

Integrating Version Control and Module Management using Three-Dimensional Visualization

Hideki Koike and Hui-chu Chu¹

Graduate School of Information Systems, University of Electro-Communications,
1-5-1, Chofugaoka, Chofu, Tokyo 182, Japan

Abstract: This paper describes a three-dimensional visualization system which integrates version control and module management using 3D space. Users can operate the RCS commands using GUIs. Moreover, when users need to rebuild an entire software, the system automatically retrieves all the necessary files, compiles them, and links them.

1 INTRODUCTION

In practical software development and maintenance processes, version control and module management are very important. In UNIX, RCS (Revision Control System) or SCCS (Source Code Control System) is used as a version control tool. Although these tools have enough functionality for version control, they have not been put to good use of both by novices and by experts.

There are couple of reasons why they have not been used by novices.

- *Invisible*
Each version history is stored in text in a RCS/SCCS history file. It is, therefore, hard for novices to understand intuitively the version histories.
- *Typing required*
Users have to type complicated RCS/SCCS commands to check in/out a certain version they need.

These problems will be mostly solved by visualizing each version history as a 2-D tree, and by preparing GUI (e.g. menus or buttons) corresponding to the RCS/SCCS commands.

From an expert's point of view, the RCS/SCCS is not satisfying to be used in practical software development process because of the following reason.

- *Management of multiple files*
A software system is generally composed with multiple files. Although RCS/SCCS can keep version histories of each file, they do not know which combination of these versions rebuilds an entire software system.

¹Email: {koike,chuchu}@vogue.is.uec.ac.jp

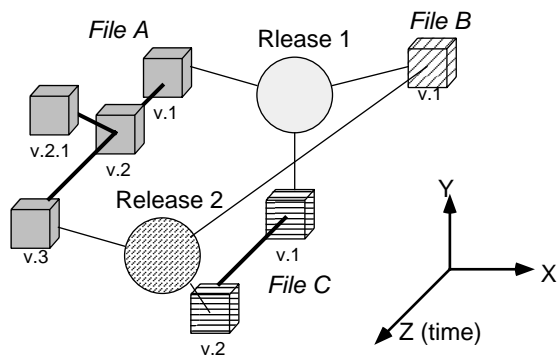


Figure 1: A 3D framework for visualizing version/module information.

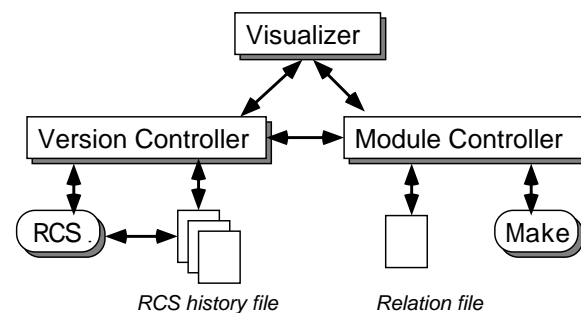


Figure 2: Architecture of VRCS.

2 3D VISUALIZATION AS A SOLUTION

We proposed a 3D visualization framework as a solution[3]. In our framework, each version history is represented as a 2D tree by taking Z axis as time. Those 2D trees are laid out in 3D space so that the files in the same module can be placed physically near each other in the XY-plane. Moreover, each major release of the entire software system is represented as a cube in center of 2D trees. Then, the files which should be compiled/linked together are connected with a link called relational link.

3 IMPLEMENTATION

Based on the framework described above, the system is developed. The system is implemented using C++ on SGI workstation and our 3D software visualization tool VOGUE. Figure 3 is a snapshot of the screen.

When the user invokes the system with a file name as an argument, the system parses its RCS history file (which is located in the RCS directory and indicated by the extension “,v”) and visualizes the version history as a 2D tree. When the user want to check out a certain version of the file, he/she just need to select a corresponding node by mouse and to choose “Check Out” from the menu. A target version of the file is retrieved by RCS and is displayed on a editor window.

When the user want to check in a file, he/she just need to select any node in the tree and choose “Check In” from the menu. After the file is checked in the RCS history file, a new node appears on the screen.

In the same way, if the user wants to compare two versions, he/she selects two nodes and chooses “Diff” from the menu. The difference is displayed on a editor window.

When the user invokes the system with a directory name, the system first reads a relation file under the RCS directory. Then it parses each RCS history file, visualizes them as individual 2D trees, make release nodes on Z-axis, make links between each release node and versions which is necessary to build the release.

When the user need to retrieve all files which compose the release, he need to select a corresponding release node and choose “Check Out” from the menu. The RCS checks

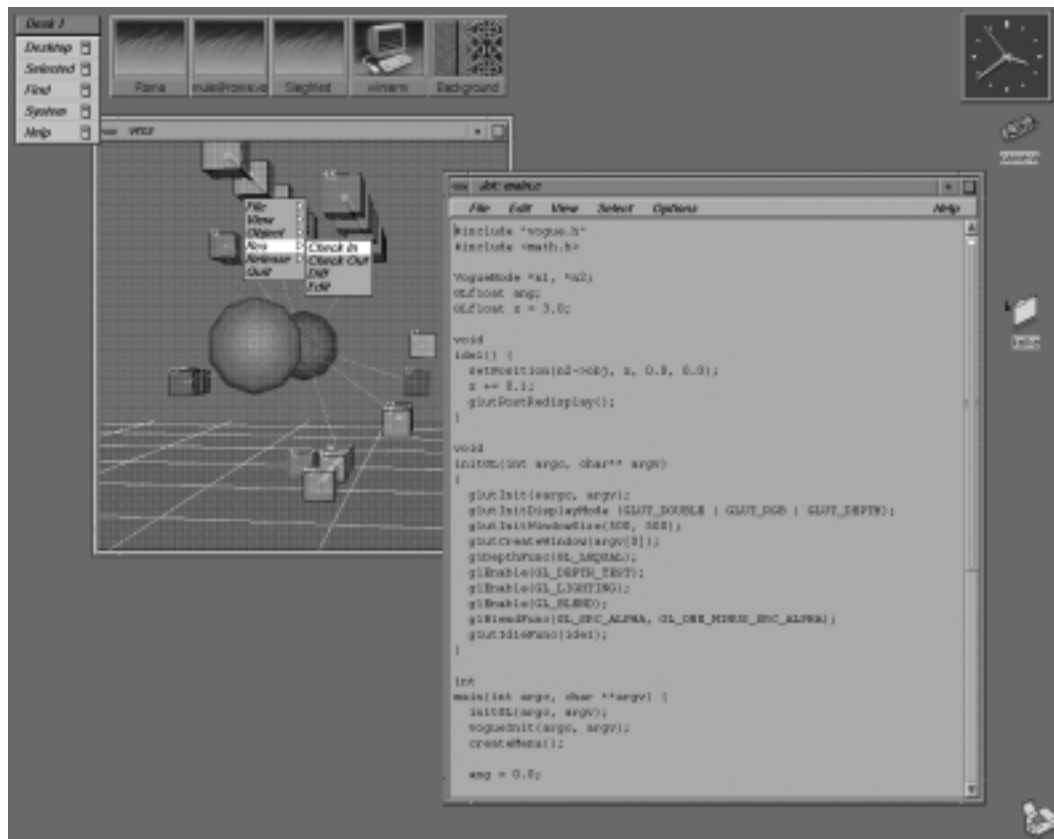


Figure 3: Visualizing version histories with VRCS. The user can execute RCS operation by menu. When the user selects a node and chooses “Check Out” from the menu, an editor window appears with the corresponding version of the file. Moreover, the versions which compose a certain release are connected by relation link via release node (indicated as sphere).

out all versions connected by relation links.

If he/she need to get an executable file for the release, he/she will choose “Make” from the menu. The system retrieves all files which are necessary to rebuild the executable file, compiles them, and links them. The users do not need to remember which versions are necessary. Just by choosing a release node and by selecting “Make”, the target executable file is rebuild automatically.

4 RELATED WORK

The GNU Make behaves as module management tool just like traditional UNIX Make. It is also capable of version control in cooperation with the traditional version control tool such as SCCS. Because of its upper-compatibility with the traditional Make, the GNU Make is now getting popular in UNIX world. AFS [4] is another example which tries to unify version control and module management. However, since the AFS requires

special directory set-ups and is incompatible with the traditional tools, it is not widely accepted by UNIX users. Although GNU Make and AFS are capable of multiple files, they are command oriented tools and therefore they are still hard to be used by novices.

SemNet[2] and Information Visualizer[1] are also pioneering work in 3D visualization. However, these systems can display only one relation at a time. Although users change their viewpoints, the obtained information is the same. For example, Cone Tree[5] shows one hierarchical structure whichever viewpoints users choose. On the other hand, our 3D framework[3] makes it possible to display two different relations without disturbing each other.

5 CONCLUSION

A complete 3-D framework for visualizing version/module information was proposed. Based on this framework, the VRCS was implemented. Tree representations of version histories and visual interfaces corresponding to complicated RCS commands help novices. Moreover, our system is not only a visualization system of RCS, but also a system to manage an entire software system. Therefore, the system would also help experts.

Our goal is not to make a new version control tool, but to put current tools, such as SCCS/RCS and Make both of which are weak by themselves, good use of by giving effective 3D visual interfaces.

ACKNOWLEDGMENT

This work is supported in part by the Foundation of "Hattori-Hokokai."

REFERENCES

- [1] S. K. Card, G. G. Robertson, and J. D. Mackinlay. The Information Visualizer, an information workspace. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91)*, pages 181–188. ACM Press, 1991.
- [2] K. M. Fairchild, S. E. Poltrock, and G. W. Furnas. SemNet: Three-dimensional graphic representation of large knowledge bases. In R. Guindon, editor, *Cognitive Science And Its Applications For Human-Computer Interaction*, pages 201–233. Lawrence Erlbaum Associates, 1988.
- [3] H. Koike. The role of another spatial dimension in software visualization. *ACM Trans. on Information Systems*, 11(3):266–286, July 1993.
- [4] A. Lampen. Advanced file to attributed software object. In *USENIX-Winter '91*, 1991.
- [5] G. G. Robertson, J. D. Mackinlay, and S. K. Card. Cone Trees: Animated 3D visualizations of hierarchical information. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91)*, pages 189–194. ACM Press, 1991.