A Flexible Architecture for Collaborative Browsing

Guillermo de Jesús Hoyos-Rivera\textsuperscript{\textcopyright}  
LAAS – CNRS\textsuperscript{\textregistered}  
ghoyos@laas.fr

Roberta Lima-Gomes\textsuperscript{\textdagger}  
LAAS – CNRS\textsuperscript{\textregistered}  
rgomes@laas.fr

Jean-Pierre Courtiat  
LAAS – CNRS\textsuperscript{\textregistered}  
courtiat@laas.fr

Abstract

Collaborative Browsing is a new and promising research area whose purpose is to provide new collaboration schemes among users browsing the Web.

To become an efficient collaboration tool, Collaborative Browsing should go far beyond a simple mechanism for coordinating the browsing of Web pages (like Web tours) and should include, in a integrated framework, awareness and communication facilities.

A general functional architecture of a Collaborative Browsing system is presented in the paper. It relies on a rule-based system that permits to precisely express the browsing and communication strategies among users of a collaborative session. The proposed architecture is therefore general-purpose. By defining the policy rules specifying the behavior of a collaborative session, it may be easily adapted to specific applications in the e-Learning or e-Business domains.

The general architecture of CoLab, a Java-based software environment implementing the proposed Collaborative Browsing system is introduced, and the features of its first operational version are presented in some details (CoLab 1.0).

Keywords: Collaborative, Browsing, Web, Policy rules, Proxy, Java.

1. Introduction

Collaborative Browsing attempts to allow several Web users to coordinate their actions, through the synchronized visualization of Web pages, and the use of awareness and communications facilities. In this way it extends the individual capabilities of each of the users involved in the process.

In this sense Collaborative Browsing can be seen as an extension of traditional Web browsing, firstly dedicated to isolated distributed users with low interactions among them, to groups of users with a mutual and more important consciousness of the group presence and interaction. This concept aims therefore to extend the multimedia document interactive access and visualization to group of users where a subset can communicate through synchronous/asynchronous communication tools, in a distributed co-presence.

The paper is organized in six main sections. In section 2 we present the motivations for Collaborative Browsing. In section 3 we present the functional architecture. In section 4 we present the main technical details of the CoLab environment, as well as its current state of development. In section 5 we present a comparison with related works. Finally in section 6 we draw some conclusions and discuss future work.

2. Motivations for the Collaborative Browsing

Today' \textit{World Wide Web} is fundamentally a publishing medium—a place to store and share documents including different types of media: text, image, audio, video, etc, allowing users to browse documents by clicking on hyperlinks. Usually users browse lonely, with no other way to share their knowledge or opinions about the visited pages to other users, than to communicate through the use of external communication tools, like e-mail or chat.

A possible evolution of the Web paradigm is precisely to define new browsing capabilities and to integrate them with communication tools in order to allow a group of users i) to browse documents in a controlled manner depending on policy rules stated at the group level, and ii) to share information by using communication tools which may be activated/deactivated according also to policy rules. Let us call Collaborative Browsing [1] this way to work.

As illustrated in Figure 1, in traditional browsing, Web users browse documents in an isolated fashion; access control to a specific Web page is usually accomplished by an elementary authentication mechanism (username-
password, IP address, etc.), and the only way to communicate is to use external facilities.

![Diagram](Image)

**Figure 1: From Traditional to Collaborative Browsing**

In **Collaborative Browsing**, a user registered within a group browses documents in a controlled manner. His browsing capability does not only depend on his previous browsing action, but possibly also on the browsing actions of other users within the group, as well as on other factors including for example timing and location. The capability of an user to visit a specific page depends therefore on the right he gets from the access control system which may implement elaborated access control rules defined at the group level. The same way to operate is also applied to control the access to communication tools.

From our point of view, a **Collaborative Browsing** system provides two main functionalities: i) a browsing facility in charge of controlling the access to documents and media, and ii) a communication facility in charge of controlling the access to general-purpose communication tools, like whiteboard, audio/videoconferencing, application sharing, etc. The way interactions (either formal interactions implemented by the browsing facility or informal interactions implemented by the communication facility) may occur within a group depends on policy rules characterizing the behavior of the group. These rules, introduced in Section 3, will be used by the access control system in charge of implementing the desired collaboration policy within the group.

Let us consider more in detail document browsing within the proposed **Collaborative Browsing** framework. Three document browsing strategies are introduced in order to better illustrate the intuition behind our proposal. These strategies are called: **Coordinated Browsing**, **Coordinated Adaptive Browsing** and **Location-based Browsing**.

**Coordinated Browsing** (sometimes called Web tours) is applicable to e-Learning. We apply here the metaphor of a **Master** (the teacher) and of a group of **Slaves** (the students). The **Master** has an active behavior as he browses several Web pages (or other types of documents and media). On the other hand, each **Slave** has a passive role consisting in just following the **Master’s** browsing activity. However, at any time, the **Master** may decide to provide some “autonomy” to some **Slaves** for a defined period. These **Slaves** can then browse in an asynchronous fashion and besides, at the same time, the **Master** could follow their browsing activity.

**Coordinated Adaptive Browsing** is applicable to **Collaborative Engineering**. We consider a collaborative work within a firm where employees (the CEO, some managers, designers and engineers) have to share information that is relevant to their function inside the firm. During a presentation of the current status of a project, it is desirable, for example, that all technical data be presented only to the designers and engineers. On the other hand, information related to the overall costs of the project should concern only the CEO and the managers. Some other information might be available to all the users. Once more, we can apply the **Master/Slave** metaphor but in a dynamic way, so as to allow different users to coordinate the session. However, once a **Master** accesses a document (URL), not necessarily the same document is presented to the **Slaves**. In fact, data sent to a **Slave** (URL) will depend on his profile.

**Location-based Browsing** is applicable to **Guided visits**. In this case, contrary to the previous examples, there is no particular user in charge of coordinating the session. Actually, user movement triggers all the browsing activity. Accordingly, users belonging to the same session will access different resources at the same time. Once registered in a session, an user starts receiving documents and media according to his current physical location. In a similar way, the access to communication channels may also be conditioned by the user location. Additional time constraints might also be added to control the browsing activity.

The previous browsing strategies correspond to three particular examples, which may be implemented by the proposed rule-based browsing system. Many other strategies may be elaborated in different application areas.

### 3. The functional architecture

The functional architecture of the **Collaborative Browsing** system is illustrated in Figure 2. A **Collaborative Browsing** session is characterized by a group of users, who may dynamically log-in/out the session, and a set of available resources (documents and communication tools). Documents are essentially hypermedia documents (like HTML, XML, possibly SMIL), classical documents (like PDF, PS, DOC, etc.) and other media (like video/audio streams). Document browsing within a group may be more or less coordinated depending on the browsing strategy defined for the session. Communication tools can be asynchronous (like e-mail, discussion board, ...) and synchronous (chat, audio/videoconferencing, whiteboard, application sharing,
The global access control policy of a session is defined by a set of static policy rules, which we assume completely specified before starting the session.

![Figure 2: The Collaborative Browsing functional architecture](image)

### 3.1. Main elements of the architecture

Three main elements may be identified in the proposed architecture: the **Collaboration Engine**, the **Browsing Manager** and the **Communication Manager**.

By making reference to the general policy framework defined by the IETF [12], the **Collaboration Engine** plays the role of a PDP (Policy Decision Point). It interprets the session policy rules for the current state, and decides which resource may be accessed by which user (i.e. which Web page may be visualized by which user, which communication channel may be used by which user).

The **Browsing Manager** and the **Communication Manager** play the role of a PEP (Policy Enforcement Point), the first one for implementing the session browsing strategy, and the second one for controlling the access to the different communication tools available for the session.

As introduced later, the **Browsing Manager** acts essentially as a centralized proxy between the session and the Web servers hosting the session documents. The **Communication Manager** is distributed among the different communication servers implementing the communication tools and is responsible for the activation/deactivation of the communication channels among users.

### 3.2. The Collaboration Engine

The **Collaboration Engine** is the core element of the proposed architecture. Based on decisions made by the interpretation of the rules in some state, it defines all the resource access rights for the session users.

![Figure 3: The Collaboration Engine behavior](image)

**Figure 3: The Collaboration Engine behavior**

As illustrated in Figure 3, the **Collaboration Engine** acts essentially as a state machine. For some input event (a browsing event – for instance, the click of some user on an hyperlink – a communication event – for instance, the activation/deactivation of some communication channel – a temporal event – for instance the timeout of some time constraint – a location notification event – for instance the notification that some user has reached some specific location – etc.) in some state, the **Collaboration Engine** will produce some action (browsing, communication or other kind of action, ...) and enter the next state.

The state of the whole system is composed of three elementary states, as described informally next.

**Session state**: it represents the information related to the browsing session, as for example the list of currently registered user id’s, or the Web page(s) that have been visited in the session. The latter plays an important role since it indicates at which point a user must start when entering into an already active session.

**User state**: it represents the information related to each user registered in the session. It must include the user’s id information (name, login, etc.). It can also make reference to particular user profiles. Other information may appear at this level, like the used device type (in order to be able to apply media adaptation techniques), the IP number and the current location of the user.

**Resource state**: it represents the information related to each available resource. Resources can be documents, media or communication channels. A resource can be accessible or not at a given moment, depending on certain conditions. Also, a certain resource can be never available for a certain type of device or user. **Meta-data** can also be associated with documents. They will be useful for providing information about a document, which may be later interpreted and processed by the **Collaboration Engine**.

Current work deals with the definition of the formalism to be used for expressing the policy rules. Either, policy-oriented specification languages or general-purpose formalisms, may be used within this framework, and we
are currently assessing some options. Three requirements are important: i) the language or model should be enough expressive and in particular it should be able to express time constraints, ii) the language or model should have a precise semantics, iii) a support tool should be available for the language or model in order, for example, to check the consistency of the rules.

Concerning this subject, we consider intuitively that a model like that of Ponder [13] can be used. Ponder is a specification language that supports authorization and obligation policies specifications. Authorization policies can be positives (allow) or negatives (deny). Obligations policies are event-triggered condition-action rules. These events can be both, actions performed by the users, or the passing of time. Key concepts of the language include roles to group policies relating to a position in an organization, relationships to define interactions among roles, and management structures to define a configuration of roles and relationships pertaining to an organizational unit, such as a department.

Other options may be considered, such as Graph Grammars [11], and other general-purpose formal description techniques.

3.3. Some applicative examples

In order to illustrate some of the concepts introduced in paragraph 3.2 lets us see how they may be applied to some browsing strategies defined in Section 2.

Configuring the Collaboration Engine for a Coordinate Browsing strategy leads to the definition of: i) two profiles (Master and Slave), ii) the group of users, iii) the set of available resources, iv) the default starting point of the session, and v) the policy rules that authorize the Master to browse, unless it has explicitly authorized a Slave under certain conditions.

Configuring the Coordinated Adaptive Browsing strategy leads to the definition of: i) different profiles according to the employee functions in the firm, ii) the group of users, iii) the set of available resources associated with the meta-data defining their semantics, and iv) the policy rules that define the association between each URL accessed by the Master and the URLs that must actually be presented to each Slave as a result of the Master's browsing action.

4. The CoLab environment

This section presents CoLab, the software environment, currently under development, which implements the general Collaborative Browsing architecture introduced in Section 3. We first define the CoLab initial requirements, and then present its general architecture identifying the main technologies used before describing the status of the current operational implementation (CoLab 1.0).

4.1. CoLab implementation requirements

CoLab platform is oriented to non-expert users. As a consequence, it should be simple and easy-to-use, minimizing the user need to configure the system. Required configuration information appears at two levels: i) at the level of the manager of a Collaborative Browsing session who is responsible to define the policy rules of the session, as well as the profiles of its users, and ii) at the level of each user of a Collaborative Browsing session, who has initially to log-in the session Access Server (details in paragraph 4.2). No other configuration is required from the user who may use any browser supporting Java Applets.

4.2. CoLab general architecture

Figure 4 illustrates the CoLab general architecture, introducing its main components: the Access Server, the Communication Servers, the Configuration and Directory Server, the Clients, and the underlying Communication Network.
the original URL as a parameter, besides with other control parameters.

Communication Servers are the components in charge of the management of the communication tools among the users of a session. Using the cooperative platform developed at LAAS/CNRS, the following tools are available and are being integrated within CoLab: chat, whiteboard, application sharing and audio/video-conferencing. Communication Servers should consult the Access Server to determine whether some user is allowed to use some communication channel.

Finally, the Configuration and Directory Server hosts all the configuration information of a session (policy rules, ...) and of its users (user profiles, ...).

There is a-priori no assumption about the underlying communication network, with the exception that it should offer enough bandwidth to ensure an adequate level of QoS. End to end communication is based on TCP/IP (HTTP/TCP/IP for the access to the Web pages, RTP/UDP/IP for the continuous streams). Current experimentations are carried out on a 100 Mbps switched Ethernet local area network, however future developments will address specifically the QoS issue within the context of WAN/LAN communication networks.

One potential problem of the CoLab architecture, as presented in this section, is related to its scalability, in particular at the level of the Access Server, which plays a fundamental role. In the current implementation, we consider a centralized architecture for both access control and Web pages caching. Further study dealing with a distribution of the Access Server to balance the workload among distributed servers will be initiated after a careful analysis of the performance and scalability issues of the current architecture.

4.3. CoLab current operational implementation

CoLab 1.0 is the first operational version of CoLab. Its main element is the Proxy Server, which will be used as the development basis of the future Access Server. It has been implemented on a PC with the Linux RedHat 7.2 OS. The Web server used is Apache 1.3. The software choice for developing CoLab 1.0 consists of the Java 2 SDK Standard Edition release 1.3.1_01, Jakarta Tomcat release 3.3 for the Servlets/JSP technology, and JSDT release 2.0.

The present development does not support all the CoLab functionality, but only a subset, which has been identified for testing the underlying technology. The Proxy Server implements the access control to the Web pages as well as their caching. It does not, however, implement the access control to the media and the communication tools, which have not been yet integrated within CoLab 1.0. The Proxy Server does not either fully implement the Collaboration Engine, but implements a special session configuration, where one client is configured as a Master, and the others are configured as Slaves, as far as the browsing activity is concerned. The principle is simple: any Slave visualizes the Web page explicitly selected by the Master. Slaves may enter/leave dynamically a session; when a Slave enters a session, it visualizes current Web page of the browsing session. Slaves by themselves do not have the capacity to browse on their own.

The general behavior of the CoLab 1.0 Proxy Server is illustrated in Figure 5.

Figure 5: Proxy Server behavior

Once the resource to be retrieved has been defined in the Master (a Web page identified by its URL), possibly via the browsing applet, the browser requests it. The Proxy Server retrieves, processes and caches it; while caching, it sends the resource (i.e. the processed HTML code) to the Master as the response to its request. After finishing the transmission of data to the Master, the Proxy Server notifies the JSDT Server that a Web page has just been loaded by the Master, and sends it the corresponding URL. The JSDT Server sends this URL to the Applets running in all the Slaves belonging to the browsing session, then each Applet commands its associated browser to load that URL, which makes its own request (each browser makes its own request) . Finally the Proxy Server responds to all the subsequent requests with the already manipulated and cached copy of the resource.

In order to assess the performance of the system we have measured the elapsed time between the moment at which the Master receives the first data related to a request, and the average moment at which the Slaves receive the same data. This time is less than 250 ms in our experimental platform.
5. Related works

Several approaches have been proposed for implementing Collaborative Browsing solutions [2-3], [4-10]. See [1] for a survey on this subject with the proposition of a general design framework.

The work described in [2-3] is similar to the technological approach used for implementing CoLab 1.0. The authors propose a system, called PROOF, which implements a system based on a proxy server to perform synchronous cooperation over the Web. Both solutions, PROOF and CoLab 1.0, have in common the following characteristics:

- A proxy server approach that lets the Web servers and browsers unmodified.
- The use of Java-based technology, including Java Applets to implement communication facilities inside the browsers.
- The implementation of a cache system that optimizes any subsequent loading of Web pages.

There are also some important differences between CoLab 1.0 and PROOF, for example:

- The underlying technology used by CoLab 1.0 is Servlets and JSDT.
- The cache files generated by CoLab 1.0 store the headers as well as the manipulated HTML code. So when retrieving a cached page there is no need to do any further processing.

CoLab 1.0 Applets are not inserted in every transmitted page, but loaded in the global environment; the loading is therefore made only once when entering the system for the first time.

- With CoLab 1.0, the browser has not to be configured neither with the FQDN or IP address nor the port number of the proxy. Passing through the proxy is a consequence of the hyperlink translation process.

6. Conclusions and future work

In this paper, we have defined a general-purpose Collaborating Browsing system, which, at least from a conceptual point of view, goes far beyond other proposals. It is an integrated system providing both a browsing facility and a communication facility for enhancing the collaboration among Web users. From a technical point of view, its major originality is that it relies on the proposed rule-based access control system to be integrated within CoLab.

A first operational implementation of this system, supporting only a subset of the whole functionality, has been developed and is operational on a switched LAN.

We have seen, through an informal presentation of different browsing strategies (Coordinated Browsing, Adaptive Coordinate Browsing, Location-based Browsing, ...), that the proposed system is very flexible. Several applications based on this Collaborative Browsing system may be developed in various application domains, like e-Learning and e-Business.

For the next future we will work mainly in two directions. One consists in implementing some technical features in the Proxy Server, like for example the processing of CGI-based pages, and all the related problems.

The second consists in the formal definition of the rule system that will be used by the Collaboration Engine to control a collaborative session. Based on the model adopted for this purpose, a first implementation of the Collaboration Engine will be developed.

7. References


