8-16-1996

Configuration Management in Collaborative Writing

Hilda Tellioglu
Vienna University of Technology, tellioglu@tuwien.ac.at

Configuration Management in Collaborative Writing

Hilda Tellioglu
Vienna University of Technology
Institute for Design and Assessment of Technology, Department for CSCW
Argentinierstrasse 8/187/1, A-1040 Vienna, Austria
e-mail: Hilda.Tellioglu@tuwien.ac.at

1. Introduction

Configuration management (CM) is a cooperative task, and needs both technical and organizational support. It is one of the most important disciplines enabling efficient and consistent use of shared information and simultaneous changes to same information items, like parts or sections of a document, or software modules. CM has been mainly developed to support collaboration in software production teams. There are different kinds of CM methods and tools which are described in the first part of the paper. The role of CM in collaborative writing is explained in the second part. The third part includes the description of two typical situations during collaborative writing to illustrate work practices of authors, when they use CM tools.

1. Configuration Management in Software Production

CM is originally developed as a discipline to support software production in teams. It is a methodology to manage the development of computer systems. In this context it is defined as "the discipline of identifying the configuration of a system at discrete points in time for purposes of systematically controlling changes to this configuration and maintaining the integrity and traceability of this configuration throughout the system life cycle" [1, p.20]. With system they mean a regularly interacting group of three items (software, hardware that hosts the software and other hardware) utilizing computers and software by forming a unified whole (p.7). To determine the system's organization and construction, all system modules must be identified. Each module, interdependencies between modules, how modules fit together and what changes made to modules are, have to be described thoroughly. Ad hoc data access is enabled to produce individual reports which can include system information. For medium- and large-scale software projects which can be categorized as 'multiple-person, network-based projects', CM is a practical necessity [6].

There are significant differences among various CM tools [2]; the tool can be file- or object-oriented, it can represent changes physically or logically, and it can apply changes sequentially or selectively. There are two main groups of CM models:

- The *sequential model* supports the building of tools for archiving and retrieval of previously named versions, and not for really managing the software systems. The version is the fundamental unit of specifications. Differences between an old version and its successor represent changes. Files are used to save changes physically, and therefore the system is limited to the kinds of information available in files (e.g. names and file contents). The system can retrieve only versions that are read into it. The basic type - the '*linear sequential model*' - does not provide a good method of dealing with parallel versions, whereas the '*branched sequential model*' allows developers to work with parallel versions through branching and merging. A tree of versions is created which can be retrieved by the system any time. DEC's Code Management System (CMS), Intersolve's PVCS and Mortice Kern Systems (MKS) are some examples based on the sequential model.
- CM tools based on the *selectable-change model* are more than simple version-retrieval systems. They are object-oriented, and therefore a change saved logically as an object can be associated with whatever information is meaningful to the developer. New combinations of prior changes can be selected to produce new versions; an old version of a file (e.g. a source code or an object file)

can be compared to an altered version and applied independently. This is made possible by characterizing each version implicitly or explicitly by a list of changes to be administered. The change builds the fundamental unit of specifications. Any combination of changes can be used to specify a version. There are some CM tools based on selectable-change model, such as Aide-de-Camp, IBM Update, CDC Update and Historian Plus.

A CM system contains a set of tools fulfilling different functions. A *version control system* controls changes to source code. An *object module librarian* manages object codes in object libraries. A *make utility* enables automatic system building. There are also other tools to support testing, maintaining and other activities in software production which are irrelevant to cooperative writing, and therefore not described in this paper.

1. Configuration Management in Collaborative Writing

There are tools for cooperative writing discussed within the CSCW (*computer supported cooperative work*) community [3]. Collaborative authoring systems aim to support both synchronous and asynchronous communication between authors writing documents together [4]. The five stages - coordination, writing and annotation, consolidation, negotiation and coordination - build up this distributed writing process. The use of CM systems can easily solve the requirements of the real-time cooperative editing. Problems and restrictions, like overwrites of text, prohibition of simultaneous work on same documents, impossibility to reuse the existing documents or document parts because of the lack of overview or of not thoroughly organized document structure(s), etc., can be avoided, if CM systems are used accurately by all participants.

The reasons for choosing CM for collaborative writing are of various kinds. First, documents' change control is required in all phases of cooperative writing. The integrated version control system enables controlling and monitoring of changes made to the whole document and to its parts. It documents the history of changes and allows to revert to a prior version of a part, if necessary. It provides the ability to determine the differences between two different versions. It also keeps track of who made the changes, what the change is involved in, and when the change was made. Second, the complexity of controlling the document's organization and structure can be reduced through an appropriate CM system. Reconstruction of multiple versions is supported by the version control system. An integrated reporter increases the transparency of document parts and their dependencies with each other. It enables monitoring of authors' access by tracking the progress and change within a document. By considering the track dependencies previous document versions can be reproduced easily. A tracker notifies authors of document changes and creates updates of change status. Third, without CM it is often the case that only one person knows the information about which document parts are needed to build the entire document, how they depend on each other and how they can be reused. If this person is not available, much of this valuable information can be lost. Finally, CM is needed to prevent making conflicting changes to same documents by multiple authors. The next section describes this issue in detail.

1. Use of Configuration Management Systems

For collaborative writing a shared work environment is necessary. Through a network authors must have access to all common resources, i.e. various types of documents and document parts. The intended document with all its parts is located on a central computer unit (server). Documents and their parts are structured in directories based on their positions in the whole document, on their relationships to other documents and on their current versions. The following figure shows an example of a document structure. Document 1 consists of 3 parts, document 2 of 2, etc. All documents build together the entire document.

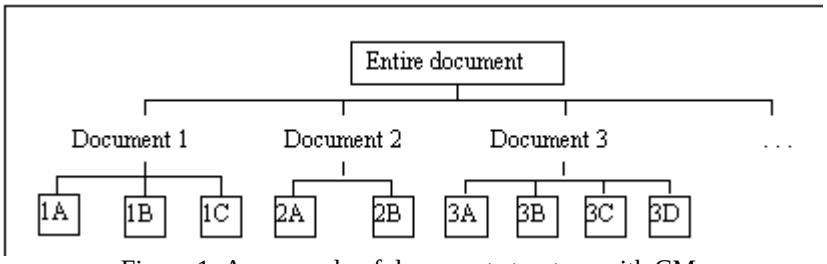

Figure 1. An example of document structure with CM.

By using branch and merge functions of CM systems multiple authors can modify same documents at the same time. The simultaneous changes cannot be lost or overwritten [6]. There are two situations which occur usually while working with CM tools. One is the *access of multiple authors to same document parts at the same time*. Author 1 sends an access request to the server. If the requested item is idle, i.e. not locked by another author, the CM system allows the access to this item. In case of a write access, the item is locked by author 1 by marking it with appropriate data like author's identification, date and type of access, etc. If an author 2 wants to make modifications to the same document item, s/he tries to get this item by sending a request to the server as well. But the CM system replies to this request with a denial by attaching the information about author 1. In this case author 2 contacts author 1 to create a branch of the locked item. Author 1 unlocks the item after putting it back onto the server. Now, a branch of this item can be created with the corresponding branch numbers. Both authors get their branched items and lock them. After finishing all modifications authors can merge their items, and the created new item gets a new version number. Table 1 shows three examples of mentioned tasks.

Tasks in collaborative writing	*Version of whole document*	*Document before modifications*		*Author 1*	*Author 2*	*Document after modifications*		*Version of whole document*
Creation of different document parts (A and B) by different authors	0	A/0	B/0	A/1.0	B/1.0	A/1.0	B/1.0	1
Working on the same document part (A) with branch/merge functions	1	A/1.0	B/1.0	A/1.1	A/1.2	A/2.0	B/1.0	2
Working on the same document part (B) with branch/merge functions	2	A/2.0	B/1.0	B/1.1	B/1.2	A/2.0	B/2.0	3

Table 1. Examples of tasks during collaborative writing. Notation is 'identification of document part/its version'.

The other typical situation is the *creation of a new document version*. After finishing all changes of document parts, one author who is responsible to create new document versions sends a request to all other authors that they put their documents back onto the server and unlock them. S/he then locks all document parts and creates a new document version. Afterwards all documents can be unlocked again to further changes and improvements. All inconsistencies are visible during this process, such as if one author forgets to put one of the document items onto the server, or if an old version of a document item is put instead of the latest one, etc.

CM helps authors to cooperate with each other. Distribution of responsibilities and the control of writing activities are supported by various CM systems. The mentioned two examples show how easy writing activities among multiple authors can be coordinated, if an appropriate CM system is used.

1. Conclusion

CM is a very important process during collaborative writing. Especially if there are more than one author, it goes without saying that the management of documents is easier, if their work environment is a network with supporting communication and coordination tools, like e-mail and scheduler. Applications supporting the document configuration, like CM systems, are essential to keep track of all steps in the collaborating group, and to maintain the document versions and dependencies between all document parts. Authors can work simultaneously without overwriting the changes of others in the group. Merging two documents becomes an easy process, if a CM tool is used for this. It is also necessary that authors understand the importance of such systems and use them regularly to insure the consistency of their documents. The existing CM tools need to be improved to avoid additional effort to use them for collaborative writing work. The acceptance of authors must be increased by integration CM tools into their work environments. Through further development CM must be used as the central part of new collaborative writing applications.

1. References

1. Bersoff, E. H., Henderson V. D. and S. G. Siegel, *Software Configuration Management. An Investment in Product Integrity*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1980.
2. Cronk R. D., "Tributaries and Deltas. Tracking software change in multiplatform environments", *Byte*, pp. 177-186, 1992.
3. Kirby, A. and T. Rodden, "Contact: Support for Distributed Cooperative Writing", *Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work*, Stockholm, Sweden, pp. 101-116, 1995. (More information about tools is available by the author.)
4. McAlpine, K. and P. Golder, "A New Architecture for a Collaborative Authoring System. Collaborwriter", *Computer Supported Cooperative Work 2*, Kluwer Academic Publishers, Netherlands, 1994.
5. Intersolv, *PVCS. User's Reference Manual*, Beaverton, OR, 1991.
6. Tellioglu, H., "Software Technology in a Networked Computer Environment: Configuration Management in Software Design and Development Teams", *Proceedings of the Conference Network Entities (NETIES)*, Athens, 1995.
7. Tellioglu, H. and I. Wagner, "Negotiating Boundaries: Strategies and tools for configuration management in software development teams", submitted to the Special Issue of Journal of CSCW on 'Studies of Cooperative Design'.
8. Wagner, I. and H. Tellioglu, "Software Cultures. Cultural practices in the creation of identity spaces in systems design", in preparation.