Joint document production server for the World Wide Web

Alain Karsenty - Bernard Merialdo
Institut EURECOM
2229 route des crêtes
06904 Sophia-Antipolis

{karsenty, merialdo}@eurecom.fr

Abstract:
We present ongoing work in the field of joint production of documents on the World Wide Web. This work fits into a larger project, W4G (Web For Groups), a European project which aims at providing a web-based asynchronous group collaboration tool. If much research has been done in the field of non-web based shared editing, there is still much work to be done to apply and develop this research for the web. In this paper, we will present the features, the implementation issues that arise during the design, as well as a sample application that has been developed.

Keywords: CSCW, WWW, groupware, shared editing.

INTRODUCTION

This paper presents ongoing work in the field of joint production of documents on the World Wide Web. This work fits into a larger project, W4G (Web For Groups), a European project which aims at providing a web-based asynchronous group collaboration tool.
Recently, much work has been done in the field of shared editing, i.e. the possibility for a set of users to edit synchronously/asynchronously a document. There are many interface and technical issues that arise when implementing such systems. Indeed, human-computer interaction has been focused on the relationship between one user and a machine, whereas in shared editing many users interact through computers. Therefore, the computer becomes more of a communication tool than a simple machine computing numbers. From the technical point of view, the designers of the system must find an appropriate architecture (e.g. replicated, centralized, hybrid) and specific algorithm that fit the needs of the application.
The research in shared editing field lead to a number of interesting results. However, applying the results to the Web cannot be done directly. When doing shared editing on the web both interface and technical issues are quite different. For instance, designing an interface based on cgi-scripts impose many constraints. One cannot do real-time updating, or drag-and-drop style of interaction (although this is now possible using JAVA).
In the next section we will review existing work, and how it relates to our system. We will then give an overview of Web4Groups and then describe the joint document server. Finally, we will describe a sample application we have implemented.

RELATED WORK

From the non-web related research, we can separate asynchronous and synchronous shared editors. Asynchronous systems clearly separate the text from comments/versions from other users, e.g. ForComments [10], Griffon [4] and Word 5.1[8]. Other systems deals with how to merge different versions of the same document. For instance, Prep [9] is a shared text editor which smartly compares different versions and shows users what has been removed/added.
Synchronous shared editors, such as the text editors GROVE [5] and SASSE [1] or the drawing systems GROUPDRAW, GROUPSKETCH [6] and CaveDraw [7], allows multiple users to simultaneously modify the document. SASSE (Synchronous Asynchronous Structured Shared Editor) also provides asynchronous features such as annotations.

As said earlier, although those systems provide interesting features and innovative implementation, they cannot be directly applied to the web. With regard to web related research, BSCW [2] allows a set of users to exchange documents in different formats. The interface provides interesting information, such as the owner of the document, what document is new, etc. The implementation is based on cgi scripts and use the PUT feature of the server to upload documents. However BSCW doesn't really deals with shared editing, it is mostly a sophisticated file server.
Another interesting approach is the definition of transformation language [3] which allows to interactively restructure HTML documents. It has been used to implement an authoring environment wired on the World Wide Web, Tamaya [11].
Finally, with regard to versioning systems, VTML [12] is a new content-type that has been proposed to communicate version information between web-browsers and clients.
From the various systems existing, there isn't yet any joint production tool available for the web.

THE WEB4GROUPS PROJECT

The purpose of the Web4Groups project is to provide, demonstrate and establish a commonly available standard service for a fluent transfer of knowledge for Internet users. The project is funded by the Commission of the European Community within the Telematics Application Program. The project started in December 1995.
The coordinator of the project is Omega Generation (Bologna, Italy). Partners of the project are organized in three groups according to their role and competence :
Developers:

- Kapsch AG , Vienna, Austria
- Stockholm University, Department of Computer and System Sciences, Stockholm, Sweden
- Institut Eurecom, Sophia-Antipolis, France
- Swiss Federal Institute for Forest, Snow and Landscape Research

Technology assessment:

- Research Unit for Socio-Economics , Austrian Academy of Sciences , Vienna, Austria

User organizations:

- European Forest Institute, Joensuu, Finland
- Swiss Federal Institute for Forest, Snow and Landscape Research
- European Information Technology Association, Gateshead, Tyne & Wear, United Kingdom
- Community of Bologna, Italy

The project is developing a service for asynchronous group collaboration. In the first phase, the service will include basic functionalities :

- asynchronous collaboration through discussions on message boards,
- links between web-pages and discussions,
- basic security,
- email and fax gateways.

The service will allow for the creation of message boards. These boards will be either public or private. The manager of the board will be able to assign roles to users corresponding to their authorization pattern. Users will browse boards and messages, and will eventually respond or extend a discussion by submitting new messages.
In the second phase of the project, some advanced functionalities will be developed :

- support for multi-language discussions,
- intelligent filters,
- joint production of documents,

- voting and rating tools.

The responsibility of Eurecom within the Web4Group project is to develop the joint production of documents service.

More details can be found on the Web4Groups WWW server at:
http://www.soe.oeaw.ac.at/w4g/Web4Groups.html

PRINCIPLES OF JOINT PRODUCTION OF DOCUMENTS

In this section, we survey the major issues that arise in the design of the joint production server, and we discuss the orientations that we have chosen. As this is ongoing work, some design decisions are not firm yet.
The role of the joint production server is to provide a facility by which a group of users can collaborate in the production of a single document. This means that users can create parts of the document, insert them in single structure, browse at other users' contributions and comment them, and retrieve part or all of the document's content. It is not in the role of the server to provide a final form of the document. Rather, it is expected that once all of the documents contents have been produced and merged by the various authors, one user (the publisher) will retrieve the whole document's content and will perform the necessary adjustments to obtain the final form.
The server maintains the current state of the document as it has been produced by the activities of the users, and provides facilities for detecting and managing conflictual behaviors (when several users modify the same section), for version management, and for annotation.
The philosophy of the design is that users are mature, so that the server will not try to solve conflicts that may arise between certain activities. Rather, it will report them to the users involved and will obey their instructions.

Document Model

We use a tree-based model for a document. A document is a tree structure which contains nodes and leaves. Nodes contain a title and an ordered list of pointers to other nodes and leaves. Leaves contain only text.
The tree structure closely follows the logical structure of the document into sections, subsections and paragraphs. We currently expect that the root of the tree contains the title of the document, and that each level of the tree corresponds to a level of headings.

Architecture

In the first implementation, the server is implemented as a proxy to a w4g server (Figure 1). Nodes and leaves are web4groups objects, and are managed by a w4g server. The user client sends requests to the joint production proxy, which interprets them and translates them into a sequence of w4g operations.

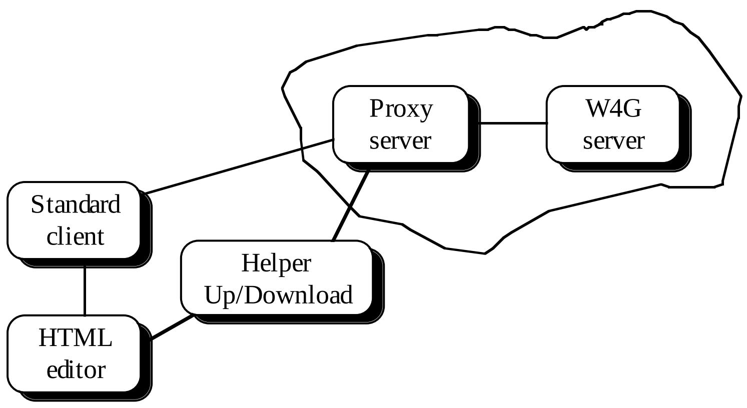


Figure 1: Architecture

Functions of the Proxy server

The basic functions of the proxy are to create, modify, and delete the document structure and contents. Because the interaction is asynchronous, the following functions are available :

- create,
- delete,
- see : allows the user to get a copy of the object for visualization only (the client interface should insure that),
- take : allows the user to get a copy while instructing the server that he intends to modify it. The server can then take appropriate actions to avoid or manage potential conflicts (explained below),
- write,

It is also possible to copy a document (or part of the document). This is useful to create frozen versions of documents.

Versions

The server maintains several versions of the objects (nodes, leaves) as they are modified. The maximum number of versions is specified for the whole document at creation time. Each time a modification is made to an object, a new version is created.
Because we trust users, the last version is always the valid one. The role of the other version is to facilitate discussions and undo function.

Merging and conflicts

An important role of the server is to facilitate the management of conflicts, when a user makes a modification over another user's modifications.

Conflict management is handled by a soft-lock mechanism:

- when a user A requests an object for modification, this object is soft-locked by the server, which means that whenever a user B requests the same object for modification, he will also receive a warning message stating that this object is being modified by A.
- as users are supposed to be mature, user B may still modify this object and submit his modifications to the server.
- when a user submits a new version of an object, this request is always accepted by the server (provided that the user has a write authorization on the document) and the soft-lock flag is reset (Figure 2).

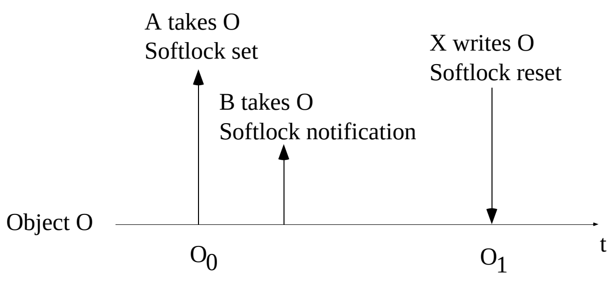


Figure 2: Soft lock notification and reset

Rule for the soft-lock:

The softlock belongs to an object and not a version.
The softlock is set when:

- a user requests an object for modification
- and the soft-lock is clear.

the softlock is cleared when:

- a user writes a new version of the object

When the softlock is set, the server also remembers the user name and the date/time of the *take* operation.

When a user submits a new version of an object, and this object has been modified by someone else in the meantime, the server creates a new version but also raises a conflict flag for this object and warns the last user (Figure 3). It is up to this user, or to any other user that will later modify this object, to solve the conflict.

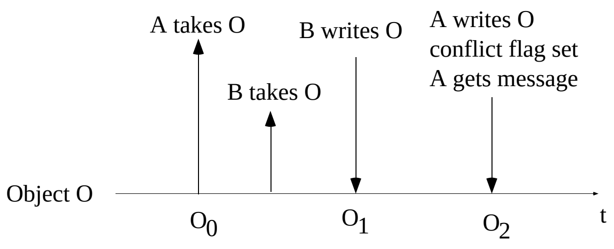


Figure 3 : Conflict flag

Rules for the conflict flag:

The conflict flag belongs to an object and not a version
The conflict flag is set when:

- a user writes a new version that is a modification of an older version than the current one

The conflict flag is cleared when:

- a user writes a new version that is a modification of the current version

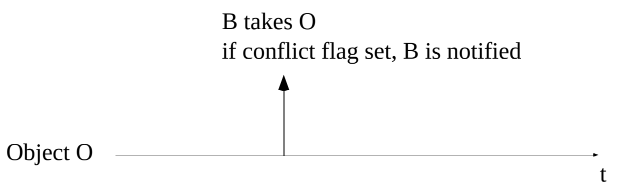


Figure 4 : Conflict notification

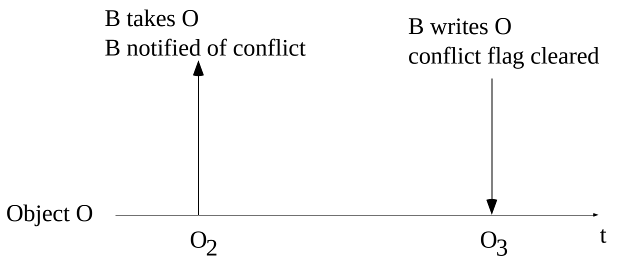


Figure 5 : Conflict flag cleared

When a user sends a *see* or *take* request, he gets the current version of the object, together with an indication of the status of the softlock and conflict flags.

Annotations

As different users produces parts of the document, it is important that the server offers a mechanism for them to express theirs comments about certain parts.
We designed a scheme to support the annotation of documents handled by the server.
One advantage of this scheme is that it allows an annotation at the word level in the document, so that it is possible to add annotations anywhere in the document.

A prototype implementation of this scheme is available. It is based on a set of cgibin scripts that receives requests from the users (through the form interface) and creates, updates and deletes local files accordingly.
The detail of the procedure is the following:

- an editable document contains a button with a label "Annotate document" that calls a script with certain specific parameters (such as the real filename of the document),
- this script will parse the document, filters HTML tags, and assigns a token number to each word of the text. the script will produce a document with the same physical layout, but where each word of text is an anchor that calls the annotation script with the proper token number.
- the user selects the word of text where he wants to place the annotation. The annotation server will answer with a form interface that contains a text area that the user will fill with the content of the annotation.
- the content of the annotation is then sent to the server which creates a new file to contain the annotation, and modifies the original document by adding an hypertext reference to this file at the proper position, as indicated by the token number. The anchor for this reference is a special annotation widget.
- by clicking on this widget, users will browse the content of the annotation. They also have the possibility to remove the annotation, in which case the corresponding file is deleted and the document is modified to remove the hypertext reference.

One advantage of this scheme is that it allows to place annotations at every word of the document. Since the modifications of the documents are done by cgibin scripts, it is relatively easy to insure that the integrity of the document is preserved (the content of the document cannot be destroyed by malicious or erroneous operations of users).

The current implementation is still an exploratory prototype that is only used for experimentation. It misses several features to serve as a production system with real users. First, it uses documents contained in files rather than Web4Groups objects. Second, there is currently no user authorization checking.

Format

As we said previously, the purpose of the Joint Document production is not to edit the document in its final form, but rather to edit the structure. Final formatting will be done separately by one user. Thus we need to have a common format for the document, which most word processors can output/input. The common format chosen is HTML. This seemed the logical choice for many reasons. It is for instance fairly easy to concatenate multiple HTML files, into one HTML file. Most word processors have or will have an input/output HTML functionnality, thus we do not provide such a function in the package.
However, we still provide the possibility to enter data in text/html/smart_text format. Smart_text smartly converts text into HTML. For instance an underline text if formatted as a Heading, indented text, is converted as List Items, we even provide the possibility to input tables.

Helper application

The way documents are saved in the server is done through the helper applications. However, depending on which HTML editor is used and how it can be parametrized, this process can be done differently.
The first solution is that when editing a section, the helper is first launched. The helper then starts the HTML editor and wait until the user quits the editor at which point the document is sent to the proxy.
The other solution is that the HTML editor is first launched, and when quitting, the helper is launched in order to save the file in the server. However, the HTML editor must be customized to be able to launch the helper application when needed.
No matter which solution is used, an application must be launched automatically when getting a file to be edited. One way to do it is to create a html/edit mime type and configure the browser to launch the application when getting this mime type (this is possible in Netscape for instance).
An important and complex work done by the helper is translating references, which we discuss in the next section.

Server document handling

The problem is how to store in the server documents which include references to other objects. If the W4G server simply stores a document, without scanning the content for references, it won't be visualized properly afterward. For instance, a gif file referenced as *file:/homes/karsenty/image.gif* won't be found when visualized by another user.
Thus, when copying file onto the W4G server, a number of operations must be done:

1. generate new names for the files
2. scan recursively for references and change them appropriately
3. when encountering the same references, do not generate each time a new file.
4. garbage collector : eliminate files that are not referenced anymore.

Issue 1 doesn't raise any difficult problem. Issue 3 can be solved by storing in a two-dimensional array the initial reference and it's new name in the server. When scanning a document, it must be checked that the reference hasn't been already scanned. However, the content of the file must also be checked since the reference to a file can point to a different content (for instance, if the reference is a meteo image that is updated every hour).
Issue 4 only raises the issue that garbage collection should be done when the system is quiescent (i.e. nobody is editing a file). Otherwise there is a risk that a referenced file will be deleted.
The most difficult problem is issue 2. It can be resolved in different ways in order to ensure that a document visualized locally can be visualized the same way by any user. The kind of references we find are of basically 3 kinds :

- absolute web references, such as http://www.netscape.com/index.html
- relative web references, such as href=«../toto.gif »
- local references, such as file://homes/karsenty/toto.gif

Absolute references do not raise any issue, except in case they are used instead of local references. For instance, one could write
 <IMG SRC = "http://www.eurecom.fr/~karsenty/toto.gif">
instead of
 <IMG SRC = "toto.gif">
This case should be handled as a relative reference.
In order to handle relative and local web references, we can have different approaches:

- pictures only mode : if we look at joint editing as a means to produce document that we can print and not as hypertext documents, there shouldn't be complex work. The only references we have to deal with is included pictures. In this case, when uploading/downloading a document, the references to pictures should be translated from one system to another, and the object that is referenced should also be uploaded/downloaded.
- absolute/relative2local : the other possibility is to transfer the references to other objects into local references in the W4G server and transfer all the referenced document in the W4G server. However, the translation should be limited to the local server, at most. For instance, I'm editing a document at Eurecom which contains the following references which are translated as (noted as an arrow -->) :
  1. Eurecom server (http://www.eurecom.fr) --> http://www.eurecom.fr
  2. my home directory at Eurecom (http://www.eurecom.fr/~karsenty/index.html) --> file:/homes/w4g/id1.html
  3. another server (http://www.netscape.com) --> http://www.netscape.com
  4. absolute files (file://homes/karsenty/test.gif) --> file://home/w4g/id2.gif
  5. relative web reference "image.gif" --> file:/homes/w4g/id2.gif

The absolute/relative2local algorithm is not trivial. It is recursive and must take into account loops which are frequent in web documents. An appropriate naming convention should also be used for storing documents in the W4G server.

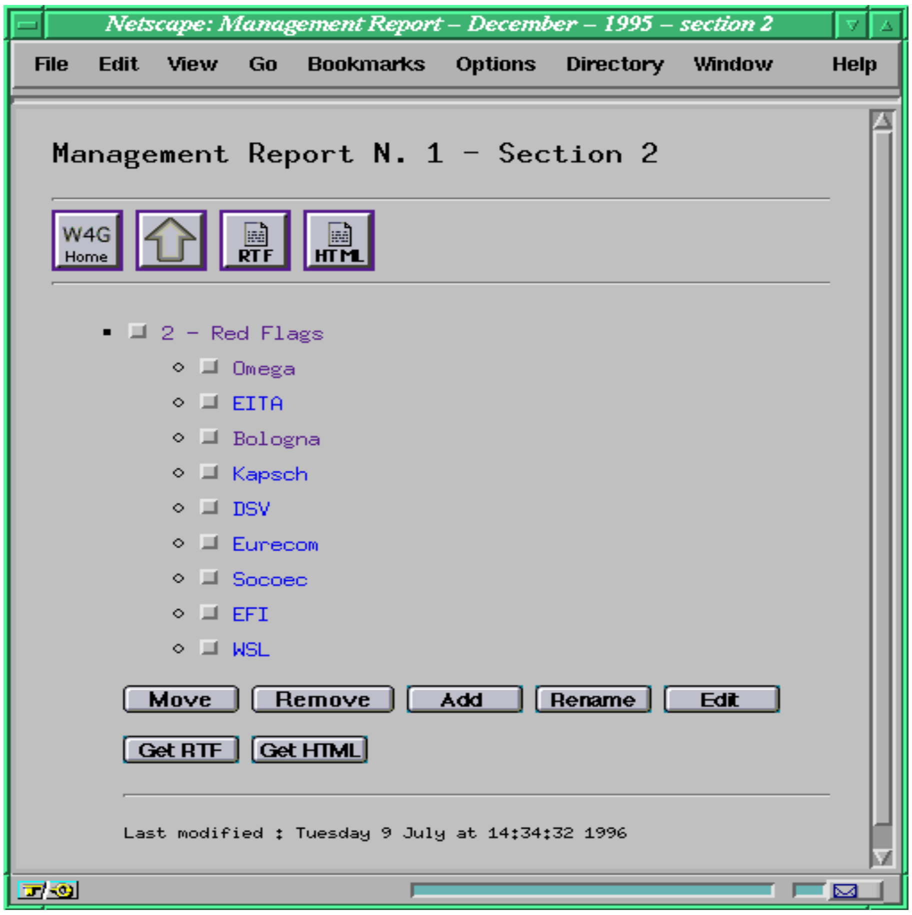
- customizable absolute/relative2local : since we cannot always guess what the user wants to translate into absolute/local, one way to figure it out is to ask him/her. When creating a hyperlink, it must be specified whether the link must be saved absolute or local. This solution however burden the user, and should only be an option.
- absolute/relative2absolute : a more simple solution, however not very elegant, consists in turning all references to absolute references, therefore it is not necessary to download the referenced objects. It is however difficult to maintain the state of a document with such a solution. The elements (pictures, hypertext links) will be dispatched at different sites and it is difficult to ensure that the elements will not move or be destroyed.

To summarize, solution absolute/relative2local is the most suited one since it allows to save hypertext documents.

A SAMPLE APPLICATION

In order to experiment with joint editing we implemented a prototype to be used by the W4G partners. Every months, partners of the project have to fill up a management report stating the activities during the month. Each management report is sent to the project manager who concatenates them into one W4G management report and send them to the project officer in Brussels.
This is a typical example of joint document production, which we took as a basis to implement a first prototype. The system, based on cgi scripts, allows the set of partners to edit the final document and visualize the document in HTML or RTF format.

When first logging in the system, the user is presented with a set of management reports for every month. The user, by clicking on the text can edit a document. When clicking on a node, the user simply goes deeper in the hierarchy, whereas when clicking on a leaf, the user is prompted with a text area which he/she can fill out to modify the content of the leaf.
With regard to cgi implementation we had two options. (1) to construct each time the pages using a cgi perl script (2) to create the whole set of pages related to a monthly report each time a modification is done. We chose to use method (2) because we wanted users to quickly navigate in the documents. This was possible since clicking on a node simply loads a page. Moreover, the documents do not contain many pages, thus re-generating the document takes a very short time.
This prototype can be tested at http://chagos.eurecom.fr:8000/



Management Report N. 1 - Section 2

- 2 - Red Flags
  - Omega
  - EITA
  - Bologna
  - Kapsch
  - DSV
  - Eurecom
  - Socoec
  - EFI
  - WSL

Last modified : Tuesday 9 July at 14:34:32 1996

Figure 6 : Editing mode for the Red Flags section of the management report

CONCLUSION

We have presented a joint document production server for the World Wide Web, part of the Web4Groups project, a European initiative for setting up non-simultaneous group communication service for the WWW. We have presented the features and the many implementation issues that arise during the design as well as a sample application that has been developed.
In the future, the user groups will help us refine the prototype. Indeed, this is a TELEMATICS European project which therefore strongly emphasize user needs. We are currently performing the user-requirement phase which will allow us to better fit to user needs. A cyclic user testing phase is also planned when the prototype is robust enough.

Moreover, we plan to implement the second prototype in the JAVA language. This will make it possible to implement a more friendly interface as well as real-time features to develop group awareness (e.g. who's editing which section at a given time).

REFERENCES

[1] BAECKER, R.M., NASTOS, D., POSNER, L.R., and MAWBY, K.L., “The user-centred iterative design of collaborative writing software,” in *Proceedings of the Workshop on Real Time Group Drawing and Writing Tools held at CSCW'92*, Toronto,Ontario, October 31 1992.

[2] Bentley, R., Horstmann, T., Sikhel, K., Trevor, J., "Supporting Collaborative Information Sharing with the World Wide Web: The BSCW Shared Workspace System" in Proc. of the 4th World Wide Web Conference, Boston, 1995.

[3] Bonhomme, S., Roisin, C., "Interactivel Restructuring HTML Documents" in Proc. of the 5th International World Wide Web Conference, Paris, 1996.

[4] DECOUCHANT, D., QUINT, V., VATTON, I., "L'édition coopérative de documents avec Griffon", dans les actes des Quatrième Journée sur l'Ingénierie des Interfaces Homme-Machine (IHM'92), Décembre 1992, pp. 137-142.

[5] ELLIS, C.A., GIBBS, S.J., and REIN, G.L., “Groupware, Some Issues and Experiences,” *Communications of the ACM*, vol. 34, no. 1, 38–58, January 1991.

[6] GREENBERG, S., ROSEMAN, M., and WEBSTER, D., “Human and Technical Factors of Distributed Group Drawing Tools,” *Interacting with Computers*, vol. 4, no. 3, 364-392, December 1992.

[7] LU, I.M. and MANTEI, M.M., “Idea Management in a Shared Drawing Tool,” in *Proceedings of the Second European Conference on Computer-Supported Cooperative Work,* September 25-27 1991, pp. 97-112.

[8] *Microsoft Word User's Guide*, Microsoft Corporation, 1992.

[9] NEUWIRTH, C.M., CHANDHOK, R., KAUFER, D.S., ERION, P., MORRIS, J.., and MILLER, D., “Flexible DIFF-ing in a Collaborative Writing System,” in *Proc. ACM Conference on Computer Supported Collaborative Work (CSCW),* , 1992, pp. 147-154.

[10] OPPER, S., "A groupware toolbox." *Byte*, December 1988.

[11] Quint, V., Roisin, C, Vatton, I., "A Structured Authoring Environment for the World Wide Web", Computer Networks and ISDN Systems, vol. 27, n. 6, pp. 831-840, April 1995.

[12] Vitali, F., Durand, D., "Using versioning to support collaboration on the WWW", in Proc. of the 4th World Wide Web Conference, Boston, 1995.