and its applicability to all client/server WEB technologies. Thus, our contribution is more fundamental, beyond the debate on the particular WEB technologies to build prototypes. More recent work has remained synchronous, constrained and master/slave based [9, 15].

Other work has focused on the applications, like customer support in banking [14] or distance learning [22]. Previous work has stressed that collaboration within the same display is also important [17]. Our presentation here applies to those applications and to an unrestricted number of instances of a browser operating on the same or remote displays in unrestricted networks (not restricted to LANs, for example, as is the case of [9]).

7. Final Remarks

Since the first papers [21] on group surfing were published, more relaxed models than master-slave and "what-you-see-is-what-I-see" have been sought. Once the fundamentals of UCWSS are in place, we can see that our proposal provides even more flexibility. In particular, the items on the GUH can be treated as objects with properties. One of these properties could be its list of owners. The first time the URL is used, the user at that browser becomes the only owner, further reflections of this page in browsers of other users add those to the list of owners. This mechanism allows flexibility for the deletion of items form the GUHs. For instance, previously we indicated that if Alice visited A, and A's reflection on Bob's browser is used to visit B, then Alice would have to wait for Bob to remove B so Alice could remove A. With lists of owners, Alice would simply remove herself from the list of owners of A. Then, if Bob also removes himself as an owner of A, the URL A is deleted. The algorithms for this are not much more complex, and could simply be algorithms for collaborative editing of objects with properties [23]. Thus, we see that we have provided a very powerful mechanism to achieve collaborative unconstrained surfing sessions.

References

[1] M. Aneiros, V. Estivill-Castro, and C. Sun. Group Unified Histories — an Instrument for Productive Unconstrained CoBrowsing Submitted manuscript, 2003.

[2] R. Bentley, T. Horstmann, and J. Trevor. The world wide web as enabling technology for CSCW: The case of BSCW. *Computer Supported Cooperative Work: The J. CSCW*, 6(2-3):111–134, 1997.

[3] E. Bier and S. Freeman. MMM: a user interface architecture for shared editors on a single screen. *UIST Fourth Annual Symposium on User Interface Software and Technology. ACM Symposium on User Interface Software and Technology*, pp. 78–86, MY, 1991. ACM, ACM.

[4] G. Cabri, L. Leonardi, and F. Zamborelli. Supporting cooperative WWW browsing: a proxy-based approach. IEEE *7th Euromicro Workshop on Parallel and Distributed Processing. PDP'99*, pp. 138–145, Funchal, 1999.

[5] A. Cockburn and B. McKenzie. What do web users do? an empirical analysis of web use. *Int. J. of Human-Computer Studies*, 54(6):903–922, 2001.

[6] A. Dix and R. Mancini. Specifying history and backtracking mechanisms. *Formal Methods in Human-Computer Interaction, Formal approaches to computing and information technology, Chapter 1*, pp. 1–23. Springer Verlag, 1998.

[7] P. Dourish and V. Bellotti. Awareness and coordination in shared workspaces. J. Turner and R. Kraut, eds., *ACM Conf. Computer Supported Cooperative Work (CSCW'92)*, pp. 107–114, Toronto, Ontario, 1992.

[8] C. Ellis and S. Gibbs. Concurrency ontrol in groupware systems. *ACM SIGMOD Int. Conf. Management of Data*, pp. 339–407, 1989.

[9] A. Esenther. Instant co-browsing: Lightweight real-time collaborative web browsing. *11th Int. World Wide Web Conf.*, Honolulu, Hawaii, USA, 2002. CD-ROM Posters ISBN 1-880672-20-0.

[10] T. Graham. Groupscape: Integrating synchronous groupware and the world wide web. *INTERACT'97*, pp. 547–554, Sydney, Australia, 1997. Chapman and Hall.

[11] T. Gross. Supporting collaboration and cooperation in digital information environments. *Workshop on Collaboration and Cooperative Information Seeking in Digital Environmetns*, Seatle, WA, 1998. CSCW-98.

[12] C. Gutwin and S. Greenberg. Design for individuals, design for groups: tradeoffs between power and workspace awareness. *ACM Conf. Computer Supported Cooperative Work (CSCW'98)*, pp. 2017–216, Seattle, WA, 1998.

[13] S. Jacobs, M. Gebhardt, S. Kethers, and W. Rzasa. Filling HTML forms simultaneously: Coweb-architecture and functionality. *5th Int. World Wide Web Conf.*, vol. 28, pp. 1385–1395, Paris, 1996. Computer Networks and ISDN Systems no.7-11, Elsevier, Netherlands.

[14] M. Kobayashi, M. Shinozaki, T. Sakairi, M. Touma, S. Daijavid, and C. Wolf. Collaborative customer services using synchronous web browser sharing. *ACM 1998 Conf. Computer Supported Cooperative Work*, pp. 99–108, Seattle, WA, 1998. ACM.

[15] Y. Laurillau. Synchronous collaborative navigation on the WWW. *CHI-99 extended abstracts on Human factors in Computer Systems*, pp. 308–309, Pittsburgh, USA, 1999. ACM. Student Posters.

[16] A. Lee. *Investigations into History Tools for User Support*. PhD thesis, Department of Computer Science, University of Toronto, Ontario, Canada, 1992.

[17] H. Lieberman, N. Van Dyke, and A. Vivacqua. Let's browse: a collaborative web browsing agent. M. Maybury, ed., *IUI 99. 1999 Int. Conf. Intelligent User Interfaces*, pp. 65–58, Redondo Beach, LA, 1999. ACM.

[18] O. Liechti and Y. Sumi. Awareness and the www. *Int. J. Human-Computer Studies*, 56(1):1–5, 2002. Editorial.

[19] A. Prakash and M. Knister. A framework for undoing actions in collaborative systems. *ACM T. Computer-Human Interaction*, 4(1):295–330, 1994.

[20] M. Ressel, D. Nitsche-Ruhland, and R. Gunzenhäuser. An integrating, transformation-oriented approach to concurrency control and undo in group editors. M. Ackerman, ed., *1996 ACM Conf. on Computer supported cooperative work*, pp. 288–297, NY, 1996. ACM, ACM Press.

[21] G. S. and R. M. GroupWeb: A WWW browser as real time groupware. *ACM Human Factors in Computing Systems, CHI Companion*, pp. 271–272, Vancouver, Canada, 1996.

[22] T. Souya, M. Kobayashi, S. Kawase, and K. Ohshima. Joint class experiments based on realtime web-browser synchronization. *3rd Asia Pacific Computer Human Interaction (Cat. No.98EX110)*, pp. 367–372, Shonan Village, 1998. IPSJ, IEEE Press.

[23] C. Sun and D. Chen. Consistency maintenance in real-time collaborative graphics editing systems. *ACM T. Computer-Human Interaction*, 9(1):1–41, 2002.

[24] C. Sun and C. Ellis. Operational transformation in real-time group editors: Issues, algorithms and achievements. *1998 ACM Conf. Computer-Supported Cooperative Work*, pp. 59–68, Seattle, WA, USA, 1998. ACM, New York, NY, USA.

[25] C. Sun, X. Jia, Y. Zhang, Y. Ynag, and D. Chen. Achieving convergence, casuality-preservation, and intention-preservation in real-time cooperative editing systems. *ACM T. Computer-Human Interaction*, 5(1):63–108, 1998.

[26] C. Sun, Y. Yang, Y. Zhang, and D. Chen. Distributed concurrency control in real-time cooperative editing sysems. J. Jaffar and R. Yap, eds., *Asian Computing Scince Conf.*, pp. 84–95, Singapore, 1996. Springer Verlag LNCS 1179.

[27] L. Tauscher and S. Greenberg. How people revisit web pages: Empirical findings and implications for the design of history system. *Int. J. Human Computer Studies, Special issue on World Wide Web Usability*, 47(1):97–138, 1997.

[28] H. Wagner. Tracking the navigation behavior of web communities. Master's thesis, Department of Computer Science, U. of Munich, Germany, 2002.

[29] Y. Yang, C. Sun, Y. Zhang, and X. Jia. Real-time cooperative editing on the internet. *IEEE Internet Computing*, 4(3):18–25, 2000.