Versioning the Web

Andy Kirby, Paul Rayson, Tom Rodden, Ian Sommerville and Alan Dix†

Collaborative Systems Engineering Group, Computing, Department, Lancaster University, Lancaster, LA1 4YR, UK

†School of Computing, Staffordshire University, Stafford, ST18 0DG, UK

email: is@comp.lancs.ac.uk

Abstract Currently the display of web pages centres upon the presentation of a single instance of each page. As the web evolves to become a long-term information store (e.g. with the increasing use of Intranets), there will be a need to provide mechanisms to manage versions of web pages. Users must be able to access predictable information (e.g. the last version which they accessed) and be able to know what page versions are available and the attributes of these versions. This is a report of some initial work in this area where we have explored some of the issues of versioning the web. We have devised a simple system which allows single web pages to be converted to a version set in a transparent way (i.e. all links continue to work). This system (V-Web) maintains and provides public and private access to a set of versions for web pages and includes facilities for both page readers and page authors.

1 Introduction

The explosive growth of the World Wide Web as a medium for information publishing and dissemination has meant that authors have focused on making information available as quickly as possible without giving much thought to the problems of information evolution. Initially, the WWW was mostly geared to the publication of ephemeral information and it is still the case that most public information is of this type. However, as companies develop Intranets as a organisation-wide mechanism to access archival information, there is an increasing need to address the issues of managing the evolution of this information.

In essence, an Intranet is an internal network where Internet protocols are used to store and access information and information stores are front-ended by some web browser such as Netscape or Internet Explorer. The fundamental advantage that they offer is easy-to-use, world-wide, platform independent access to information. There is no need to replicate databases in different sites, develop different versions of database access systems for different platforms, etc. The information maintained on Intranets will change over time and different versions of the information (e.g. product descriptions) may have to be maintained. Providing support for these mechanisms through existing WWW mechanisms is likely to result in an explosion of URLs and a great deal of user confusion.

In this paper, we discuss some of the problems of providing version support in the context of the World Wide Web and briefly describe a prototype versioning system for web pages which we have developed. The work is not specifically concerned with *software* configuration management but with more general issues of providing access to versions of Web pages. We believe that this is of relevance because, for software

developers, Intranets can be used to provide access to source code, documentation, system specifications, etc.

Our work is distinct from that of Reuter et al. [1] who have demonstrated that Web browsers may be used as a front-end to a revision control system. They provide distributed revision control through a client-server system where the server is a revision control engine and the client is a web browser. They were concerned with providing access to versioned files (source code say) rather than web pages. There is clearly the possibility here of adapting it so that versioned web pages are managed by the revision control engine. However, this would certainly require some changes to their current user interface to hide the existence of the revision control system.

Issues of versioning have been largely ignored by the WWW community until recently. Vitali and Durand [2] proposed VTML, a markup language for storing document version information, but this requires a parser to sit between the web server and the document store to process all documents on the fly. Work in progress at GMD on the CoopWWW project includes version management features [3]. The Working Group on Distributed Authoring and Versioning on the World Wide Web was set up in May 1996 to address problems created by the addition of write capability to HTTP. Internet standards in this area may eventually emerge, but it isn't clear if they will cover the versioning issues we are addressing here.

1.1 Why readers need version management

It is obvious why creators and maintainers of archival information need some support for version management but less obvious, perhaps, why such a facility is useful for readers. Why should readers of web pages have to be concerned with what versions of these pages exist? Consider the following scenario.

The Communications of the ACM has been the archival repository of many of the most important advances in computing. Many key algorithms are found in its issues before the plethora of modern computer journals began. The nature of its role within computing has changed and so also has the nature of its articles - submissions must now satisfy length limitations more severe than most conference papers. For a period during this transition authors were encouraged to produce appendices and algorithms electronically and to include only ftp or web references to these in the printed text. These electronic portions of the paper were to be stored at the author's own sites. This practice is no longer supported, but for the period while this held the CACM was not an archival publication. Whereas it is still possible to read the code of the algorithms developed in the early sixties, no such guarantee can be made of code and text stored on author's own ftp or web sites. The authors may have moved to a different institution and the electronic address no longer be valid. Alternatively, the address may be valid, but the author may have updated the document. If you put a citation to such material in your own works can you be sure that others will be able to access it in the future, and if they do whether they will see the same material you do.

Springer Verlag is another crucial player in the development of computing with the authoritative and rapid publication of numerous conference and workshop proceedings in their "Lecture Notes in Computer Science" and later "Workshops in Computing" series. Economic publication of such material is getting more and more difficult hence the recent introduction of the "Electronic Workshops in Computing" series. In contrast to the above, Springer store all electronic materials at their own site thus guaranteeing the archival nature of the work (and incidentally control over access). However, in so doing they forgo many of the advantages of electronic publication, as with paper distribution no corrections can be made, neither can new developments be incorporated or referenced. This is truly an electronic facsimile of the paper mode of publication.

These two scenarios between them demonstrate the twin problems of electronic publication. On the one hand one needs to be able to reference a particular version of a document and be sure that subsequent readers will be able to trace the same text you have read. On the other hand it is useful to allow authors to correct and even update material allowing a dynamic evolving literature.

Not only is this ability to audit and trace documents necessary for academic reasons it also has legal implications. Suppose a piece of medical software fails killing a patient. The failure is traced to an obscure bug in a complex algorithm. The software developer counters a claim for damages by asserting that best practice was used in developing the program as the key algorithm was taken from the key paper in the area. However it is subsequently found that the bug present in the delivered software is not present in the (electronically) published version of the code. Is the developer negligent? Perhaps the current version of the electronically published algorithm has been corrected and the bug was actually present in the original published form as used by the developer. Does this mean we would prefer that the authors of electronically published material did not correct any mistakes in order to improve their archival status? Surely not as this would also lead to potential harm (and lawsuits).

A version managed repository can reconcile these apparently contradictory requirements. New versions of a document can be added so long as any externally referenced versions are retained. Readers of such a repository may need two kinds of reference: (i) a reference to a particular version or (ii) a reference to the most up to date version. Perhaps the role of the publisher may shift to being the guarantor of such repositories.

2 The V-Web system

When we consider the problems of providing versions of web pages, there are a number of issues which must be addressed:

1. It is unrealistic to expect web page creators to think about versioning. There must be a simple and straightforward way of converting single web pages to versioned entities.
2. Each web page can have an unknown number of links - it must be possible to convert a web page instance to a versioned page in such a way that future accesses to the original URL continue to work.
3. There must be some mechanism to inform users that they are dealing with a versioned entity and it must be easy for them to access different versions of the current page.
4. Each version of a page should indicate visually, if possible, its unique characteristics so that readers can understand the difference between versions. This is a particular problem where the differences between pages are not in the page content but in the scripts (e.g. Java applets) associated with elements on the page.
5. The point-and-click, easy-to-use, interaction metaphor of web browsers must be maintained. As far as possible, users should not have to register as readers of pages nor should they have to go through some login sequence before accessing information.
6. Several different people, with different skills and requirements, may be involved in the creation and maintenance of web pages. The versioning mechanism should support the collaborative development of WWW information.

There are also more complex issues of version management such as the management of a set of linked web pages as a single version where each page may itself be a versioned entity. This is clearly akin to software system versioning where each component of the software system may itself be versioned. So far, we haven't addressed this problem but have focused on providing simple version support.

The V-Web prototype which we have developed concentrates upon the presentation of a single instance of each page. Our long-term objective is to address all of the above problems and the current system has been written to support this research. As well as normal page browsing and linking, our system includes the following capabilities:

- Facilities to create and maintain multiple instances of the same web page.
- Facilities to present versioned web pages as a collection of pages and to access any of the pages in the version set.
- Facilities to add version instance information (owner, creation date etc.) to a specific version of a web page.
- Facilities to support for filtering and selection of versions of web pages using the version instance information.

Any version management system must be self-contained and independent of the HTTP server in its presentation of the versioned web page. In our system, existing web pages are replaced by versioned web pages so consistency is maintained in the presentation of the page contents. Readers of pages see, in essence, the page as it would appear in its unversioned form after it has been converted to V-Web format. In the remainder of this section, we expand on this by discussing the underlying model of a versioned web page and by explaining the features of the V-Web system which are available to readers and authors of web pages.

2.1 The V-Web model

Users of the Web are rightly irritated when they re-access a page and find that it no longer exists in that location. The principal design constraint on the V-Web model, therefore, was the need to maintain page access through the original URL of the unversioned page.

In our version model, therefore, the original unversioned page is replaced by a reference to a list of versions of the page which is maintained in a separate directory along with information about each version instance. So long as the link or replacement page conforms to the HTML standard then the HTTP server will present the new page in the same manner as the old page. The challenge is to ensure that accessing the link to the version set automatically results in the display of page contents rather than simply present lists of alternative pages. This is necessary as we must assume that most readers don't care about versions and don't want to be surprised when an unversioned page is replaced by a V-Web page.

Figure 1 shows how the addition of a versioned web page fits in with existing storage of web pages. In this diagram, *fourth.html* has been replaced by a V-Web page which includes 2 versions of the original page.

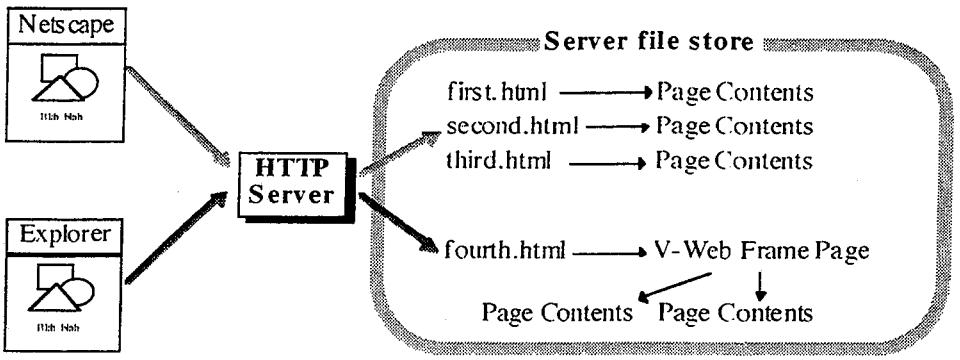

Figure 1. Replacement of existing web page with V-Web page

We achieve our objective of presenting the contents of a versioned page to readers on the initial page access by replacing the original unversioned page with an HTML frame set as shown in Figure 2. There are three frames on this page:

1. A display frame where the contents of the 'current' version are displayed. The frame acts like any other browser window, allowing links within the web page to be traversed with new pages being displayed within the frame.

2. A version information frame which provides information about each known version instance. Users may select any version from this list for display in the display frame. The display of this frame is optional so that users who don't care about versions will not normally use this frame.

   This information is stored in an descriptor file for each V-Web page. This descriptor is read either by a CGI script which produces the version instance data frame of the V-Web page, or by a Java applet which provides a scrolled list view of the information. The format of the archive file is extensible so that other information such as date and time of last view or instance comments may be added if future experiments suggest that this will be useful.

3. A toolbar frame which provides access to the functionality of the versioning system. Toolbar icons are linked to CGI or Java scripts which provide the system functionality. Again, this display is optional so that user can extend the display frame to a full screen.

Each of the V-Web Frame pages replaces the existing page contents file. This allows the HTTP server to return the V-Web page contents whenever a request is made for the original page.

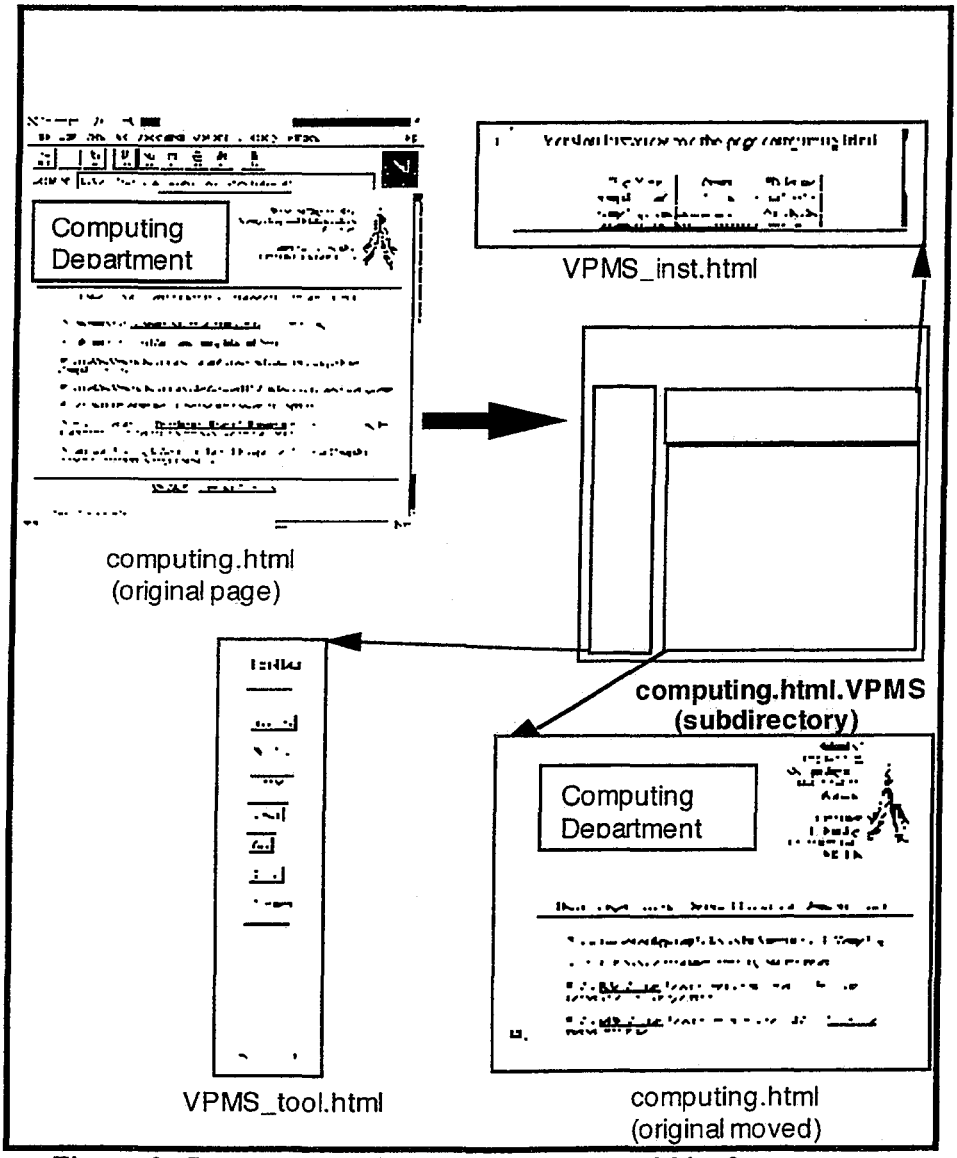

Figure 2. Presentation of the V-Web page within frames

2.2 Accessing V-Web pages

V-Web pages are accessed in exactly the same way as any other web pages by specifying their URL. When a V-Web page is accessed, the reader is presented with a display as shown in Figure 3. Within the web browser, the toolbar is displayed on the left edge, the list of versions in the top window with the largest window reserved for page display.

The default version which is displayed is the latest version of the page. Ideally, it should be possible for users to specify their own defaults but this requires users to register as 'interested' in the page. Such a mechanism is necessary for E-mail

notification of page changes and is currently being implemented. This access control mechanism, which is briefly described in the following section, will also support private default versions for page readers.

Alternative pages for display are selected by clicking on the appropriate version in the 'version instances' window. This causes the currently displayed page to be replaced with the specified version. If the displayed page contains a link to another V-Web page, we replace all of the frames in the display with the frames of the referenced page i.e. we don't attempt to have versions within versions. While this would not be technically very difficult, we believe that it would be confusing to users and would lead to display windows which were too small to be of practical use.

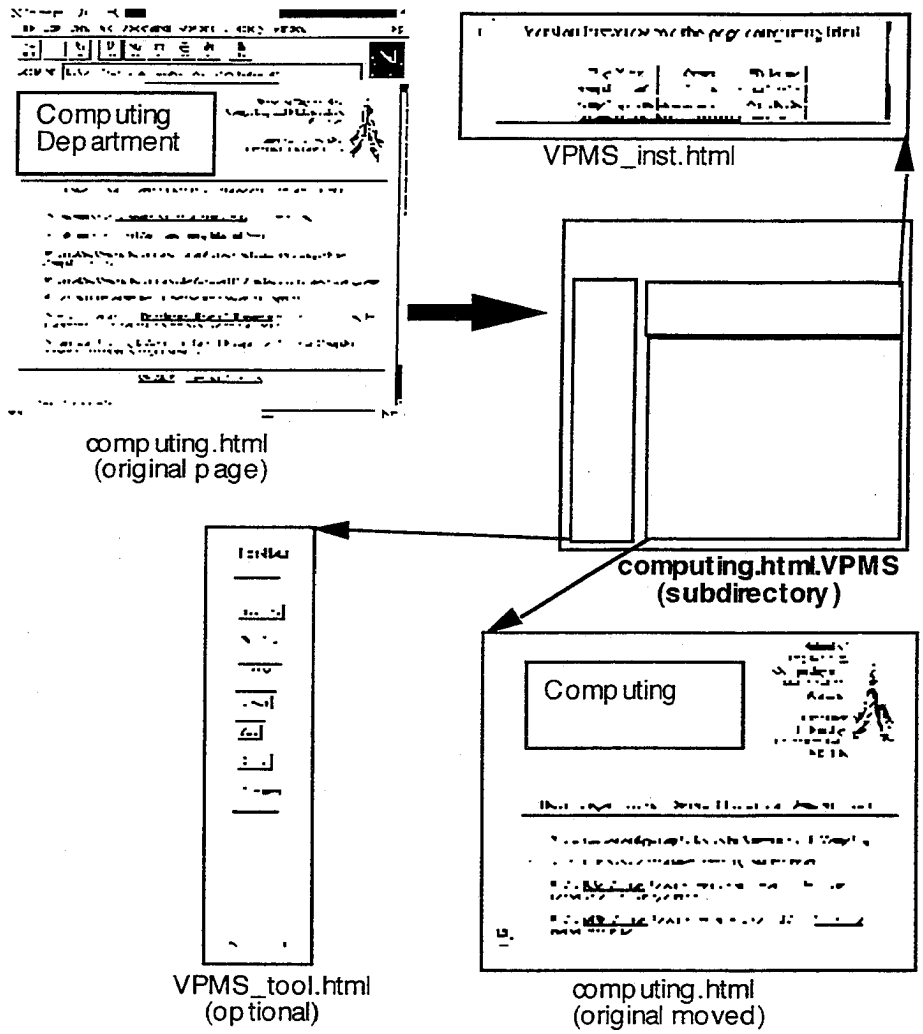

Figure 3 A V-Web page display

Readers of a page can use the top three toolbar icons for the following functions:

- Instances - provides access to filtering facilities to select the versions in the instance list and to sort the version list (e.g. by name)
- Full Page - replaces the current display by a full page version of the page in the display window. The browser facilities ("Back' in Netscape) are used to return to the version display.
- Manager - used to access the underlying version management system facilities for version creation, deletion, etc. Depending on access permissions, this may require the user to login to the management system.

2.3 Authoring V-Web pages

Authors have the same rights as readers to easy-to-use facilities which don't involve a high learning overhead. If the creation and management of versioned web pages is significantly more difficult than normal web page creation, authors will not use the system.

We assume that authors have access to some means of web page creation either through one of the web page writing programs or through direct HTML specification. The V-Web facilities are not used for initial page creation. Once a page has been designed, it is then converted to a versioned entity.

Conversion involves the following steps:

1. The author either navigates from their web browser to a control page for the V-WEB system or, from within the V-Web system, requests a new version of a page to be created.
2. The URL of the page to be converted to a versioned entity is entered in a field along with version information and display options.
3. A button is clicked and the conversion is carried out. The page is replaced by a frameset plus a page directory as discussed above.

Page versions can be managed via icons in the toolbar (the bottom 4 icons in Figure 3) which provide the following operations:

- create a V-Web page from a normal web page
- add a new instance to a V-Web page
- delete an instance from a V-Web page
- add a comment to the V-Web page (or an instance)

These operations are implemented using a CGI script written in Perl. This receives data from a HTML form on the V-Web page via a HTTP POST request, it parses the information, and performs a security check as described below. A new instance of a page can be passed to the script using the 'multipart/form-data' MIME type. Finally, the archive file is updated and a web page returned to the user describing the results of the operation. We originally hoped to implement these actions via a Java applet, but the standard security model for applets prevents reading of the client's filestore and would also require HTTP PUT to be enabled on the server.

We are currently developing access control mechanisms for the V-Web system so that it may be used to support the collaborative authoring and maintenance of web pages. This will allow new versions of pages to be developed from within the system while still retaining a stable public version set. This is becoming increasingly important as teams become involved in developing sets of web pages. It may be

necessary to integrate the work of page designers, content specialists and the site webmaster to create a page. The V-Web system means that each of them can work independently on their version of the page then integrate this work to create a new public version. Support for version integration is currently on out 'to-do' list.

Figure 4 illustrates the situation where there are several users with different version sets. Steve is unregistered and sees the public version set with page1.html as the default; Paul has registered and has set his default to page2.html; Andy has a number of private versions with his default set to newpage4.html.

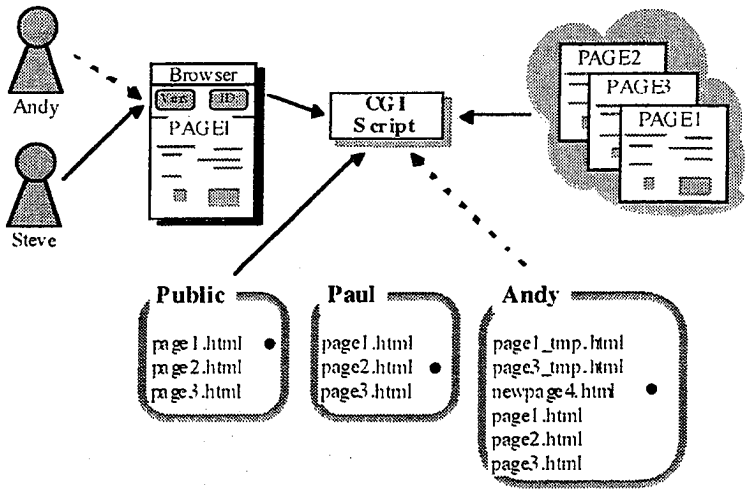

Figure 4 Supporting multiple authors

We provide a login mechanism for user registration. This currently supports individual users but group facilities will be added in a later version of the system. Page authors must login to create new versions of pages. The registration system records a login name and password, E-mail address plus information about the V-Web pages to which the user has access.

Unversioned pages can be converted to versioned pages without affecting links so the system may be used to provide a publishing environment where versioned and unversioned web pages exist side-by-side. Therefore, if we return to the journal publishing example described above, we can see how V-Web may be used to extend electronic publishing capabilities.

If an electronic journal wishes to maintain some information (e.g. appendices to papers giving detailed research results) at author's sites, these should be set up as V-Web pages. Initially there will only be a single version but, because of the frame structure, readers will be alerted to the fact that new versions exist. They will be able to register for E-mail notification when a new version has been created. As new versions are created, they are entered in the page and readers can navigate between them and check how the results have evolved.

If the publisher wishes to maintain all information, the easy access provided by the Internet means that authors can update the paper at the publisher's site. There will be no need for the publication of 'errata' correcting errors in the previous issue of some journal. The possibility exists of continuous publication where an initial version of the paper is updated over a period of time as the research is carried out. This, however, poses some problems for those sections of the academic community who must 'publish

or perish' and for the quality assurance of published results. When do new versions of a paper have to be re-reviewed and does the creation of a new version mean an additional publication?

3 Conclusions

We have been working in this area for less than 9 months so, naturally, the work is immature. The table below summarises the progress we have made against the problems identified in the introduction to this paper.

Identified problem	V-WEB solution
Converting single pages to versioned entities	Accomplished using a simple mechanism which converts the existing page to a frame set which includes a reference to a version directory.
Preserve the original URL of pages	The frame set which replaces the original page has the same URL - access causes the default version to be displayed.
Inform users of page changes and provide access to new versions	When a reader revisits a page which has been converted to a versioned page, a version list is presented. Clicking on this list causes that version to be displayed in a frame. We plan to add 'user registration' so that users may register their interest in new versions. E-mail to them will be generated automatically when a new version is created.
Visually indicate differences between versions	We have not yet addressed this problem.
Maintain easy-to-use philosophy of the Web	This has been achieved - all operations are 'point and click' and web browser functionality is maintained.
Provide support for collaborative page authoring and maintenance	Currently being implemented. We are providing user login facilities, access control, facilities for private version lists and E-mail change notification.

There is a fundamental conflict between the inherent discipline of configuration management and the inherent anarchy of the World Wide Web which must be resolved. The work described here is a small step towards such a resolution where we have provided simple facilities for web page version management which don't force versions on users and which allow versioned and unversioned pages to be mixed. Software configuration management systems provide much more powerful solutions but these are so complex that they are likely to be unacceptable to WWW users and authors who have become used to an informal style of working.

References

[1] Reuter, J., Hänßgen, S.U., Hunt, J.J., and Tichy, W.F. "Distributed Revision Control via the World Wide Web". in *6th Int. Workshop on Software Configuration Management*. 1996. Berlin.

[2] Vitali, F. and Durand, D.G. "Using versioning to support collaboration on the WWW". in *Fourth WWW Conf.* 1995.

[3] Appelt, W. "CoopWWW - Interoperable Tools for Cooperation Support using the World-Wide-Web". in *Proc. ERCIM Workshop "CSCW and the Web"*. 1996. St Augustin.