Visualizing Software Release Histories with 3DSoftVis

Claudio Riva
Software Technology Laboratory
Nokia Research Center
P.O. Box 407, FIN-00045
Helsinki, Finland
+358 40 749 0596
claudio.riva@nokia.com

ABSTRACT
3DSoftVis is a three dimensional visualization tool developed for the analysis of the evolution of an industrial software system. This paper briefly introduces the technique based on the Software Release Histories and present the main capabilities offered by the tool for their analysis.

Keywords
Information visualization, software evolution, maintenance

1 INTRODUCTION

The evolution of a large software system is a marvelous process. Release after release, programmers modify the system adding new features and changing the existing ones. The evolution happens as a global process driven more by changes in functionality than by low-level tinkering with code. At the architectural level, changes are reflected by added, removed or modified software modules.

The *system release* is the mechanism to control the implementation of the changes. Changes become an integral part of the system when the new release is delivered and no other modifications are allowed. At a generic release, a system may consist of components developed in different moments of its life. The system release identifies uniquely the system configuration and composition at a specific time. Similarly to the internal rings of a tree-trunk that unveils the tree history, software releases represent fixed events in the system history.

The history of a software system's evolution contains a wealth of hidden information valuable to managers and engineers in the development and enhancement of the system. To our knowledge, such information is rarely used or even taken in consideration.

Our work investigated the use of three dimensional visualization techniques to make such information more concrete and visible. In particular, we considered two graphical techniques: color and third dimension. The visualization techniques have been applied to the analysis of the evolution of an industrial software system. A detailed description can be found in the work of Jazayeri et al. [1] [2] [3].

To support our techniques, we have developed a visualization tool called 3DSoftVis. This paper introduces the tool and gives an overview of its main features. Although the tool is specialized to the analysis of the industrial case study, we believe that the techniques are generally applicable to the analysis of software release histories and should become part of standard tool for software management activities.

2 VISUALIZING THE SOFTWARE RELEASE HISTORIES

The case study's architecture is organized as a *layered system* with four levels: *system, subsystem, module* and *program*. Each level consists of one or more elements. Table 1 shows an example of the software release history. Each row represents the version numbers of a *program* element. The first and second column (labeled Sub and Mod) contain respectively the *subsystem* and *module* to which the *program* belongs. The third column (labeled Prog) contains the name of the *program*. The columns labeled from 1 to 20 represent the twenty system releases that have been considered. The version number of each element is the RSN of the release where it had the latest change. For example, if a *program* changes its implementation at release 1, 2 and 5 then its version numbers are the sequence < 1 2 2 2 5 5>.

The database is populated with 20 releases of the software product. Each release contains 8 *subsystems*, 47 to 50 *modules* and 1500 to 2300 *programs*.

Table 1. The software release history.

Sub	**Mod**	**Prog**	**1**	**2**	**3**	**4**	**5**	**6**	**…**	**20**
Sub1	Mod A	Prog A	1	2	2	2	5	5	…	18
Sub1	Mod A	Prog B	0	2	3	4	5	0	…	20

The software release history consists of three entities: time, system structure and version numbers. The entities are visualized in one three-dimensional diagram. The coordinate *z* stores the time information expressed in release sequence number (RSN). The system structure is displayed by 2-D or 3-D graphs. The graphs are spatially positioned along the coordinate *z* at the value of their own RSN. Four 3-D graphical objects are used to visualize the system elements: cubes, spheres and bars for the architectural entities and lines for the dependencies among them. Version numbers are shown using colors. Colors are associated with version numbers through a color scale. The structural entities are painted according to their version numbers. A percentage bar is a graphical object that offers a compact representation of a group of elements. The size of each colored block that composes the bar is proportional to the percentage of modules that have the same version number.

3 OVERVIEW OF 3DSOFTVIS

3DSoftVis consists of three components: a database, a 3-D visualization engine and the graphical user interface. The database contains the data regarding the evolution of the software system. The 3-D visualization engine transforms the data extracted from the database into 3-D models. The user interface presents the graphs to the user through multiple windows and allows the viewer to customize the views interactively. Figure 5 shows a snapshot of the tool.

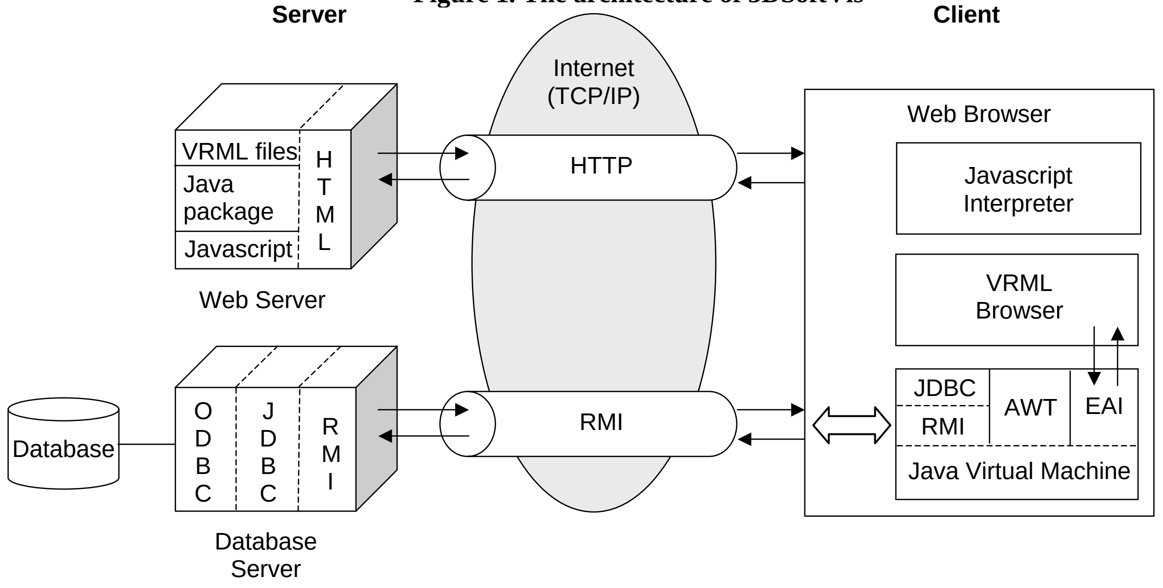
In the design of the 3DSoftVis, we decided to adopt an innovative approach based on recent World Wide Web technologies. Our goal has been to develop an application that can be published on the web. The tool is currently available on the web for a demo [6]. The tool has been developed using web components such as Java, VRML(Virtual Reality Modeling Language), JDBC and HTML. The architecture of 3DSoftVis is depicted in Figure 1. A more detailed description can be found in [4]. Concerning the architecture, the major features of 3DSoftVis are summarized below.

- World Wide Web Application: the web architecture makes the tool accessible from the network. Users can run the tool simply accessing the web link. On the client side 3DSoftVis requires a minimal configuration and relieves users from installation troubles. The application and the database reside on a remote server and they can be easily upgraded for maintenance.
- Graphical engine based on VRML: VRML is an easy solution for developing 3-D graphical tools. The choice of VRML simplified considerably the implementation of the graphical engine. VRML is also a standard web component that can be easily integrated in a web application.
- Collaborative environment: data and the application for their analysis are bound together on the server machine, so that the most updated database and the most recent release of the tool are immediately available. Groups of researchers can use 3DSoftVis by sharing the same working environment without the constraint of having to be located nearby or working on the same machine-platform.

4 3DSOFTVIS AT WORK

3DSoftVis offers three kinds of visualizations for the analysis of the evolution of the software release histories. We used these three different views to examine the industrial case study. The views are described below.

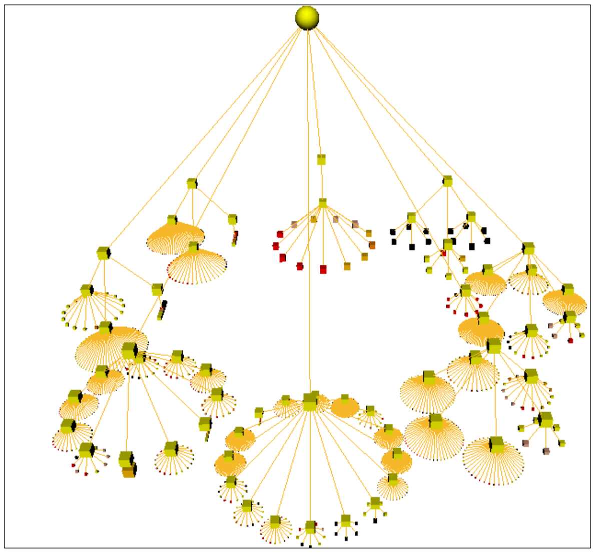
Figure 1. The architecture of 3DSoftVis



3-D Visualization of system structure at a single release

The case study has a tree hierarchical structure. 3DSoftVis allows us to visualize the tree hierarchy with 2-D and 3-D tree graphs. 3-D graphs are implemented using the Cone Tree technology [5]. Figure 2 shows the whole structure of one system release of the case study. The 3-D layout allows the viewer to navigate the system structure and zoom on subsystems.

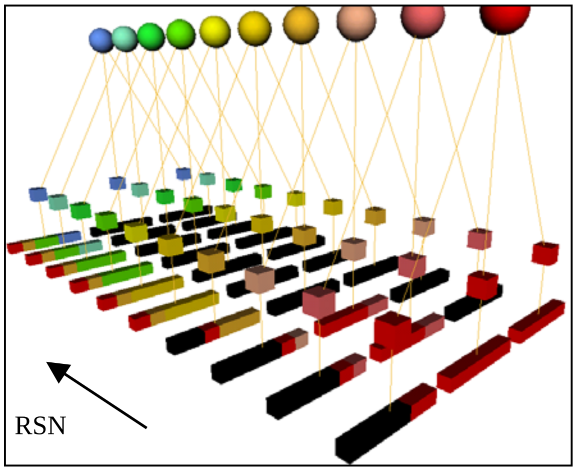
Figure 2. The structure of the system in 3D.



3-D visualization of historical evolution

This view allows us to visualize multiple releases of the system in order to examine the historical evolution. Each 2-D tree graph represents the system structure at a particular release. The third coordinate is expressed in RSN. Figure 3 shows an example. The 3-D diagram represents 10 releases of one *subsystem*. The 2-D graph visualize all the *programs* and *modules* that belong to the subsystem.

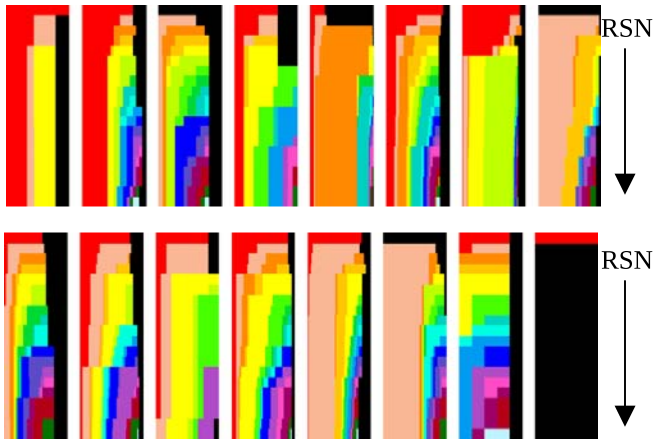
Figure 3. Visualizing the history of one *subsystem*.



2-D Visualization of historical evolution

2-D visualizations can be obtained by projecting the 3-D diagram onto 2-D space. The picture obtained by the projection reveals a concise and informative representation of system evolution. Figure 4 shows an example. Each picture visualizes the evolution of one *module* element through the percentage bars. Each row is associated with a system release. For each system release (from 1 to 20), the percentage bars are displayed. The percentage bars show with the same color the percentage of *programs* that have the same version number. This representation highlights the critical times—when major changes seem to have happened—in the evolution of the whole *module*.

Figure 4. 2-D view of the evolution of modules contained in one subsystem.



5 BENEFITS OF 3DSOFTVIS

3DSoftVis allowed us to examine the evolution of the large industrial case studies in a simple and effective way. We summarize the major benefits the tool gave us.

3-D layout and navigation

Visualizing both system structure and historical evolution in one view, the user can analyze both the information at the same time. The user can also virtually navigate into the representation to find the best perspective to look at the data. New representations can be easily obtained by just rotating, zooming, projecting or moving the graphs.

Being an interactive tool, the user can also retrieve information concerning the structural elements, like name and version numbers, by simply clicking on the graphical objects. This is faster and more attractive than examining textual tables.

Observations on the historical evolution

Color visualizations can move the process of understanding information from being a cognitive task to a perceptive task. For example, the 2-D representations make evident the useful trends in the historical evolution of the modules.

The stability of the modules is evident by two factors that are clearly depicted in the representations:

- changing rates: changes of versions are visualized as changes in colors. This gives to the viewer a qualitative and intuitive indication of the changing rates. High changing rates (unstable module) are identified by regions where colors change very quickly.
- Growing rates: the black color represents those elements that have been removed from the module or that have not been implemented yet. The modification in size of the black region is proportional to the growing rates of the module.

Pattern Detection

Visualization enables the humans' pattern matching skills. The human visual system is able to detect regions characterized by repetitive use of the same shape, same color, same filling or the same texture. Information visualization exploits such ability for the analysis of numerical data. Indeed, the identification of visual patterns leads to the detection of dependencies and relationships among system elements. In Figure 4, it is possible to identify modules that have the same evolution pattern as they have similar visual patterns.

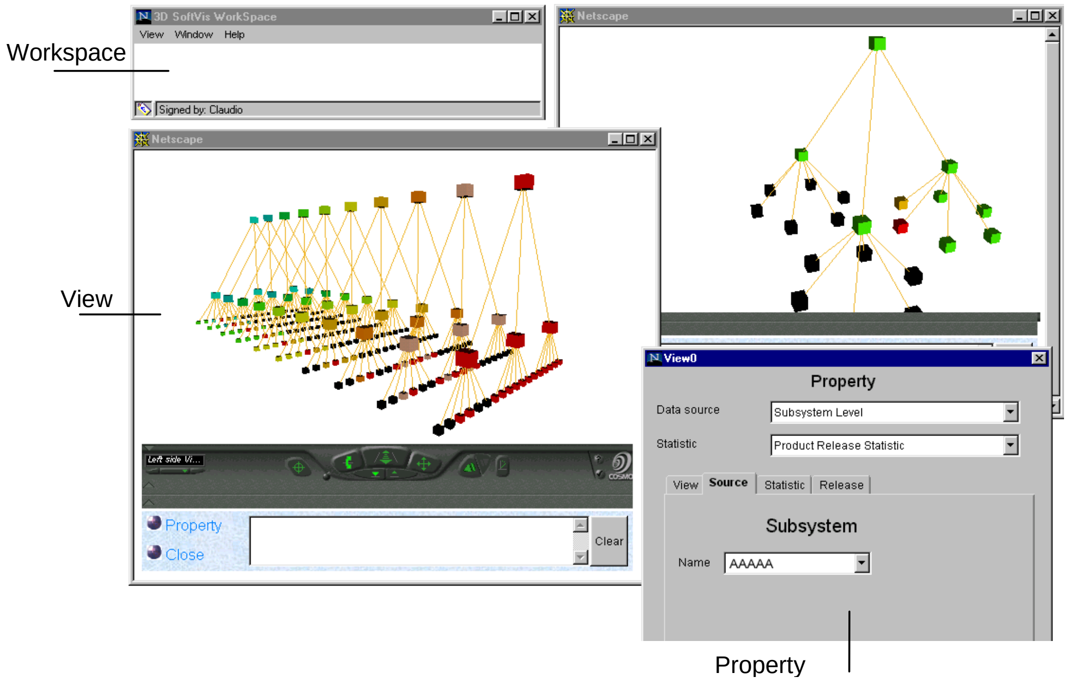
ACKNOWLEDGEMENTS

I gratefully acknowledge Mehdi Jazayeri and Harald Gall for their invaluable contribution and all the people at the Distributed Systems Group at the Technical University of Vienna. This work was supported by the European Commission within the ESPRIT Framework IV project ARES (*Architectural Reasoning for Embedded Systems*).

REFERENCES

1. Jazayeri M., Riva C., Gall H., Application of Information Visualization to the Analysis of the Software Release History, *Proceedings of the Joint EUROGRAPHICS and IEEE TCVG Symposium on Visualization (VisSym '99)*, May 26-28, Vienna, Austria, 1999, 237-246.
2. Gall H., Jazayeri M., Riva C., Visualizing Software Release Histories: The Use of Color and Third Dimension, *Proceedings of the International Conference on Software Maintenance (ICSM '99)*, Aug 30-Sep 3, Oxford, England, 1999, 99-108.
3. Riva C., Visualizing Software Release Histories: The use of Color and Third Dimension, Master's Thesis, Politecnico di Milano, Milan, Italy, June 1998. Also in: http://www.infosys.tuwien.ac.at/~riva/Docs/bibl/riva98/full
4. Jazayeri M., Riva C., Experiences with developing Native World Wide Web Applications, Internal Report, at http://www.infosys.tuwien.ac.at/~riva/Docs/bibl/jaz98a/full
5. Robertson G. G., Mackinlay J. M. and Card S. K., Cone Trees: Animated 3D Visualizations of Hierarchical Information, *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '91)*, ACM Press, 1991, pp. 189-194.
6. 3DSoftVis: demo at http://www.infosys.tuwien.ac.at/visualization

Figure 5. A snapshot of 3DSoftVis.


Workspace
3D SoftVis WorkSpace
View Window Help
Signed by: Claudio
Netscape
View
Netscape
Property
Close
Clear
View0
Property
Data source
Subsystem Level
Statistic
Product Release Statistic
View Source Statistic Release
Subsystem
Name
AAAAA
Property