Extending Google Docs to Collaborate on Research Papers

Technical Report

Stijn Dekeyser Richard Watson

The University of Southern Queensland, Australia

Abstract

Researchers have many software options to write research papers on their own, but comparatively fewer options when they want to collaborate with others. Techniques such as Concurrent Versioning Systems and Computer Supported Cooperative Work groupware systems are still used far less often than email exchange of documents annotated with changes. We argue that the two main reasons for this state of affairs in addition to lack of exposure are (1) usability problems, and (2) the need for users to resolve conflicts. Google Docs appears to be well-placed to deal with these problems effectively; however, there are problems with this software as well, especially when used by researchers to collaborate on articles. We compare Google Docs to other solutions, and propose extensions to deal with its shortcomings. Proof-of-concept implementations of these extensions are also detailed.

1 Introduction

Collaborating on research publications is one of the core tasks of academics. In another age one would post a handwritten or typewriter-produced paper to a distant colleague and expect to receive corrections, additions, and suggestions several weeks later. After several iterations, a version would emerge that could then be submitted to a conference or journal. In today's Information Age, many researchers are employing a modern version of this technique by exchanging digital files typically via email and sometimes using more sophisticated tools such as groupware systems. Far fewer, and typically more technically astute, researchers use versioning control systems such as CVS and SVN.

One reason for this state of affairs is the lack of exposure of potential users to systems such as CVS and Computer Supported Cooperative Work (CSCW) techniques. However, we argue that there are two additional reasons for the relatively small user base of such systems. One is that collaborative systems are generally hard to setup and use. A typical setup requires installation and configuration of both server- and client-side applications. Usage requires knowledge of differences between concepts such as *update* and *commit*. Graphical User Interfaces to such systems exist (e.g. TortoiseCVS [14]) and help to a certain extent, but still require training and understanding. In addition, it is non-trivial to set up groups of collaborators for a wide range of ad-hoc projects. User accounts have to be created, and the collaborators will need access to the necessary software. These issues combined make such systems difficult to use for ad hoc collaboration projects, which are typical for academic researchers.

The second reason we believe that systems such as CVS/SVN and CSCW fail to capture a larger user base, is the intrinsic way these systems deal with the concept of conflicts. Simply stated, in each case a document is physically present at each of the collaborators, who work on it without requiring on-line communication to a server. When they decide to commit, their version is merged to the server's which conceptually is the only correct version at any one time. The merge process may or may not result in conflicts; typically, the more time between commits, the larger the chance for conflicts to occur. While limiting the time between commits to the shortest feasible time may hence reduce conflicts, it is not possible with these techniques to prevent conflicts altogether. We argue that the potential existence of conflicts, however rare, are an impediment to larger use as users are required to resolve the conflicts.

The first problem of usability referred to above can be solved by extending users' existing software with protocols and tools, such that setup is no longer a problem, and use becomes intrinsic to the user's favorite applications. An obvious target application for such integration is the web browser. Google Docs, which we discuss in Section 3 has taken this approach.

The second problem requires a different, on-line approach to document collaboration. Instead of having users work off-line locally and a server merging versions whenever an appropriate network connection is established, users can work on-line on a centrally-stored document, using a collaboration protocol as defined by the server. Such protocols may have significantly different properties, hence standardisation is needed such that clients may implement a specific protocol. Some of the protocols may still allow for conflicts to happen, while others may prevent them altogether. Also, additional tools will need to make off-line editing possible in such an environment. Google Docs employs such an on-line approach to document collaboration; however, the collaboration protocol used on the server still allows for (very rare) conflicts, and there is no support for off-line editing.

Motivation and Contribution In this paper we propose the use of a very simple system, Google Docs, and two proposed extensions to work on academic, ad hoc document collaboration projects. In addition, we compare this approach to other solutions and detail areas for improvement of Google Docs. The proof-of-concept implementations of our proposed extensions are also detailed.

2 Existing Solutions

We briefly describe a wide variety of existing solutions for collaborating on documents. Many of these are in the public domain, while some are developed at [*name of institution omitted for double-blind review*] and one is a research proposal currently lacking an implementation.

2.1 Email

Collaborators can use email to exchange versions of a document. There are a number of scenarios.

The authors may work purely sequentially. At any time, the document resides with one author who adds or edits material before emailing the new version to another author. This purely sequential approach has the advantage of being conceptually simple but has clear disadvantages, as those who do not have custody of the document are unable to contribute.

A more sophisticated approach may be to distribute parts of a task to different authors, who then email their completed section to a single "collator". While offering some concurrency, co-authors would still have to wait a significant time before viewing their colleagues' contribution. Revisions are likely to require a sequential approach.

While the email approach has the distinct advantage of being free of document format restrictions, this is balanced by very limited concurrency, and a significant manual overhead in managing versions and possibly merging many contributions into a single document.

2.2 CVS and SVN

The widely used Concurrent Versioning System (CVS) and the more recent Subversion (SVN) are commonly used to maintain repositories of text files. These are client-server applications with very similar characteristics. A single server maintains a repository of documents. An author uses a client program to *check out* a document, makes changes the local copy, then uploads the document to the server with a *check in* operation.

These systems provide good levels of concurrent working. If a document in the repository has been modified (by another author) after an author checks it out but before checking in a revised version, the system will attempt to merge the two documents. It is usually quite successful at this, but conflicting updates can occur which require manual intervention to resolve. The probability of conflict increases as time between check ins increases.

These are robust and proven systems but suffer a number of shortcomings that discourage casual use:

- The user must learn a set of commands and understand how they are used.
- The user must learn how to deal with check in conflicts. These systems do report conflicts but often it is a non trivial task to decode the conflict reports.
- The user must have client software installed on their computer.
- A networked computer must be available to host both the server software and document repository. The server must be manually configured to allow collaborators to share documents but exclude others from accessing the files. Hence, user accounts and passwords must be managed.
- Although documents of any format may be stored in the repository, only text (ACSII) based files may be edited concurrently and merged into the repository. This limits the utility of these systems for some kinds of word processor documents.

2.3 CSCW

Computer Supported Cooperative Work (CSCW) is a generic term which combines the understanding of the way people work in groups with the enabling technologies of computer networking and software services and techniques. A large body of commercial systems manage group work from a technical, managerial, and social perspective [1, 5, 10]. The level of automatic synchronization differs from product to product, but often the entire workspace is distributed at each participant's site, while each copy is kept up-to-date by interchanging appropriate control messages [6].

In addition to the problems associated to the previous two technologies described above (which are often used within CSCW systems), a distributed groupware system can suffer concurrency control problems due to events arriving out of order. Other problems are outlined in [6].

2.4 Other Solutions and Proposals

2.4.1 GOOD

GOOD [7] is a proprietary system of [*name of institution omitted for double-blind review*] and is a fully integrated publishing system that uses a single XML document to describe study material and make that material available in multiple media types. Course authors may collaborate on the material through the use of a customized XML editor. The editor is built on top of a CVS repository and hides the synchronization complexity from the user. However, the other problems associated with use of CVS and SVN still apply.

A number of systems [4, 9] similar to GOOD are used by other academic institutions.

2.4.2 ICE

The Integrated Content Environment (ICE) [13] is another system developed at [*name of institution omitted for double-blind review*] aimed for collaborating on documents, but not limited to course material. Central in ICE is the ability to author content in different word processing applications and managing collaboration through a web interface. For this purpose, ICE converts binary documents to the ICE template which is strongly based on XHTML and styles. The resulting textual format is managed by SVN to allow document collaboration. Hence, again some of the complexities of CVS/SVN collaboration is hidden from end-users, but other problems still apply.

2.4.3 XML Concurrency

An entirely different approach to document computing, as mentioned in the introduction, involves having users work on-line with special client software that communicates with a document server using a collaboration protocol. As we will see in Section 3, this is the approach taken by Google Docs. However, various kinds of collaboration protocol may exist; the Google service is character-based and does not fully exclude end-user conflict resolution. A different kind of protocol may be envisaged [12] that uses a locking protocol to guarantee serializability and is based on document semantics.

A native XML database or XML-enabled relational system offers the ability to query and modify documents stored on a server through a standardized query language. However, XML documents are almost always locked as a whole rather than specific elements, so concurrent collaboration on an XML document through an XML database is not feasible. Recently a number of researchers [2, 8, 11], however, have focused on devising locking protocols that work on the level of XML elements, and on schedulers that guarantee serializability of schedules of read and write operations on documents.

Compared to the other approaches described in Section 2, an XML concurrency control technique presents a unique and valuable property: XML documents describing non-textual information (e.g. SVG for graphics) can also be collaborated on, as long as the client software

implements the server protocol. This opens up the ability, for example, for architects and engineers to work together on blueprints.

Compared to Google Doc's collaboration protocol, the XML locking approach has the dual drawback of having a coarser granularity (element content can be much larger than the few tens of characters used in Google Docs) and being less efficient due to the overhead in maintaining locks and a transaction dependency graph.

3 Google Docs

Google Docs is a so-called "web 2.0" application. Authors edit a document held on a Google repository by using a simple browser editor developed using the AJAX methodology. Users register once with the service and can then create documents and invite collaborators who may update the document. There is also a "viewer" category of user; these can only read a document.

Changes to a document are automatically transmitted to the server; this happens at approximately 30 second intervals. If a conflict does occur, the conflicting change is reversed and the current state of the document displayed together with a message that shows the conflicting text. If necessary this can be re-applied to the document. Because of high frequency of applying updates to the repository, conflicts are very unlikely. If a conflict does occur, it is likely to be very small and hence easy to deal with.

An extensive revision history is maintained. It is possible to view the entire document as it appeared at any time past. An author can choose to revert to an earlier version. There are also tools to compare any two versions of a document.

Documents can be saved to the author's computer in a variety of formats, such as PDF, HTML, and Microsoft Word.

We have used Google Docs to collaborate on the present paper, as well as on another paper and slides for a seminar. We found the interface to be very usable, effective, and efficient. In addition, setting up collaboration with colleagues proved to be exceedingly simple.

We suggest that Google Docs is a excellent platform for ad hoc collaboration on document creation. We consider the following as the primary advantages:

- It is a lightweight application. No configuration of the author's computer is neccessary (beyond having a browser application installed). Registering a login identity with Google is required, but is a very simple step.

- Concurrent online editing works very well; multiple editors are supported and, in our experience, update conflicts are extremely rare. Indeed, it took some effort, and very precise timing, on our part to trigger an update conflict during our evaluation of the system.

4 Google Docs Shortcomings

As mentioned in the previous section, we have used Google Docs in a collaborative setting on a small number of texts and have had a mostly positive experience. However, there are some shortcomings, large and small, that deserve attention.

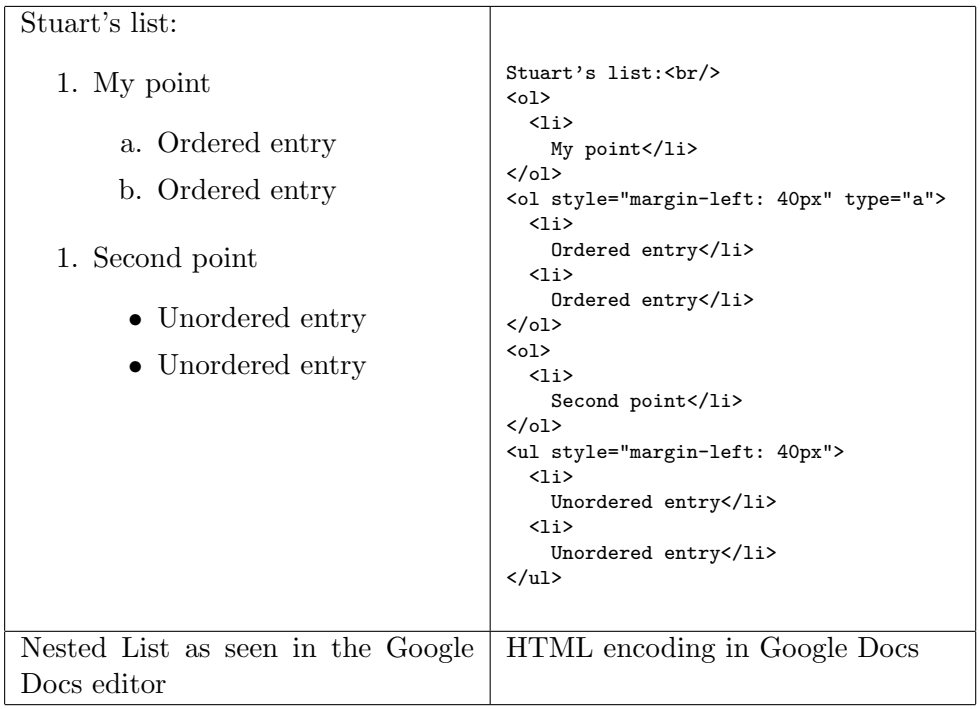

Figure 1: Google Docs Editor Problems.

Editor

Google Docs implements an easy to use, fast, and comprehensive HTML editor that mimics the functionality of a word processor application. However, as it does not construct XHTML and does not use a a clean template such as ICE does (see Section 2.4.2), there are numerous problems with the output, whether shown as a webpage, exported to a word processing document, or as PDF. A clear example of this problem is shown in Figure 1. When creating and modifying nested ordered and unordered lists, Google Doc's underlying HTML code quickly loses the implied semantics and mixes in style directives. Users can edit the underlying HTML directly, but this solution is fraught with problems. We partially solve this problem in Section 5.1.

Academic requirements

Academic papers and reports typically contain mathematical formulae, citations, figures, tables and a bibliography. None of these are supported natively in Google Docs; Section 5.1 describes our approach to providing these features.

Output

Google Docs documents can be exported as word processing files, HTML, or PDF. However, the output layout is very hard to control. Conceptually, a CSS file can be associated to the HTML, but this is often insufficient for academic papers, where a complex style is required by the publisher. Again we propose solutions in Section 5.1.

Off-line support

As mentioned in the introduction, Google Docs currently does not support off-line editing of documents and subsequent merging with the on-line copy. Researchers often work on articles while traveling, so this constitutes a real restriction. We propose solutions in Section 5.2.

Vendor independence

To gain the advantages of Google Docs one must use the browser based editor. This may be at best annoying and at worst unacceptable to those accustomed to more powerful editors. The availability of an API that allowes network applications to communicate with the server would enable deployment of non-Google editors.

Another vendor-related issue is that source documents reside on a Google server. The authors of the document have no guarantee that access to their documents is restricted to themselves.

Text-based only

Finally, the collaboration protocol used by the Google Docs service can be applied only to textual documents. It cannot be used for collaborating on graphics or other content. We described a different approach, XML concurrency control, to solve this problem in Section 2.4.3.

5 Extending Google Docs

We now propose solutions to some of the shortcomings described in the previous section. Proof-of-concept implementations of these solutions are available at [3].

5.1 Automatic Conversion to LaTeX

In order to solve Google Docs' problems with control over layout, referencing, and mathematical formulas and make the tool useful for ad hoc collaboration between researchers working on scientific articles, we propose a web service that converts a Google Doc to a LaTeX-generated PDF.

The web service takes two URLs (one for the main article and one for a BibTeX document both collaborated on through Google Docs), a list of settings, and a group of LaTeX style- and package files and returns a PDF, the LaTeX version of the article, and the BibTeX file. The web service process flow is illustrated in Figure 2. We briefly detail important parts of the process.

Importing Google Docs To allow the web service to convert the article stored on the Google Docs server to LaTeX, and in the absence of an API for Google's service, the text and its accompanying BibTeX document must be made accessible over the web. This requires the user of the web service (or any collaborator) to first publish the document through the relevant Google Docs interface. Publishing means that the document receives a unique URL and can be viewed by anyone. Hence, the web service can retrieve the documents through the user supplying the appropriate URLs.

Initial Cleansing The published document conforms to HTML 4.0 Transitional, and contains a number of Javascript functions (both inline and by reference) and also style information

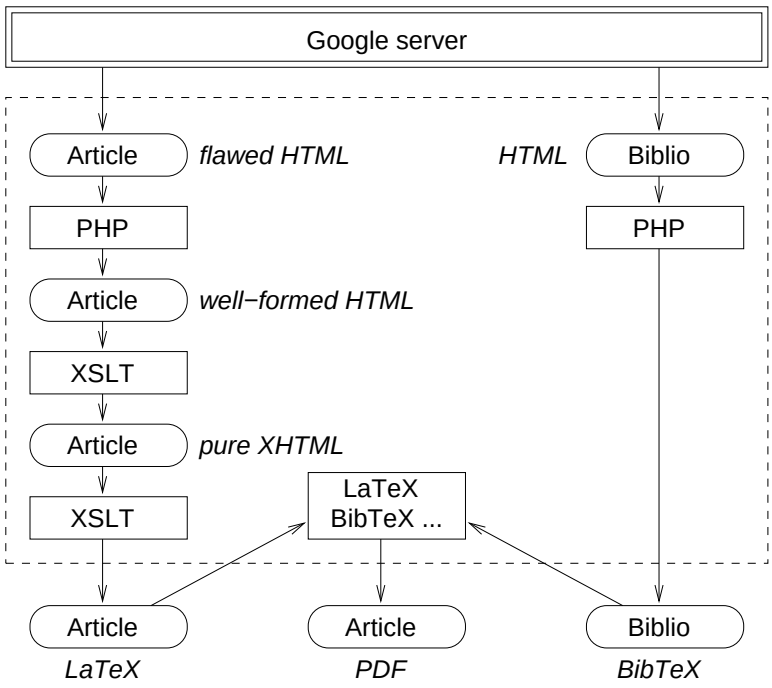

Figure 2: Conversion to LaTeX.

embedded in the HTML tags. In other words, the source is not usable for processing with an XSLT stylesheet. Thus, the PHP web service script first cleanses the HTML and generates well-formed HTML not containing functions or style.

Canonicalization As described in Section 4, the Google Docs HTML editor web interface is susceptible to introducing logical errors in its nested lists and other structures. Hence, an XSLT push stylesheet is employed to generate a canonical representation of nested lists. In our proof-of-concept implementation this stylesheet has only basic functionality, but its improvement should be relatively straightforward.

Figure 3 illustrates the working of this component given the example shown in Figure 1.

LaTeX Transformation The central component of the conversion process involves another XSLT stylesheet which transforms XHTML to LaTeX. The push stylesheet uses a relatively large number of constituent templates which translate a specific XHTML structure into the equivalent LaTeX code. Lists and sections are supported fully, and basic tables (not containing cell-spanning etc) also convert successfully. Fonts, font sizes, full justification and color are essentially ignored, although center alignment, bold and italic are supported.

Mathematical formulas entered in Google Docs using LaTeX encoding is copied as-is, and subsequently processed by the LaTeX compilation phase. A similar treatment is reserved for citations and intra-document cross-referencing, giving scientific authors full control and allowing them to utilize existing LaTeX skills fully. The collaborating authors share BibTeX bibliography entries in a second Google Doc file and hence have full power over the entries as well.

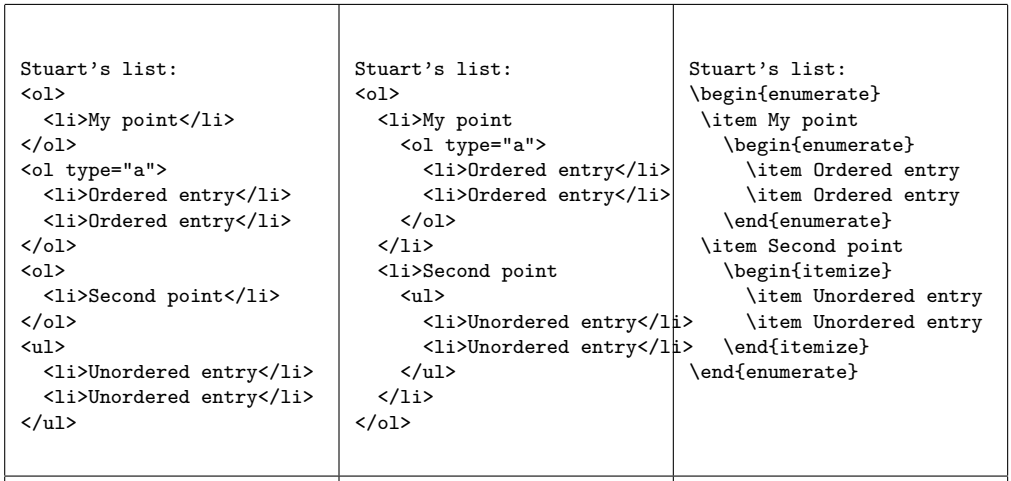

Figure 3: Conversion Process for Nested Lists.

Security Issues Since LaTeX is a Turing Complete language, the `settings` parameter passed to the web service may contain malicious code (e.g. infinite loops or file IO). This code would be run by the web service during the LaTeX compilation phase. Therefore the service should be protected in a number of ways. There should be user authorization and event logging, and the web server itself should be an isolated virtual server. An alternative is to make the service available as a stand-alone program run by end-users rather than as a web service.

5.2 Synchronising Off-Line Documents

For a variety of reasons a collaborator may wish to work off line. For instance she may be travelling and, although she takes her laptop computer so she can continue to work on a joint paper, will have only occasional access to the Internet. The current document can be downloaded from Google Docs, but no facility exists to merge a local document with the current document. Simply replacing the server copy with the altered local copy will overwrite all changes made by collaborators in the intervening time.

However, a merge procedure can be implemented using a handful of simple Unix tools.

Figure 4 shows the complete process of working offline. There are three phases to performing offline document update. First the current document is downloaded, then the user edits a working copy of this document, and finally the working copy is merged with the (possibly updated) current repository document.

Two copies of are made of the downloaded document; one (doc_0) is retained unchanged to facilitate the merging process, while the working copy (local) can be edited arbitrarily.

When the local copy is to be merged with the server document the following steps are followed:

1. Using `diff`, create a patch file describing the local changes to doc_0.

2. Download doc_1, a fresh copy from the repository (it may have changed from doc_0).

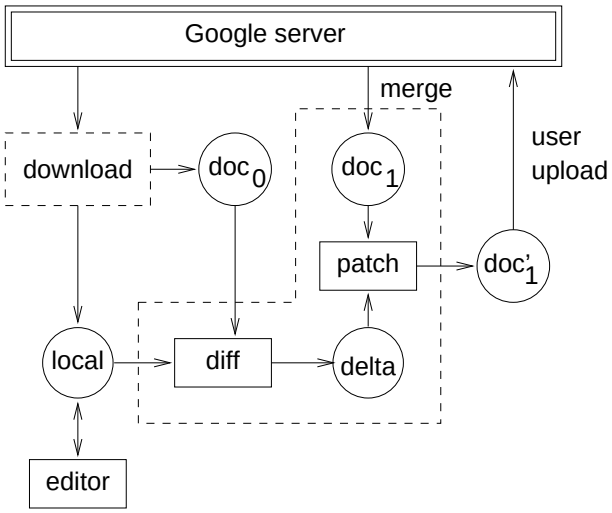

Figure 4: Working offline.

3. Using `patch`, merge the local changes into the current local version of the document; this creates an updated local document doc'$_1$.

4. Upload doc'$_1$. This involves the awkward, and unfortunately manual, step of deleting the contents of the current Google document, and pasting the contents of doc'$_1$ into the browser window. Clearly this will be unsatisfactory for large documents. An API into the Google service would ameliorate this problem.

Another limitation here is that no other collaborators should perform updates between steps 2 and 4.

The merge and upload phase would be much simpler if the Google Docs editor could be run offline and write a journal of update transactions which could be later uploaded and applied by the server. (This may not be much of an extension as in normal online mode it would appear that update transactions are created, though they are transmitted immediately to the server.)

6 Conclusion and Future Work

We have described our experience in using Google Docs as a platform for ad hoc collaboration on scientific papers and have compared this approach to other existing techniques. We then detailed some of the problems use of Google Docs currently implies, and proposed extensions that increase the software's usability for researchers.

As Google has announced that it will make an API to its service available in the future, we will be able to modify our extensions such that their use will become easier and more robust.

References

[1] *Proceedings of the ACM 2006 Conference on Computer Supported Cooperative Work*, Banff, Canada, November 2006. ACM.

[2] S. Dekeyser, J. Hidders, and J. Paredaens. A transaction model for XML databases. *World Wide Web Journal*, 2004.

[3] S. Dekeyser and R. Watson. Google Docs to LaTeX. Website, 2006. `http://www.sci.usq.edu.au/research/googledocs.php`.

[4] S. Dobratz. Thinking the long term: the XML-based publishing workflow for handling electronic theses and dissertations at Humboldt University. In *ETD2005: Evolution Through Discovery; the 8th Int'l Symposium on Electronic Theses & Dissertations*, Sydney, September 2005.

[5] D. Eseryel, R. Ganesan, and G. S. Edmonds. Review of computer-supported collaborative work systems. *Educational Technology & Society*, 5(2), 2002.

[6] S. Greenberg and D. Marwood. Real time groupware as a distributed system: concurrency control and its effect on the interface. In *Proceedings of the ACM conference on CSCW*, pages 207–217. ACM Press, 1994.

[7] J. Radajewski and S. MacFarlane and S. Dekeyser. GOOD Publishing System: Generic Online/Offline Delivery. In *Ninth Australasian Document Computing Symposium*, Melbourne, December 2004.

[8] Kuen-Fang Jea and Shih-Ying Chen. A high concurrency XPath-based locking protocol for XML databases. *Information & Software Technology*, 48(8):708–716, 2006.

[9] U. Muller and M. Klatt. An XML-based publishing platform. In *ETD2005: Evolution Through Discovery; the 8th Int'l Symposium on Electronic Theses & Dissertations*, Sydney, September 2005.

[10] J. F. Nunamaker, Alan R. Dennis, Joseph S. Valacich, Douglas Vogel, and Joey F. George. Electronic meeting systems. *Communications of ACM*, 34(7):40–61, 1991.

[11] Peter Pleshachkov, Petr Chardin, and Sergey Kuznetsov. A DataGuide-based concurrency control protocol for cooperation on XML data. In *ADBIS*, pages 268–282, 2005.

[12] S. Dekeyser. Towards a new approach to tightly coupled document collaboration. In *Ninth Australasian Document Computing Symposium*, Melbourne, December 2004.

[13] Peter Sefton. The integrated content environment (ICE). In *AusWeb06: The 12th Australasian World Wide Web Conference*, Noosa, Australia, 2006 2006.

[14] TortoiseCVS. Enjoyable version control. Web Article, 2006. `http://www.tortoisecvs.org/`.