Multi-Grain Version Control
in the Historian System

Makram Abu-Shakra and Gene L. Fisher

Department of Computer Science
California Polytechnic State University
San Luis Obispo, CA 93407
{mabushak, gfisher}@csc.calpoly.edu
http://www.csc.calpoly.edu

Abstract. This paper describes Historian, a version control system that supports comprehensive versioning and features to aid history navigation. Comprehensive versioning is supported through frequent and automated creation of versions which typically results in a large number of versions. To reduce user overhead in history navigation, the hierarchical structure present in most documents is utilized to support fine-grained version control. The series of document editing operations is also organized hierarchically and can be used for navigation as well.

1 Introduction

This paper describes the Historian editor-based version control system. Historian supports fine-grain version control of documents, with an end-user interface that provides efficient and convenient version retrieval. A document is a collection of structured text, where the structuring is defined on a document-specific basis. For example, a prose document is structured as paragraphs within sections within chapters; a program is structured as declarations within functions within files.

A number of existing systems provide features similar to those available in Historian, notably COOP/Orm [10] and Orwell [19] The contributions of Historian compared to these systems are:

a. support for comprehensive version control at multiple levels of granularity along both structural and temporal lines
b. a graphical user interface for convenient historical navigation and low-overhead version retrieval

Structural granularity is based on the natural hierarchical organization of a document. For example, versioned items in an object-oriented program range from functions, to classes, to modules, as described well in [7] Temporal granularity is based on the frequency of version creation during document editing. For example, the frequency of version creation can range from saving at each individual edit operation, at the document save operation, or at each document check-in operation.

© Springer-Verlag Berlin Heidelberg 1998
The support of different levels of granularity for both types provide unique advantages to the software configuration process. While our approach focuses on the effects of granularity on viewing the evolution of documents, existing systems have explored its effects on other areas such as merging alternatives and group awareness.

The focus of Historian is notably more narrow than many full-feature configuration management systems. From the end user's perspective, Historian appears as a largely standard text editor, with specialized features for version navigation and retrieval. In this sense, the end-user look and feel of Historian are much like groupware editors such as PREP [13,14] and SEPIA [5]. Historian can be aptly viewed as a specialized component that can be integrated into a larger CM environment.

The term fine-grain version control appears first to have been used by the authors of Orwell [19] to refer to the version control of individual program methods (i.e., functions) as opposed to files or modules in an object-oriented program. This is fine-grain on the structural level. In COOP/Orm [11] the term is extended to mean frequent creation of versions as well. This is fine-grain on the temporal level. While a number of other good systems have addressed fine-grain version control on one or both of these levels, none has focused on convenient and low-overhead retrieval, which we view as an important issue. Historian provides version control that is comprehensive, fully automatic, efficient, and supports focused retrieval.

The motivation for Historian has come from our own frustration as authors when changes to a document eliminate versions that we might later wish to retrieve. Such frustration arises when working alone, and is amplified when working collaboratively with other authors, as reported in [3] for example. While version control systems have always addressed this problem to some extent, systems typically keep track of changes only when files are checked in, or at other explicitly invoked user operations. Comprehensive and fully automatic version control introduces the security that any version of a document is retrievable, without the user having to request version saving explicitly.

There are two major problems with comprehensive version control: excessive use of storage space and the high overhead of navigating through a large number of versions. Solutions to storage problems such as file compression and differencing algorithms are well known and are utilized in Historian to keep memory usage at a minimum. User interface solutions to the latter problem are largely unaddressed in CM systems, and the primary focus of Historian. In order to keep comprehensive version navigation practical, features are introduced to aid the user in searching for desired versions. In our experience, version navigation in most CM systems is already awkward and time-consuming; saving a comprehensive set of versions makes matters even worse.

We began our research with the assumption that reducing user overhead in navigation will make version retrieval a more common practice in software development and will thus increase productivity. In our experience using RCS [20] or RCS-based systems, we have found that saved versions were very rarely