Collaborative Extensions for the UpLib System

William C. Janssen

9 June 2004

TR−04−3

parc℠
Palo Alto Research Center

Proceedings of the Fourth ACM/IEEE Joint Conference on Digital Libraries (JCDL), June, 2004, Tucson, AZ. Pages 239−240.

Portions of this paper are **Copyright 2004, Palo Alto Research Center**

This paper is part of the PARC Technical Report series.

For more information on PARC, please visit our Web site at http://www.parc.com/.
Our address is:

Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304 USA

You can contact us via telephone at 650−812−4000.
You can also send e−mail to info@parc.com; it will be forwarded appropriately.

Collaborative Extensions for the UpLib System

William C. Janssen
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California, USA
janssen@parc.com

ABSTRACT

The UpLib personal digital library system is specifically designed for secure use by a single individual. However, collaborative operation of multiple UpLib repositories is still possible. This paper describes two mechanisms that have been added to UpLib to facilitate community building around individual document collections.

Categories and Subject Descriptors

H.3.7 [**Information Storage and Retrieval**]: Digital Libraries—*systems issues, user issues*; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*collaborative computing*

General Terms

Design,Human Factors

Keywords

Personal digital library, Collaboration, Metadata sharing, Extension sharing

1. INTRODUCTION

The UpLib personal digital library system [6] provides a secure long-term storage and retrieval system for a wide variety of personal documents such as papers, photos, books, and email. The system creates a full-text indexed repository accessed through an active agent via a Web interface. It is suitable for collections comprising tens of thousands of documents, and provides for ease of document entry and access as well as high levels of security and privacy. It is highly extensible through user scripting, and is also intended to be useful as a platform for further research into digital libraries and computer-augmented reading.

However, the focus on individual libraries does not support some of the useful collaborative aspects of other systems designed to support multiple individuals simultaneously. We are exploring ways to restore this capability by using external server-based collaborative frameworks to create a community of UpLib users. Two of these collaborative mechanisms are described. One provides a way of sharing document metadata; the other provides a mechanism for sharing and locating customization extensions to the base UpLib system.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
JCDL'04, June 7–11, 2004, Tucson, Arizona, USA.
Copyright 2004 ACM 1-58113-832-6/04/0006 ...$5.00.

2. SHARING DOCUMENT METADATA

UpLib allows the addition of metadata values for each document, either automatically or with a Metadata Editor built into the user interface. Systems such as CDDB [1] have shown the power of shared metadata development by a community working on the same document corpus, and we wished to harness that power for communities of UpLib users whose repositories overlap to a significant degree. For example, two researchers in the field of digital libraries can be expected to have many of the same technical papers in their individual filing cabinets or on their individual hard disks. It would be useful to share metadata previously entered or corrected by others.

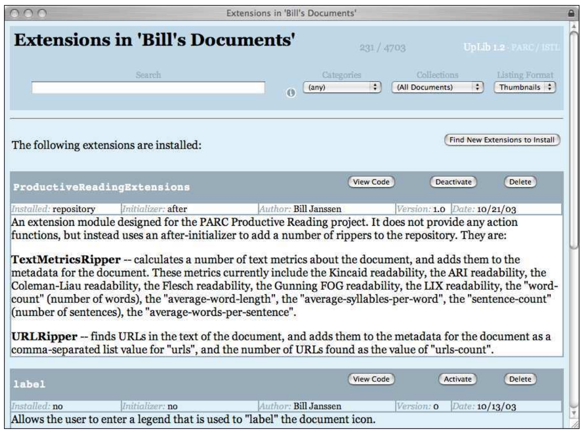
To identify two documents from two different repositories as being the same, a globally unique identifier for that document is necessary. In many cases the copies in two different repositories are bit-for-bit identical, as they are digital copies of the same original file; a digital hash of the document, using an algorithm such as SHA-1, can then be used as its unique ID. In other cases, document fingerprinting of the document text, as discussed in [5] and [4], will allow two slightly different versions of the same document to be identified as the same. Our system currently implements the hash identifier, which works well for photos, PDF files, and Powerpoint documents; we plan to add text fingerprints, as well.

We constructed a library service which takes a document identifier along with a set of metadata for that document and an optional *handle* identifying the user providing the information, and stores that information in a database. We then added the ability to contact this service to the repository code, using the built-in UpLib extension system. When the user presses the "Share Metadata" button in the Metadata Editor, a subset of the user's metadata is calculated, then sent along with the document's identifier and the user's handle; if no handle is set by the user, the information is sent anonymously. The particular metadata fields to send can be individually determined by settings in the user's configuration file, to avoid sending sensitive fields.

Similarly, a "Find Metadata" button calculates the document fingerprint, then sends it with a request to the metadata library server for any metadata known about this document. A list of known submissions for that document is downloaded to the repository daemon. The user can specify preferred submitter handles in a configuration file; if specified, they are checked to find the latest submission for that document from a preferred submitter. That metadata is then requested from the library, and associated with the document in the repository. The user can also specify whether found metadata overrides or augments existing local repository metadata for the document. The updated metadata for the document is finally displayed in the Metadata Editor for the user to check and, if necessary, correct.

3. SHARING UPLIB EXTENSIONS

Each repository is "guarded" by a daemon, which controls access to the documents in the repository and provides RPC services to various clients of the repository, such as the Web-based UpLib user interface. The daemon's Python code base and operation can be modified dynamically through an extension mechanism, which allows a client to to load (or re-load) dynamic library modules installed for that repository simply by requesting particular named services from the server. These modules can perform arbitrarily powerful modifications to the server, and can perform modifications to the standard configuration of the repository daemon at startup time; they are often used to add a new document analysis engine to the standard document addition pipeline. The standard Web interface to the repository daemon includes an Extensions Manager, shown in figure 1, which provides a view of all extensions loaded in that repository, and provides controls for the user to deactivate, activate, or view the code of, existing extensions.



Extensions in 'Bill's Documents'

The following extensions are installed:

ProductiveReadingExtensions

Installed: repository *Initializer:* after *Author:* Bill Janssen *Version:* 1.0 *Date:* 10/21/03

An extension module designed for the PARC Productive Reading project. It does not provide any action functions, but instead uses an after-initializer to add a number of rippers to the repository. They are:

TextMetricsRipper -- calculates a number of text metrics about the document, and adds them to the metadata for the document. These metrics currently include the Kincaid readability, the ARI readability, the Coleman-Liau readability, the Flesch readability, the Gunning FOG readability, the LIX readability, the "word-count" (number of words), the "average-word-length", the "average-syllables-per-word", the "sentence-count" (number of sentences), the "average-words-per-sentence".

URLRipper -- finds URLs in the text of the document, and adds them to the metadata for the document as a comma-separated list value for "urls", and the number of URLs found as the value of "urls-count".

label

Installed: no *Initializer:* no *Author:* Bill Janssen *Version:* 0 *Date:* 10/13/03

Allows the user to enter a legend that is used to "label" the document icon.

Figure 1: A display of some of the extensions loaded into an individual UpLib repository. Extensions are downloaded from a shared extensions server.

The number of users willing to construct significant extensions is low compared to the number who could benefit from those extensions. Studies such as [3] and [7] suggest the utility of providing a mechanism that allows the efforts of "translators" to be easily shared with the wider user population; systems such as PARC's CedarChest [8] and Mozilla Firefox's extension library [2] convince us of the benefits of a structured system for aggregating and distributing extensions.

To support this, we added a structured documentation system for extensions. This allows us to associate, with each extension, not only documentation on the functionality of the extension, but characteristics such as the version, author, or Web site (if any). In the future, we plan to add other characteristics incorporating user feedback, such as "grades". We then created a library service which accepts documented extensions and stores them in a database; it also allows the database to be searched, and extensions therein to be downloaded. These extensions are primarily Python code, but may also include code in other languages such Java jar files, Perl scripts, or even compiled executables.

Finally, we added two buttons to the Extensions Manager. The first, labelled "Contribute", appears on the title bar for each locally developed extension which is documented. Pushing it packages up the code files and documentation for the extension as a zip file, and uploads it to the extensions library. Pushing the second, labelled "Find New Extensions to Install", provides a filtered view of extensions available in the library. For each, one may read the documentation, browse the code, visit its Web site, or download the extension to the repository. Once an extension has been installed in a repository, the Extensions Manager allows the user to activate it, deactivate it, or delete it.

4. CONCLUSION

We have presented the addition of metadata sharing and extension sharing to individual personal libraries, to form a shared community, by creating implementation-language independent network services that can be accessed from the repositories. We are continuing to explore this idea, using the UpLib system, as part of the current research at PARC into computer-augmented reading.

5. REFERENCES

[1] http://www.gracenote.com/.

[2] http://texturizer.net/firefox/extensions/.

[3] T. J. Allen. Communication networks in R&D laboratories. *R&D Management*, 1(1):14–21, October 1970.

[4] A. Z. Broder. Identifying and filtering near-duplicate documents. In *Proceedings of the Conference on Combinatorial Pattern Matching*, volume 1848 of *Lecture Notes in Computer Science*, pages 1–10. Springer-Verlag, 2000.

[5] N. Heintze. Scalable document fingerprinting. In *Proceedings of the 1996 USENIX Workshop on Electronic Commerce*, pages 191–200, November 1996.

[6] W. C. Janssen and K. Popat. UpLib: A universal personal digital library system. In *Proceedings of the 2003 ACM symposium on Document Engineering*, pages 234–242. ACM Press, November 2003.

[7] W. E. Mackay. Patterns of sharing customizable software. In *Proceedings of the 1990 ACM Conference on Computer-supported Cooperative Work*, pages 209–221. ACM Press, 1990.

[8] D. C. Swinehart, P. T. Zellweger, R. J. Beach, and R. B. Hagmann. A structural view of the cedar programming environment. *ACM Transactions on Programming Languages and Systems*, 8(4):419–490, 1986.