

Synchronous Editing via Web Services: Combining Heterogeneous Client and Server Technologies

Jon A Preston and Sushil K Prasad

Department of Computer Science

Georgia State University

Atlanta, GA USA

jon.preston@acm.org and sprasad@gsu.edu

ABSTRACT

The primary goals in a synchronous collaborative editing system (CES) involve ensuring a high level of concurrent access (editability) to users while maintaining the properties of the CCI model (consistency, causality-preservation, and intention-preservation). We present a system that enables CCI-compliant synchronous editing of a shared document by utilizing dynamic locking – scaling the space each user “owns” to accommodate concurrency-controlled editing by multiple users. We utilize caching so that many users may edit the document synchronously without having to incur the communication cost associated with optimistic concurrency control techniques (ex. merging and operational transformation). Additionally, our CES uses Web Services to transparently connect clients to legacy document repositories (such as CVS, VSS, and RCS), enabling synchronous editing capabilities to these systems where such capabilities are not natively supported. Because we employ Web services, users may use a heterogeneous collection of client editing software and a heterogeneous collection of server repository software seamlessly together.

Categories and Subject Descriptors

H.5.3. [Information Interfaces and Presentation]: Computer-supported cooperative work – *synchronous interaction, collaborative computing, Web-based interaction.*

General Terms

Algorithms, Performance, Experimentation, Human Factors, Theory.

Keywords

Collaborative Editing System (CES), Web Services, Synchronous Editing, Configuration Management System (CMS), Software Engineering.

1. INTRODUCTION

[7] establishes the well-accepted CCI standard by which

collaborative editing systems may measure success in managing concurrent access to shared documents. Recent CES employ optimistic concurrency control techniques to ensure CCI and must consequently deal with the problem of combining disparate copies of the shared document through operational transformation (OT) or other merge techniques.

Motivated by configuration management systems such as Revision Control System (RCS) that employ pessimistic concurrency techniques to avoid the problems of merging changes to shared documents, we revisit the idea that pessimistic locks offer potential CES research opportunities. We recognize that using pessimistic locks reduces the concurrent access, thus the CES community has lately focused on optimistic concurrency control approaches; but what if locks were dynamic and could grow and shrink automatically? Similar to techniques adopted by [2], we dynamically manage lock granularity to allow maximum concurrent access among many authors and avoid the need to merge disparate versions of the shared document, which can not be resolved occasionally. The locks are managed automatically and transparently to the user, improving usability of CES as defined by [1], so there is no a priori knowledge required of what section(s) the users desire. A significant reduction in communication cost is also achieved using our dynamic locking (DL) scheme as compared to the communication cost of multicasting all changes to all users within the CES (as done in OT-based systems) [5].

2. ARCHITECTURE

Having established the effectiveness of our theoretical algorithms [4][5], we incorporate these lock management and caching algorithms into an architecture for enabling synchronous collaborative editing via heterogeneous client editors and heterogeneous server repositories [6]. This architecture extends the work of [2] (which connects heterogeneous client technologies within a CES) by adding server heterogeneity and notification mechanisms. Our architecture supports a heterogeneous mixture of client and server technologies connected via Web Services as shown in Figure 1; as a result, existing client and server technologies are extended to include synchronous collaboration capabilities.

The **Client Application Listener** component listens to change events that occur within the application (edits to the document) and cache changes when possible, sending these changes to the server coordinating the CES. This component also receives update notifications from the server and sends the changes to the existing client application (editing software), thus maintaining consistency among all users' copies of the shared document.

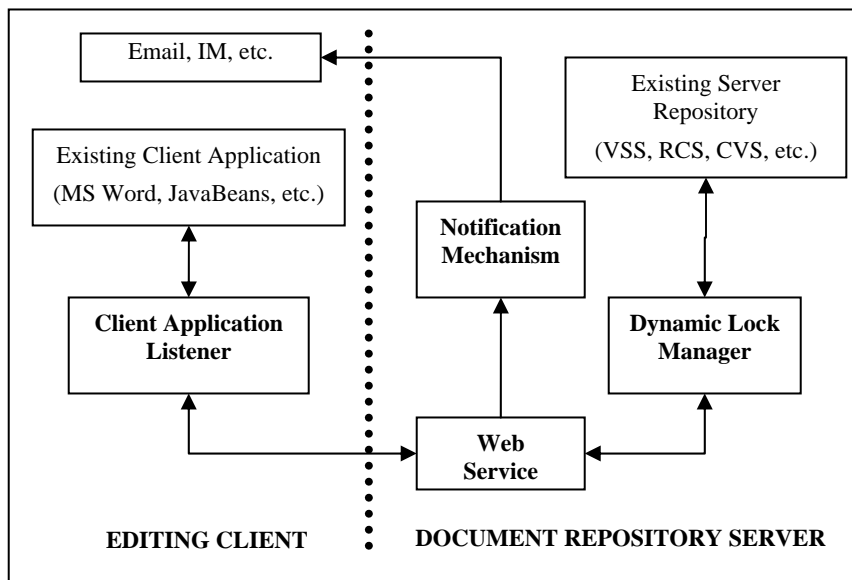


Figure 1. Open-system architecture for distributed repositories

The **Web Service** component provides an API for traditional CMS systems (check-in and check-out, etc.) as well as an API for managing changes among the users that are collaborating together (insert, delete, move, etc.). This component also provides an API by which users can subscribe to receive synchronous and asynchronous notification when a document has been changed.

The **Dynamic Lock Manager** component acts as a proxy that checks-out and checks-in documents from the existing server repository (such as RCS, CVS, VSS, etc.). This component receives check-in and check-out events from the Web Service component and processes these requests via the existing server repository. This component provides the ability to manage artifacts at a finer granularity (viewing an artifact as a collection of sub-artifacts); as an example, a user can edit page one of a shared artifact at the same time another user is editing page two. This component tracks who is currently working on each artifact in the server repository and is thus able to “push” these changes to the necessary clients.

The **Notification Mechanism** component is responsible for passing on any events for which the user has requested notification (document change, check-out, etc.) to the users’ preferred email, IM, cell phone text messaging, etc. Clients may subscribe for notification when changes are made (even if they are not currently editing the document); thus the system supports synchronous and asynchronous collaboration.

3. DEMONSTRATION SYSTEM

This demonstration will show a functioning implementation of our algorithms and architecture as previously described in this paper as well as in [4], [5], and [6]. Specifically, we exhibit two client application listeners: one that connects to Microsoft Word and one that connects to a text editor. We also show two dynamic lock managers: one that connects to Microsoft Visual Source Safe (VSS), and one that connects to Revision Control System (RCS). The Web Services API remains consistent among both server technology implementations, thus any client technology may connect to and utilize either of the server repositories with the

extended concurrency control and dynamic locking capabilities we have developed.

Additionally, we demonstrate the notification mechanism for synchronous and asynchronous collaboration. Notifications may be synchronous (real-time) for users currently active in the CES as well as asynchronous for users that have requested to be notified asynchronously; for example, if a manager subscribed to be notified a project specification document was modified, our notification mechanism sends a message to the user when this change occurs. The notification mechanisms demonstrated include email and telephony (text messaging).

Because our system’s strength lies in maximizing concurrent access while minimizing communication cost, this demonstration also provides visual representations of the document tree used to dynamically manage locks (client ownership information) and displays network cost.

4. REFERENCES

- [1] Greenberg, S. and Marwood, D., “Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface”, *ACM Conferences on Computer Supported Cooperative Work*, ACM Press, Nov. 1994, pp. 207-217.
- [2] Li, D. and Li, R. “Transparent Sharing and Interoperation of Heterogeneous Single-User Applications”, *Proceedings of CSCW’02*, New Orleans LA, November 2002, pp. 246-255.
- [3] Pacull, F., Sandoz, A., and Schiper, A. “Duplex: A distributed collaborative editing environment in large scale”, *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW ’94)*, 1994, pp. 165-173.
- [4] Preston, J. A. and Prasad, S. K., “A Deadlock-Free Multi-Granular, Hierarchical Locking Scheme for Real-time Collaborative Editing”, *7th International Workshop on Collaborative Editing Systems*. Sanibel Island, FL, 2005.
- [5] Preston, J. A. and Prasad, S. K., “Dynamic Locking of Varying Granularity in Generalized Document Trees to Maximize Concurrency and Minimize Communication in Synchronous CES”, *The 2nd International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, Atlanta, GA, 2006, under review.
- [6] Preston, J. A. and Prasad, S. K., “A Web-Services-based Open-System Architecture for Collaborative Editing Systems”, *Fourth International Conference on Cooperative Internet Computing*, Hong Kong, China, Oct. 2006.
- [7] Sun, C., Jia, X., Zhang, Y., Yang, Y., and Chen, D., “Achieving convergence, causality-preservation, and intention-preservation in real-time cooperative editing systems”, *ACM Transactions on Computer-Human Interaction*, 5(1), Mar. 1, pp. 63-108.