

## OBJECT-ORIENTED DOCUMENT VERSION TRACKING METHOD AND APPARATUS

### COPENING APPLICATIONS

This application is the one of four sibling patent applications filed on an even date herewith and commonly assigned, including U.S. patent application Ser. No. 08/638,908, entitled "METHOD AND APPARATUS FOR ORGANIZING A DOCUMENT USING HIERARCHICAL ARRANGEMENT OF OBJECTS", and U.S. patent application Ser. No. 08/638,992, entitled "METHOD AND APPARATUS FOR CONSOLIDATING EDITS MADE BY MULTIPLE EDITORS WORKING ON MULTIPLE DOCUMENT COPIES" and U.S. patent application Ser. No. 08/637,310, entitled "METHOD AND APPARATUS FOR DISPLAYING MODELESS BAR INTERFACES IN A COMPUTER SYSTEM". The subject matter of the above-identified copending patent applications is hereby incorporated by reference.

### FIELD OF THE INVENTION

This invention relates to document editing and to editing systems which store document revisions, and, in particular, to a method and apparatus for minimizing storage space necessary to store multiple revision copies of a document.

### BACKGROUND OF THE INVENTION

Word processing software has revolutionized the manner in which people generate documents and, in some cases has created whole industries such as the desktop publishing industry. However, when documents are created by more than one author, or are very large and complex, they often are revised many times before a final document is created. During the course of the document creation, when a new round of changes is being made, it is frequently desirable to maintain a complete copy or version of the document before the new revisions are made. The version copy may then be used as a comparison document with the current document to quickly locate the changes or may be used to reverse or "undo" the changes. Depending on the complexity of the document or the number of people involved in its creation, there may be multiple copies or versions of the document which must be stored.

Current word processing applications often lack the functionality necessary to efficiently organize and manage multiple document versions. In particular, it is increasingly desirable to include all version copies in a single master document, so that the copies can be stored, retrieved and distributed as a single entity. Although some word processing applications have provided facilities for maintaining multiple version copies, they generally do not allow a single file to include more than one version. Therefore, separate files need to be created for each version copy. The use of separate files has several disadvantages. First, a large amount of memory is consumed since the entire document must be stored and retrieved for each version. Secondly, the large amount of duplicate information, which must be manipulated due to the multiple versions, requires more processor time and therefore, slows the entire word processing system.

In order to overcome this latter difficulty, some systems only store copies for a predetermined number of the latest document versions or only load into memory the latest versions. However, with these latter systems, it is frequently not possible to keep a record of document changes since not

all versions of the document are simultaneously available. In addition, in such system, since all changes are not available, some changes cannot be undone and redone.

Further, since separate copies of the versions must be maintained, there is an increased possibility of separating the copies and either losing some copies or failing to transmit the copies to editors or reviewers.

Accordingly, there is a need for a versioning tool which allows for the creation of a single master document containing all versions without duplicating the entire document for each version. More particularly, there is a need for word processing software which enables authors to create and revise documents and maintain copies of all previous revisions so that all changes can be undone and redone. There is a further need for a versioning tool which is compatible with an object-oriented word processing program.

### SUMMARY OF THE INVENTION

The foregoing objects are achieved and the foregoing problems are overcome by one illustrative embodiment of the invention in which a single file holds multiple versions of a document. The document is composed of an interconnection of objects which themselves have versions and are stored in the file. When the document is changed by changing any of the interconnected objects, a check is first made to determine whether the object version is same as the document version currently being edited. If not, a copy of the object is made and saved. Any version of the document can be reconstructed by interconnecting object versions which have a highest level which is equal to, or less than, the desired document version. Therefore, only objects which are changed are duplicated and copies of objects are only made when an object changes.

In accordance with a preferred embodiment, the interconnection of the objects to form structure of the document is maintained by an interconnection of object pointers and demand loader objects. Each demand loader object includes a list of object versions for the associated object plus a pointer to one of the versions. The demand loader object also includes methods for loading the associated object into memory from storage when the object is accessed. In this manner, the whole document does not have to be loaded in order to be accessible.

### BRIEF DESCRIPTION OF THE DRAWING

The above and other features, objects and advantages of the invention will be better understood by referring to the following detailed description in conjunction with the accompanying drawing in which:

FIG. 1 is a block diagram of a computer system suitable for use with the present invention;

FIG. 2 is a schematic block diagram of the program elements comprising the inventive versioning tool;

FIG. 3 illustrates schematically the structure of a base document object used in the inventive system;

FIG. 4 illustrates in a block schematic manner the use of object pointers to reference an object;

FIG. 5 illustrates in a block schematic manner the construction of a demand loader object;

FIG. 6 is a schematic illustration of multiple versions of documents which share objects;

FIG. 7 is a block schematic diagram of the internal construction of a demand loader object which illustrates a version pointer list.