

- [3] R. Cole, P. Eklund: Analyzing an Email Collection using Formal Concept Analysis. *Proceedings of the European Conf. on Knowledge and Data Discovery*, pp. 309-315, LNAI 1704, Springer, Prague, 1999.
- [4] R. Cole, G. Stumme: CEM - An Email Analysis Tool. *Proceedings of the 8th International Conf. on Conceptual Structures*, pp. 309-315, LNAI 1704, Springer, Darmstadt, 2000.
- [5] R. Cole, P. Eklund and G. Stumme: CEM - Visualization and Discovery in Email, *Proceedings of the European Conf. on Knowledge and Data Discovery*, pp. 309-315, LNAI 1704, Springer, Prague, 1999.
- [6] R. Cole, Using Force Directed Placement and Genetic Algorithms for Concept Lattice Layout, *Proceedings of Australian Computer Science Communications*, Los Alamitos, CA, 2000. IEEE Press, 2000.
- [7] Michael K. Coleman and D. Stott Parker. AGLO - Publications and Implementation. *Software - Practice and Experience*, pages 1415-1438, December 1996.
- [8] R. Cole, P. W. Eklund, D. Walker: Using Conceptual Scaling in Formal Concept Analysis for Knowledge and Data Discovery in Medical Texts, *Proceedings of the Second Pacific Asian Conference on Knowledge Discovery and Data Mining*, pp. 378-379, World Scientific, 1998.
- [9] B. Ganter, R. Wille: *Formal Concept Analysis: Mathematical Foundations*. Springer, Heidelberg 1999 (Translation of: Formale Begriffsanalyse: Mathematische Grundlagen, Springer, Heidelberg 1996)
- [10] R. Godin, Gecsei, J. and Pichet, C: Design of a Browsing Interface for Information Retrieval, *SIG-IR*, pages 246-267, 1987.
- [11] R. Godin, and Missaoui, R. and Alaoui, H. Incremental Concept Formation Algorithms based on Galois (Concept) Lattices. *Computational Intelligence*, Vol. 11, number 2, pp. 246-267, 1995.
- [12] G. Stumme: Hierarchies of Conceptual Scales. *Proc. Workshop on Knowledge Acquisition, Modeling and Management*. Banff, 16.-22. October 1999
- [13] R. Wille: Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (ed.): *Ordered sets*. Reidel, Dordrecht-Boston 1982, 445-470
- [14] R. Wille. Line diagrams of hierarchical concept systems. *International. Classification*, 11:77-86, 1984.
- [15] P. Wille: Conceptual Graphs and Formal Concept Analysis. In: *The 4th International Conference on Conceptual Structures*. LNAI 1257, pages 2-18, Springer Verlag, 1997.
- [16] F. Vogt, C. Wachter, R. Wille: Data Analysis based on a conceptual file. In: *Classification, data analysis and knowledge organization*. pages 131-140, 1991.

## Implementing Shared Document Preparation with Lightweight Editing

Michael J Rees

School of Information Technology  
Bond University  
Qld 4229, Australia

mrees@bond.edu.au

### Abstract

Virtually all web pages are read-only, yet the first web browser allowed users to read and edit every page. Special ad-hoc mechanisms are needed to make all or part of a page editable by a user. This paper describes Pardalote lightweight editing, a document management feature for allowing many users to share the editing of a web page using only a web browser. A brief overview of how Pardalote is implemented is followed by examples of shared document preparation using Pardalote. The benefits of such web document management are discussed. Future Pardalote extensions using XML precede the closing remarks.

**Keywords** Shared document management, cooperative document preparation, lightweight editing, I-grains, fraglets, user interface design, computer supported cooperative work.

### 1. Introduction

Tim Berners-Lee describes the development of the World-Wide Web and the first browsers and servers in [1]. In 1990 the excellent design of the NextStep operating systems running on the Next machine made it very straightforward to allow any web page displayed by the web browser to be edited in-situ. Only when the pressure mounted to provide browsers on several other hardware platforms was the in-browser web page editing feature abandoned. Those second stage browsers simply displayed web pages and set the browser model that still holds today.

Although the original Mosaic browser thought to provide us with annotation capability, it was not until 1995 that MIT organised a meeting [2] to discuss methods for making web pages into a collaborative medium. Many large software companies and large research projects were represented there. Each paper presented a different mechanism to achieve collaboration via the Web. Several of these solutions are still available today as commercial products.

In the same year, 1995, the working group that was to produce the WebDAV interoperability specification, World Wide Web Distributed Authoring and Versioning [3], was formed. WebDAV [4] is an extension to the HTTP protocol that supports web document metadata, namespace management (like file system directories), overwrite protection, and versioning management. In effect, WebDAV allows web documents (any file that has a URL) to be edited asynchronously by many users, providing collaborative authoring. Of course, WebDAV support must be built into browsers and servers, a not inconsiderable implementation effort. Once achieved, however, this will provide access to genuinely collaborative web documents at last.

In the view of the author at the time of writing, WebDAV offers considerable benefit in the longer term but in the context of this paper, at the cost of a heavyweight solution in terms of software implementation. Users wishing to collaborate via web pages using WebDAV will need to wait for its use to become widespread.

A lightweight solution to collaborative web document editing is one that can use existing browsers and servers. To be truly lightweight, the solution must be a convenient one for all types of users involved:

- Members of collaborative teams who wish to jointly prepare and share web documents, the ultimate end-users
- Original authors of the collaborative web documents
- Web site administrators who install the software that makes collaborative editing of web documents possible

Probably the best example of such lightweight editing is the work of the Sparrow Project [5] from Xerox PARC. The author in [6] and [7] has described examples of Sparrow's capability. The first user type, end-users, is very well catered for in Sparrow. The user is presented with a very simple and intuitive user interface to edit nominated sections of the web document (HTML page). Edits are restricted to the textual content in specially marked locations.



Although the Sparrow user interface is very simple, CGI scripts are used to update the web document contents as editing proceeds. This entails a round trip to the web server for every update, and is thus inevitably slow in response over congested Internet links. Sparrow document authors (type 2 users), who wish to incorporate editable sections in their pages, must learn and use special tags, which are interpreted by the Sparrow CGI scripts. The author of this paper determined to adopt the simplicity of the Sparrow user interface and to overcome the response time problems. This led to the Pardalote project using a new approach to implementing lightweight in-situ editing. At the same time, Pardalote addressed the issues of web document authoring to make it extremely straightforward to incorporate lightweight editing in specified parts of the document.

WebDAV and Pardalote represent two end of a spectrum of in-situ editing of web documents, from heavyweight to lightweight. Before describing Pardalote in detail from Section 3 onwards, some middleware solutions are discussed in the next section.

## 2. Other middleware approaches

In the middleware-editing category, an impressive Australian solution called EditLive [8] is worth mentioning. End-users are presented with web documents with editable sections. A reasonably rich DHTML editing tool is provided, so that user can make sophisticated, formatted edits of web document content to which they are given access.

In the context of this paper, EditLive is a middleware solution because it requires end-users to download and install the editing software. While installation is very straightforward, this additional install step can lead to problems for some users.

EditLive software on the client exists in the form of a browser plug-in specifically implemented for a particular browser. On the Microsoft Internet Explorer (IE) platform, EditLive makes full use of the DHTML editing component. This provides the user access to sophisticated, WYSIWYG editing with control of features like:

- Fonts, font sizes, colours and weights
- The standard HTML styles like headings, paragraphs and lists
- Tables
- Image insertion and placement

The end-user need not know HTML and is presented with a word-processor-like user interface. This can be used in a very simple way, but the more advanced formatting features place the end-user editing into the middleware category. In addition, the DHTML editing control does not possess all the usual editing conventions, seemingly lacking the simple use of the Backspace and Delete keys to delete characters adjacent to the cursor.

Web document authors have some work to do. While simple web page templates can be provided, authors need to understand scripting to design their own templates. The scripts call upon the client-side EditLive plug-in objects to activate the editing components. While not difficult, scripting is beyond many web page authors.

EditLive uses File Transfer Protocol to send the new edited information back to the web server in order to update the file. Web page authors and web site administrators are provided with settings and tools to control which end-users have this access and for which files. Again, this requires that these users have suitable knowledge and training in order to utilise the required access control.

Mention should be made in this section of a couple of in-situ annotation solutions that effectively provide the end-user with a simple editing facilities within web pages. Microsoft Office 2000 provides the Office Server Extensions (OSE) [10], which run with the Microsoft IIS web server. When using IE to view a web page resident on IIS with OSE installed, the end-user selects the online discussions tool built into IE. This then presents a view of in-document discussions as shown in Figure 1.

Using special file and folder icons the OSE discussion boxes are attached to the paragraph of text to which the comments refer. Each comment is tagged with the author, date and time. Other users may reply to comments, and edit and delete their own comments.

While the browser gives the impression the discussions are integrated into the web page, the discussion text is actually stored in a SQLServer database on a machine on which the OSE are installed. The discussions database can be stored on any machine anywhere on the Internet. When IE prints the web page, the discussions are interleaved with the contents of the page. Another feature of OSE not relevant here is the ability for users to be informed by email of any discussion changes made by other users.

Once again, OSE presents end-users with a lightweight editing task-only text is entered and modified. Web document authors have the easiest task of all; they simply arrange their documents to be part of a web site that grants end-users access to OSE

discussions. Web site administrators have to install OSE and manage user accounts in order to grant access to the discussions facility, and arrange for routing email notifications.

Another excellent Australian software development in the area of middleware editing is the PageSeeder package [9] from Weborganic Pty Ltd. What is now PageSeeder grew out of the need to allow remote learners the opportunity to enter discussions into shared web pages as part of their online education materials. For simplicity, PageSeeder originally allowed users to read in-page discussions entered by other users, and enter their own discussion points by submitting email requests.

In its latest commercial form, PageSeeder allows end-users to click on the proprietary seed symbol, and enter their comments/questions/answers at that point in the document. A new browser window containing a suitable form pops up for this entry process.

For each seeded entry, PageSeeder add an additional line into the web page at the end of the seeded paragraph. The line contains the subject of the entry as a hyperlink together with author and date information. Clicking on the hyperlink pops up another window containing the full text of the entry together with menus for creating new entries and replying to the current entry, and so on. This view is shown in Figure 2.

Each user interaction with PageSeeder requires a round trip to the web server, and is implemented using Java applets embedded in the web page. These applets operate in both the major browsers and control which end-user can view and/or edit the seeded comments. Web page authors must learn the appropriate additional HTML tags to insert the applet objects and set their parameters. PageSeeder provides web page uploading and downloading for those users with page

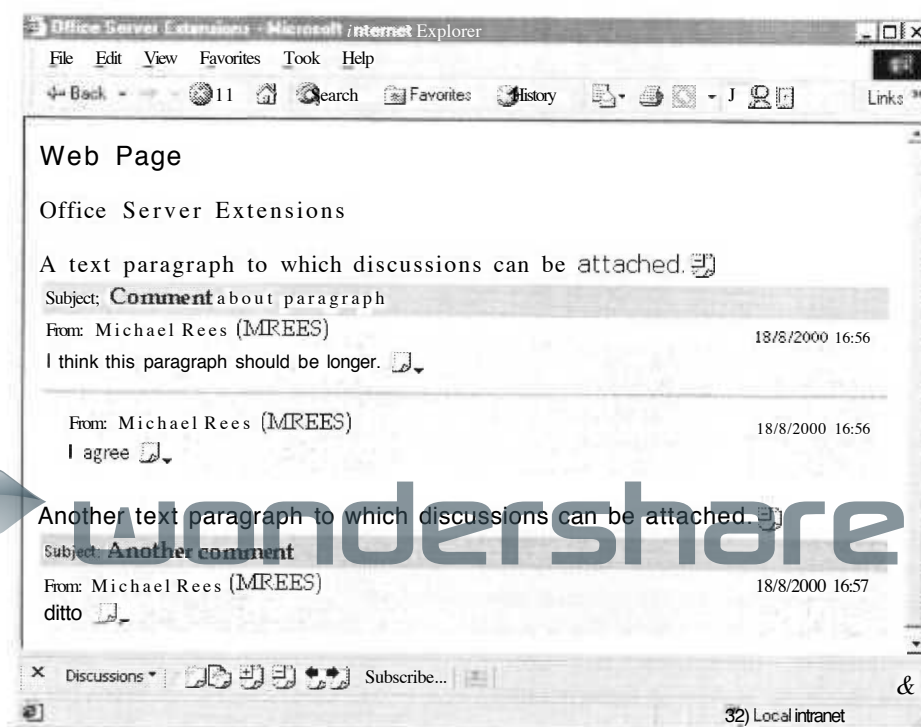


Figure 1. Microsoft Office Server Extensions with in-document discussions.



Figure 2. PageSeeder seeds and comment expansion.

authoring permission. This makes page authoring a middleweight solution, as is the web site administration. Details of the latter will soon be available at the Weborganic web site.

### 3. Lightweight editing with Pardalote

Pardalote has been expressly designed to offer a lightweight solution to all three categories of users: end-users, page authors and web site administrators. This sets it apart from all in-situ editors mentioned so far. Version 4 of Pardalote described in this paper achieves this aim admirably. Being an experimental system, however, this aim is achieved at the cost of making Pardalote browser-specific—in this case Microsoft Internet Explorer only.


Pardalote 3 is described in [7] and incorporates these ultra-lightweight features:

- All in-situ web page edits are achieved with instant response times without requiring a web server round trip—an extremely simple and responsive user interface for end-users
- After linking to a CSS style sheet and adding one `<script>` tag, editable sections of web pages are simply marked up using style sheet classes—page author training is minimal
- One simple ASP web page containing less than 10 lines of script is needed to update edited web pages as part of a small FrontPage web. installed once—web site administration is minimal


This third version of Pardalote uses Microsoft DHTML behaviors, a proprietary feature of Cascading Style Sheets. This slowed the downloading of pages containing large numbers of Pardalote editable sections. In addition, an edited Pardalote 3

page adds additional, hidden HTML tags that make subsequent editing with commercial editors like FrontPage more difficult. (A similar problem is exhibited by all middleweight solutions mentioned above.)

Pardalote version 4, described in this paper, has been completely reimplemented avoiding DHTML behaviors, and using asynchronous update for no-wait saving of edited pages. Also, no hidden HTML tags are used, so that once pages have been edited with Pardalote, they remain easily editable by editors like FrontPage and others.

For comparison, Figure 3 shows the user interface for editing paragraphs. Pardalote uses the  symbol to indicate which parts of a page can be edited by the end-user. For improved consistency in version 4, all parts of the page controlled by Pardalote have a light blue (grey in black and white) background.

In Figure 3 the user has just clicked on the text of the first bullet point to bring up the edit panel (this occurs instantly since all the HTML is generated within the browser scripts). The text of the list item can be edited within the text box form element. Clicking on the Save button changes the page to reflect the new contents, closes the edit panel, and returns the page to its original format.

Each part of the page marked with , is called an I-grain (short for information grain). I-grains can be any text tag such as headings, paragraphs, list items, table cells, addresses, and so on. Every I-grain can be edited, and there can be any number of I-grains on each page. This allows great flexibility in the layout of the page for shared editing. Page authors can impose rigid structure so the page remains readable and consistent with other pages in a web site. On the other hand, very little non-editable text may be used to

allow the group sharing the editing to make significant changes to page layout as modifications are made.

Note the range of editing operations:

- New I-grains can be created from any existing I-grain. The new I-grain is inserted after the current I-grain.
- I-grains can be moved up and down in the page any number of times.
- An I-grain can be permanently deleted from the page.
- An I-grain can be frozen. In other words, it ceases to be an I-grain and becomes text that is static and can no longer be edited by Pardalote.

Pardalote 4 web page authors need only add the following two lines to the `<HEAD>` tag of a web page to switch on editing capability:

```
1. <LINK id=lweStyle
   href="/lwe/lweStyles.css"
   type=text/css rel=stylesheet>
2. <SCRIPT language=JavaScript
   id=lweScriptInclude
   src="/lwe/lwe.js"></SCRIPT>
```

Previously, the web site administrator will have created the /lwe FrontPage web that contains:

- Pardalote style sheet, `lweStyles.css`
- Pardalote lavascript page, `lwe.js`
- Image files and help pages

In total, the Pardalote files occupy a lightweight 46 Kbytes of disk space. Of course, the web site manager must set write permission for the default web server user account in all those folders holding Pardalote pages. This allows the Pardalote page files to be updated by the ASP script.

Once the style sheet is linked to the page, the FrontPage editor shows the class in the style list box. Thus to make a piece of text editable, the user just selects the appropriate style. For the list item I-grain in Figure 3 the Pardalote style class is `lwePara-f` defined in the style sheet.

Style classes are provided that provide the page author with different combinations of the above editing operations. For example, if the end-user is intended only to modify an I-grain, then the page author applies style class `lwePara-e`. About a dozen combination style classes are supported for paragraphs.

All Pardalote page edits are temporarily stored in memory of the end-user's browser. To make the changes permanent and visible by other users, the Update page button in the control panel inserted at the top of each page (and at the bottom) must be clicked. This sends the complete HTML of the page, encapsulated as XML, to the web server, where an ASP script overwrites the original page. The update operation is implemented asynchronously, and the user can continue to edit the page while the update takes place.

Pardalote is designed for small collaboration teams where all users know and trust each other. All users sharing a web page have equal privilege to insert, change, move and delete any I-grain. A simple check is made in the ASP script to detect whether another user has updated the shared page while the current user has been making changes I-grains. If this happens, the update does not take place, and a warning message generated. For typical small

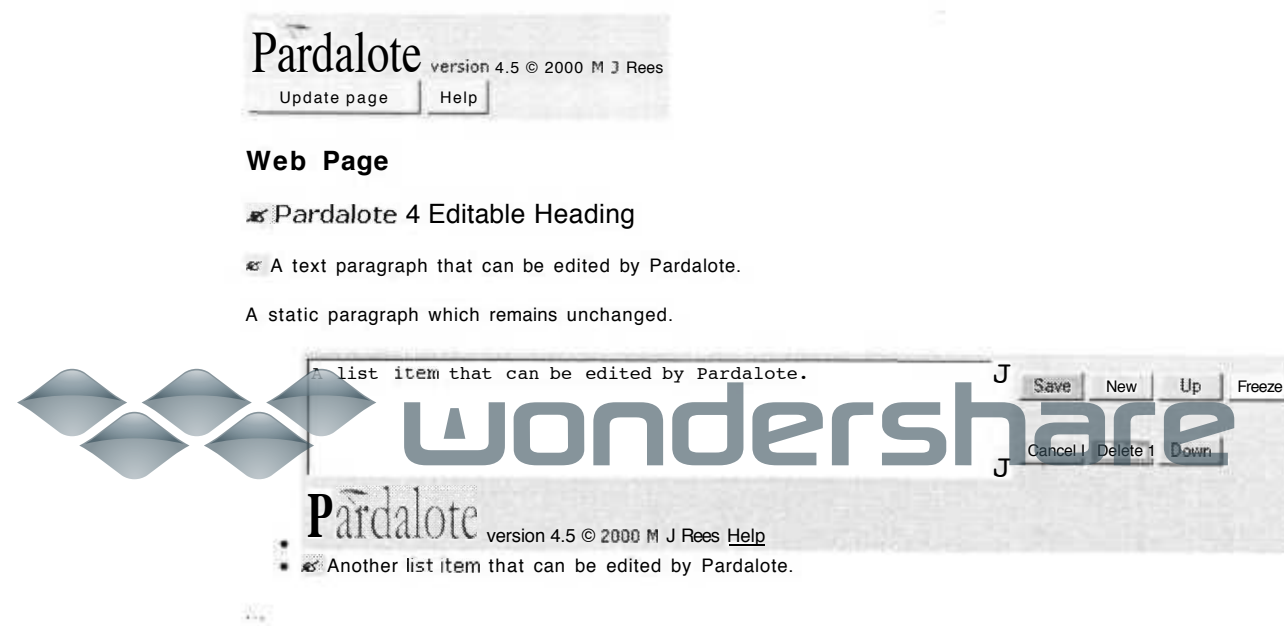


Figure 3. Pardalote 4 in-situ editing.

### ss Pardalote Planning Meeting

Place: xs Meeting Room 5

When: 23 December 2000 at 14:00

Chair: Prof Jim Yeates

Present: xxx

Apologies: yyy

#### 1. Use of XML

- Discussion
- Action

#### 2. Pardalote 5 Extensions

- Discussion
- Action

Pardalote version 4.5 © 2000 M J Rees Help

MeetingManager: Created with Pardalote lightweight editing (c) 2000 M J Rees

Figure 4. Pardalote meeting document management.



changes, the user can copy changed text to the clipboard, refresh the page to view the new version, redo the edits, then update again. Such a situation is extremely rare, and the solution fits well with the lightweight philosophy built into Pardalote.

#### 4. Document management with Pardalote

A simple meeting manager document constructed as an editable page is used to demonstrate the document management capabilities of Pardalote. This example allows a single web page to support the following phases of holding a meeting:

1. Announce meeting: title, date, time and place
2. Agenda setting: initial and refinements
3. Taking minutes during meeting, and formulating action items
4. Minutes update following meeting

This example makes use of a composite editing structure supported by Pardalote called the I-grain pod. A page author constructs a pod by creating a <DIV> tag holding one of more I-grains interleaved with static text, if required. A pod style class is then assigned to the <DIV> tag.

Figure 4 shows the upper part of a meeting document page soon after it has been created. Note that the I-grain pods are represented by the pod edit character. The user has clicked on the pod edit character of the second I-grain pod to display the pod edit panel shown near the bottom of Figure 4. Note the same edit functionality buttons are available allowing pods to be deleted, moved and created. However, no editing of I-grain pod content is possible in this view.

To edit individual I-grains within a pod, the user clicks on the normal I-grain edit character as before. This allows I-grains within a pod to be added, deleted and modified using the same mechanisms as already described.

One other Pardalote feature is shown in Figure 4. The numbers prepended to the agenda item titles are generated by the Pardalote script by inserting the <SPAN> tag:

```
<span class=pardNumber></span>
```

The SPAN will be replaced by the next number in sequence. Moving agenda item I-grain pods relative to one another will cause all to be renumbered automatically.

When the meeting document is first created, the users likely to attend are informed by email. Accessing the Pardalote page, they are able to edit existing agenda items and add new ones. Meeting organisers can amend the meeting title, location, place and chair at any time.

During the meeting, the minutes secretary can use a machine (perhaps a laptop with a wireless modem) to edit the minutes. Projecting the screen of this laptop allows the whole meeting to view the minutes as they are created.

Following the meeting, users can continue to refine the minutes. Eventually, the document can be frozen to prevent further changes, as described below. In this example notice that only a single meeting document is ever produced. There is thus only one version in one place that supports all phases of the meeting.

Shared documents will normally have a final version where no further editing is needed. The Pardalote editing features and special editing characters will no longer be needed. In effect, the editing is frozen.

Pardalote provides such a feature by adding an extra attribute to the <SCRIPT> tag inserted in the document head. Adding:

```
lweFreeze=""
```

will cause the Pardalote document control panel to be displayed as shown in Figure 5. When the user clicks on the Terminate editing button, all trace of Pardalote editing is switched off, and the page becomes regular static HTML.

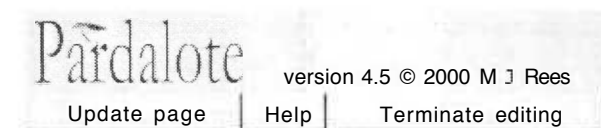


Figure 5. Button to freeze editing.

If the page author adds the attribute:

```
lweFreeze="../readonly/dual.htm"
```

then the document control panel is displayed as shown in Figure 6. In this case, when the user clicks on the hyperlink "../readonly/dual.htm" Pardalote will both update the page to allow continued editing, and save a second, frozen copy of the page in the indicated page file.

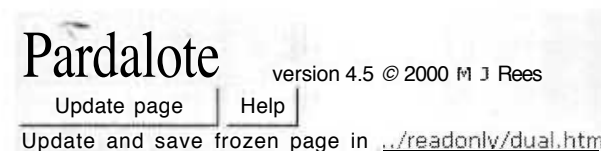


Figure 6. Update and save frozen page.

Such a facility is useful when the collaboration team wish to share a read-only copy of their latest document with a wider audience.

Only the Pardalote paragraph and pod I-grains have been described in this paper. Pardalote also supports section, hyperlink and in/out indicator I-grains. Examples of these additional I-grains can be found in [7].

Section I-grains are particularly useful for lists of various kinds such as project lists, comments, news items, meeting notices, and so on. Available with all I-grains, the Pardalote extend feature is especially helpful for web pages containing lists.

All list I-grains except the last in the list are given style classes that limit users to modifying the text

contents (in other words, no delete or move). The last I-grain in the list is given a style class with extend capability, such as lwePara-x. When the New button (see Figure 4) is clicked, the **current** I-grain loses extend capability, and the newly created, last I-grain retains it. This allows users to extend the list continually but all users are prevented from deleting or moving any list I-grains. This provides simple document management of pages containing shared lists.

#### 5. Conclusions and Future Work

Pardalote 4 provides a genuine lightweight implementation for in-situ editing within a web page. End-users have no need to download any special software, they simply use the Internet Explorer 5 browser. Page authors insert two lines of HTML into their pages, and then use their preferred web page editor to apply Cascading Style Sheet classes. Web site administrators copy one, exceedingly small web site (lwe), and set write permission on selected folders.

A comparison of Pardalote with other solutions mentioned in this paper is presented in Table 1.

Table 1. In-situ page editing solutions compared.

Solution	End-users	Page Authors	Web Site Administrators
Pardalote	LW	LW	LW
Sparrow	LW	MW	MW
EditLive	MW	MW	MW
OSE	LW	MW	MW
PageSeeder	LW	MW	MW

Key: LW = lightweight. MW = middleweight

Pardalote, of course, simply addresses in-situ editing, and does not provide any web site management facilities that are present in many of the competing solutions. Nonetheless, Pardalote is highly effective for single-web-page document management controlled by small collaborative teams.

Development and evolution of Pardalote is continuing. Pardalote 5 is being implemented, heavily based upon XML and XSLT. Each I-grain is represented as an XML data island. XSLT style sheets are applied dynamically when switching between the display form and editing form of an I-grain. With this approach it will be possible to save each I-grain individually as it is changed, thereby considerably lessening the clashing update problem.

At the same time, each I-grain represents a fragment of information to be shared, a fraglet. Since these fraglets are represented in XML, they may be identified and stored separately in an XML-oriented database. Once in this representation, fraglets can be searched, filtered and otherwise managed. In addition, they can be passed for processing to other, related applications for summarising, merging and reformatting into other types of information. Processing and manipulation of fraglets will form a large part in Pardalote 5 development, and will be presented in future papers.

#### References

- [1] Berners-Lee, T.. Weaving the Web. 1999, London: Orion Business Books.
- [2] MIT Workshop on WWW and Collaboration, <http://www.w3.org/CollaborationWorkshop/>.
- [3] IETF WEBDAV Working Group, <http://www.ics.uci.edu/pub/ietf/webdav/>.
- [4] Wiggins, J.W.a.M.. WEBDAV: IETF Standard for Collaborative Authoring on the Web. IEEE Internet Computing. 1998. September - October: p. 34-40.
- [5] Chang, B.-W.. In-Place Editing of Web Pages: Sparrow Community-Shared Documents. <http://www7.conf.au/programme/fullpapers/1929/com1929.htm>.
- [6] Rees, M. User Interface for Lightweight In-line Editing of Web Pages. in 1st Australasian User Interface Conference 2000. Canberra: IEEE. February: p. 88-94.
- [7] Rees, M. Implementing Responsive Lightweight In-page Editing. In AusWeb 2000. Cairns: Southern Cross University Press. June: p. 265-283.
- [8] EditLive, Ephox Pty Ltd, <http://www.ephox.com/>.
- [9] Weborganic Pty Ltd. PageSeeder, <http://www.weborganic.com/>.
- [10] Microsoft Office Server Extensions, <http://www.microsoft.com/Office/deployment/chap13.htm>