White Coats: Web-Visualization of Evolving Software in 3D

Cédric Mesnage
Département d'informatique
Université de Caen, France

Michele Lanza
Faculty of informatics
University of Lugano, Switzerland

Abstract

Versioning systems to store, handle, and retrieve the evolution of software systems have become a common good practice for both industrial and open-source software systems, currently exemplified by the wide usage of the CVS system. The stored information can then be manually retrieved over a command line or looked at with a browser using the ViewCVS tool. However, the information contained in the repository is difficult to navigate as ViewCVS provides only a textual view of single versions of the source files. In this paper we present an approach to visualize a CVS repository in 3D (using VRML) by means of a visualization service called White Coats*. The viewer can easily navigate and interact with the visualized information.*

1. Introduction

The first software versioning systems appeared more than thirty years ago. Since then, software industry has recognized the importance of versioning systems and is using them extensively since more than two decades now [12] [16] [3], boosted by the emergence of open source software. Recording the history of a software system during its development has the advantage of allowing to reconstruct the original design intentions of the developers, as well as their subsequent variations in time. Versioning systems also allow efficient sharing of a project, and hence ease team software development. The facilities given by versioning systems and the amount of data retrieved fostered the research field of software evolution [10], whose goal is to analyze the history of a software system and infer causes to its current problems, and possibly predict its future.

One of the main difficulties in managing the complexity of evolving software systems is to handle the large amounts of data and present them to a software engineer in an intuitive and easy-to-understand way. Currently used means include textual representations by tools such as ViewCVS[1] and some graphical representations offered by tools such as the Maven project[2]. The problem with those approaches is that they are heavily based on CVS itself, *i.e.*, they only display the information provided by CVS without abstracting or aggregating it. However, they both come with an intrinsic advantage: The information is rendered as web pages and is thus easily accessible.

Other, more advanced tools have been implemented recently in the area of mining software repositories ([17], [11], [9] and [14]) but most of them are based on standalone tools and only few of them use visualisation to present the extracted information. We are convinced that providing a solution which presents information on an easily accessible platform, *i.e.*, a web browser, is crucial for a widely spread usage of such tools.

Our approach provides easy (web)access to versioning information by rendering the information on web pages. Moreover, by means of a VRML plugin the user can navigate the 3D visualizations, *e.g.*, zoom, filter, fly through, interact with, inspect, etc. any of the visualized entities. Our approach is complementary to textual approaches, *i.e.*, when the user has visualized and identified an entity of interest he can quickly access the underlying information. In this article we present *White Coats*, a 3D web visualizer that allows inspecting and navigating the repositories of software systems.

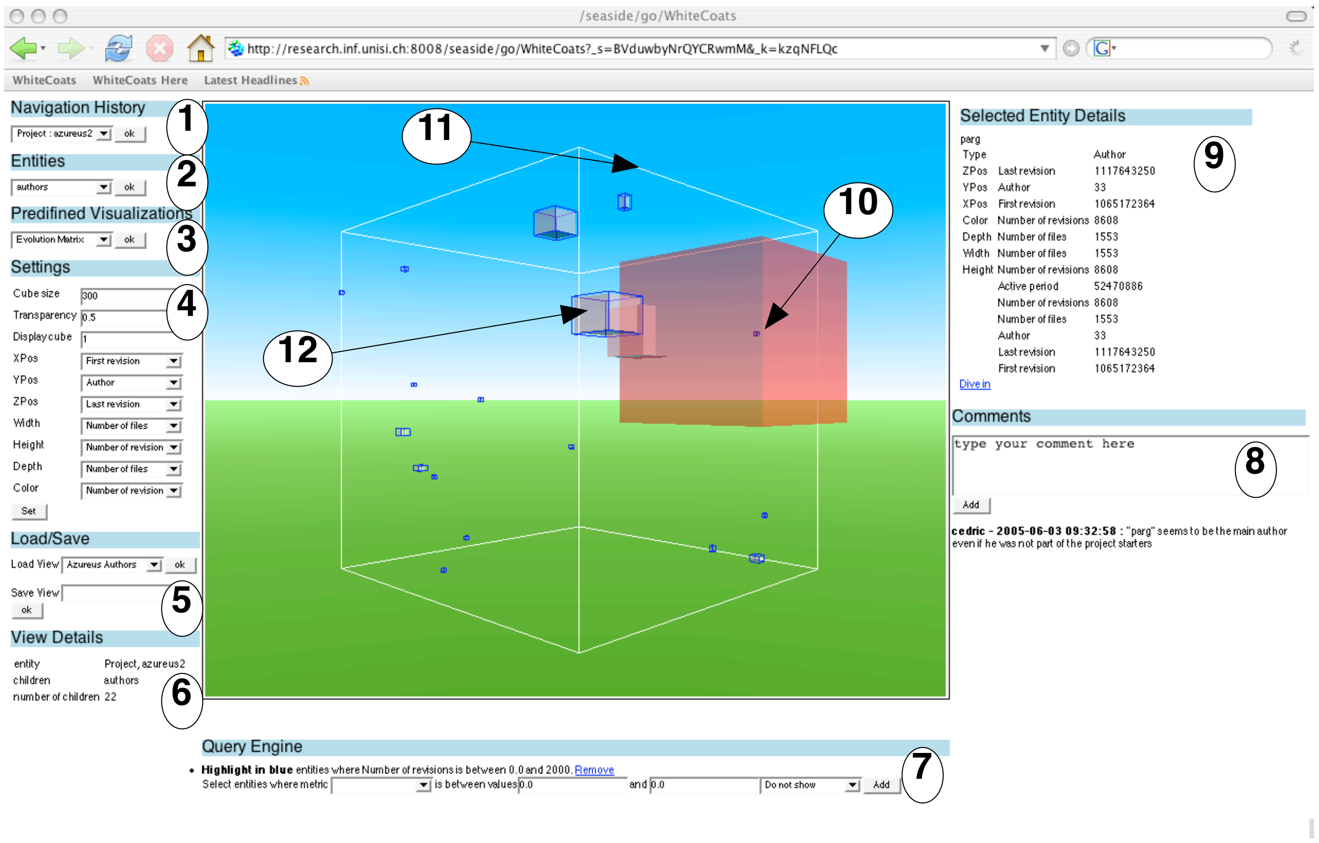
Structure of the paper. In the next section we present the overall architecture of *White Coats* and the visualization and interaction means that it provides. We then present example visualizations obtained from various case studies. We conclude by discussing our findings, looking at related work, and describing our future work in this domain.

2. White coats

The user interface of *White Coats* (see Figure 1) is composed of one central 3D viewport (navigable and interactive) and many panels that provide complementary information:

1 See http://viewcvs.sourceforge.net/ for more information.

2 See http://maven.apache.org/ for more information.



/seaside/go/WhiteCoats

http://research.inf.unisi.ch:8008/seaside/go/WhiteCoats?_s=BVduwbyNrQYCRwmM&_k=kzqNFLQc

WhiteCoats WhiteCoats Here Latest Headlines

Navigation History

Project : azureus2 ok

Entities

authors ok

Predifined Visualizations

Evolution Matrix ok

Settings

Cube size	300
Transparency	0.5
Display cube	1
XPos	First revision
YPos	Author
ZPos	Last revision
Width	Number of files
Height	Number of revision
Depth	Number of files
Color	Number of revision

Set

Load/Save

Load View Azureus Authors ok

Save View

ok

View Details

entity	Project, azureus2
children	authors
number of children	22

Selected Entity Details

parg

Type		Author
ZPos	Last revision	1117643250
YPos	Author	33
XPos	First revision	1065172364
Color	Number of revisions	8608
Depth	Number of files	1553
Width	Number of files	1553
Height	Number of revisions	8608
	Active period	52470886
	Number of revisions	8608
	Number of files	1553
	Author	33
	Last revision	1117643250
	First revision	1065172364

Dive in

Comments

type your comment here

Add

cedric - 2005-06-03 09:32:58 : "parg" seems to be the main author even if he was not part of the project starters

Query Engine

- **Highlight in blue** entities where Number of revisions is between 0.0 and 2000. Remove

Select entities where metric is between values 0.0 and 0.0 Do not show Add

Figure 1. The user interface of WhiteCoats.

1 Navigation History: This panel gives the user the ability to navigate to previously visualized entities. Diving into an entity adds that entity to the history list.

2 Entities: When an entity is visualized, the user selects the kind of entities related to it that he wants to display. For example if a directory is the main entity, the contained subdirectories or files can be displayed.

3 Predefined Visualizations: A set of applicable, predefined visualizations (*e.g.*, "Evolution Matrix"). When one is selected, the other panels are updated accordingly.

4 Settings: A visualization can be configured by setting the associations between visual attributes (*e.g.*, width, height, position, *etc.*) and metrics (*e.g.*, "Number of files") or by changing the size of the reference cube and the transparency of the entities (useful when they overlap).

5 Load/Save: Saving a view stores its configuration, *e.g.*, entities displayed, metrics settings, *etc.*. The saved view is shared with other users exploring the same project. Loading a view updates all panels.

6 View Details: A summary of the displayed data, such as the name of the visualized entity, the type of entities displayed, and the number of entities in the visualization.

7 Query Engine: One can add queries to filter the displayed entities. By setting a lower and upper threshold on a metric entities are selected on which an action is performed (*e.g.*, "Do not show" or "Highlight").

8 Comments Panel: Users can read and write comments associated to an entity. The comments are displayed with a date and the comment's author user name.

9 Selected Entity Details: When a user clicks on an entity, it displays related information, such as its name, the metrics associations, *etc.*. A "Dive in" link makes the selected entity become the main entity, *i.e.*, the internals of that entity are displayed.

10 A visualized entity: An entity is displayed as a box in the visualization, its position, size and colors depending on the associated metrics.

11 The reference cube: A transparent wireframe containing all displayed entities. The reference cube eases 3D navigation. By setting the size of the cube, it scales the contained boxes.

12 A highlighted entity: When an entity is highlighted because if matches a query it displays a wireframe box around it. The wireframe color can be set.

2.1. Handling Evolution Information

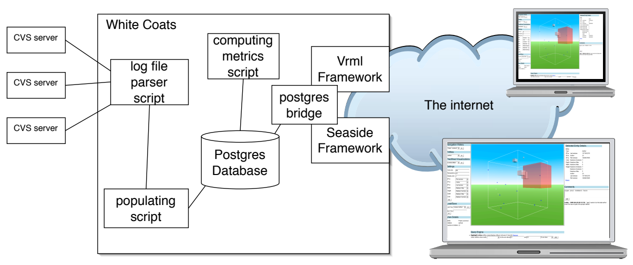


Figure 2. The overall Architecture of White Coats.

To visualize evolution information, White Coats uses a set of scripts to populate a database which is then the source of information for the visualizations. The populating process (Figure 2) is done as follows:

1. The user indicates the URL of a CVS repository.

2. A set of scripts is executed that connect to the repository and download all log files which contain information about every revision of every file. The log files are parsed and a database is populated with the retrieved information (see Figure 3 for the DB schema we use). A set of directly retrievable metrics (*e.g.*, number of added lines) and other information (*e.g.*, id of the author) is also stored into the DB.

3. A second set of scripts computes information that cannot be retrieved directly (see Table 1) and stores them into the DB too.

In case we have already done this for a certain repository, our service checks and updates the information.

Retrieved and computed metrics. Some metrics are retrieved from the CVS logs (such as the number added or removed lines), while others are computed in a second phase. In Figure 1, we list some of the metrics. The metrics are computed in a preprocessing step; we considered computing them on-the-fly but did not due to scalability problems (a project like mozilla has approx. half a million revisions).

Database model. The database model (3) is inspired from the RHDB (Release History Database) [2]. The main difference between the RHDB and our model is that we store metrics in the database.

Property Name	**Entity types**
Retrieved from CVS Log Files	
Added/Removed lines	Revision
Author	Revision
File	Revision
Revision Date	Revision
Simultaneous Revisions	Revision
Computed	
Depth	File, Directory
First/Last Revision	File, Directory, Author
Added/Removed lines	File, Author
Active period	File, Author
Number of Authors	File, Directory(?)
Number of Revisions	File, Directory(?), Author
Number of Files	Directory, Author
Parent Directory	Directory

Table 1. Entity Properties.

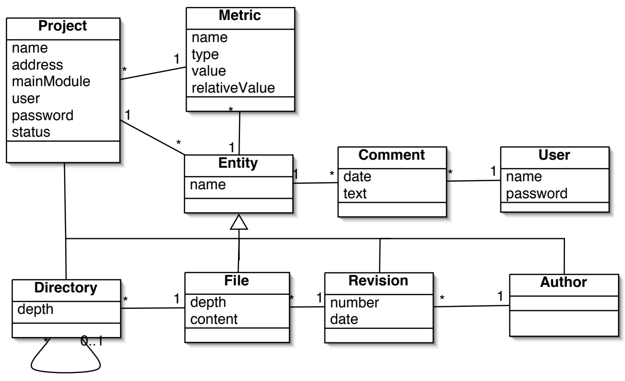


Figure 3. The Database model.

2.2. Visualizing Evolution Information

The retrieved evolution information is visualized in 3D using VRML. Extending the concept of polymetric views [8] to 3D the user can set a number of metrics which influence the position and shape of the visualized entities (see Figure 4). Currently we can map up to 6 metrics (3 for the position, 3 for the shape) and additionally represent more information on texture and transparency of the visualized entities. As we have seen with 2D polymetric views, the point is not to display as much information as possible, but to discover views whose combinations of metrics convey certain types of (useful) information to the viewer. In the next section we present some example visualizations.

Reference Cube. As we see in Figure 1, the displayed entities are displayed within a surrounding wireframe cube. This *reference* cube gives an idea of the orientation to the user. We can also set the size the cube, giving the user the ability to spread the entities if they overlap.

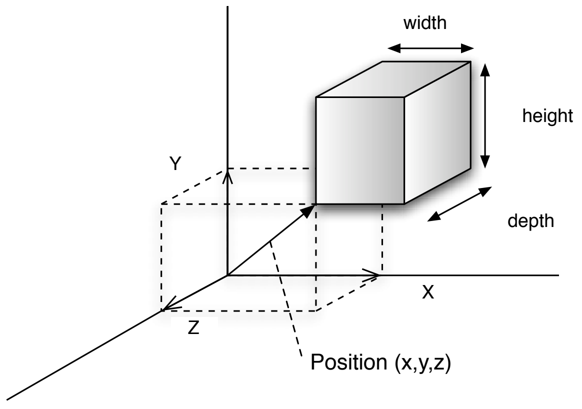


Figure 4. 3D Polymetric view principles.

Horizon. As a supplemental navigation aid we also provide a ”sky” and a ”ground” background to let the viewer know into which direction he is looking.

Normalizing metrics. We use metrics in the visualizations to assign values to the dimensions of the entities to be displayed. The distribution of the values of the metrics would however lead to unreadable visualizations. The relative value of a metric is a number between 0 and 100 which we use in the rendering process so that the dimensions associated with metrics always have values between 0 and 100. We normalize metrics to make them fit within the reference cube, *i.e.,* the reference cube has a side length of 100.

Interactivity. White Coats uses VRML for the visualizations, whereas navigation and interactivity are given by the used VRML browser plugin [3] we use. This plugin allows to navigate in the view by rotating around the objects or flying into it. We defined a set of viewpoints (front, back, bottom, right side, left side) which can be chosen during the navigation.

The visualized entities can be selected using the mouse pointer. When clicked, the user can dive into the entity, inspect it, *etc.*. In case of a diving, a new visualization is created, *i.e.,* a new VRML document is created on the fly and displayed.

3. Selected Examples

The following selected examples have been made on the bittorrent project Azureus[4] which at the time of writing of this paper was tje most active project on sourceforge. Due to space limitations we focus on some views to show the possibilities of White Coats.

3 In our case, we use the Cortona plugin. Accessible at: http://www.parallelgraphics.com/products/cortona/

4 See http://azureus.sourceforge.net/

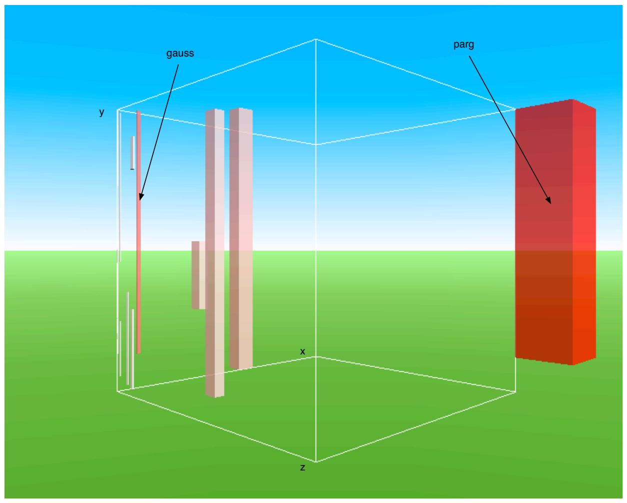


Figure 5. The Author’s activity of the Azureus project.

Authors view. In this view (see Figure 5), the position on the y axis is the first revision date of the author and the height corresponds to the active period. The position on the x and z axis are respectively the number of revisions and number of files, the width and depth are also the number of revisions and the number of files. The color represents the number of lines added by the author. We can see that one author worked on most files and did most revisions (named “parg”), this may be the one to contact. Another author (named “gouss”) did not work on so many files but added a lot of lines (his color is red).

Evolution Matrix view. In Figure 6, we show the Evolution Matrix ([6]) of the project Azureus which represents over 15’000 revisions. On this visualization the x axis corresponds to the date of the revision, the y axis to the author and the z axis to the file. The width, height and depth are not used here (every box has the same size). The color is the number of revisions which appeared at the same time and we added to this a query which highlights in blue the revisions of more than 300 added lines. We can see that one particular file suffered of many revisions of more than 300 added lines by the same author. This file may be an interesting candidate to refactoring. This example shows also the limits of the paper media to explain the possibilities of White Coats: the user would now navigate (”fly through”) to this file history and have a closer look.

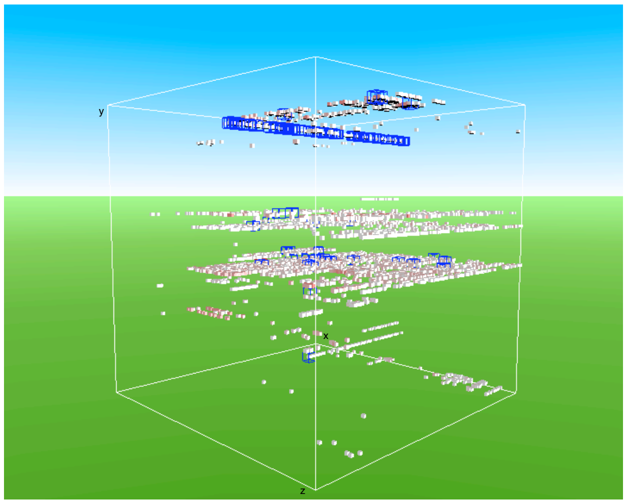


Figure 6. The Evolution matrix of the Azureus project.

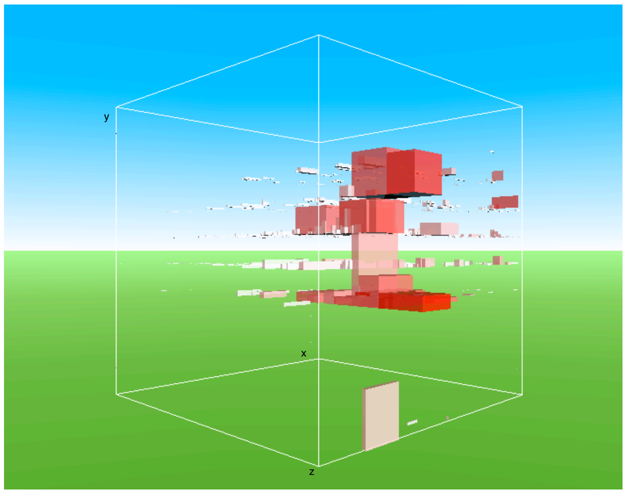


Figure 7. The Files Activity view of the Azureus project.

Files Activity view. In Figure 7, we show all the files of the Azureus project (over 3500 files). The x axis is the date of the first revision on that file, the y axis is the depth of the file in the directory and the z axis the date of the last revision. The width is the number of authors, the height the number of revisions. The depth and color are the number of removed lines and the number of added lines. We see that the most active files (most number of revisions) have been revised recently but are not necessarily the files with the most number of added/removed lines.

4. Discussion

Our goal with this research is to obtain an integrated, interactive view of CVS repositories which is easily accessible, *i.e.*, over a web browser. White Coats allows the user to easily select entities of interest and inspect them. Moreover, the fully automated information extraction could make White Coats become a central point of approach for CVS repositories, instead of going over ViewCVS. We are working currently on publishing White Coats as a publicly available ”web service”.

Still, the approach has some limits:

Scalability. Using 3D visualization can be useful to display several metrics in the same view. However, when several entities are displayed, it becomes less readable. Navigating through the view by rotating, and the use of the reference cube helps the user, but the view can still be confusing. We plan to add 2D visualization to White Coats so that the user can choose between 2D or 3D visualizations.

Usability. A major drawback of White Coats (although not due to White Coats!) is the 3D navigation itself given by the VRML plugin. The user can indeed fly through the 3D space, but not with the ease that would make it the preferred way of navigation. Instead we found ourselves often applying queries to filter out irrelevant details and only when few entities remained in focus we would actually start 3D navigation.

5. Related Work

Software evolution has established itself as a research field in the past few years, leading to many publications in this area.

Lanza’s Evolution Matrix [7] visualizes the system’s history in a matrix in which each row is the history of a class. A cell in the Evolution Matrix represents a class and the dimensions of the cell are given by evolutionary measurements computed on subsequent versions. However, the Evolution Matrix has been implemented in a stand-alone tool (CodeCrawler) and uses only two-dimensional visualizations.

Jazayeri analyzed the stability of the architecture [5] by using colors to depict the changes.

Taylor and Munro [15] visualized CVS data with a technique called *revision towers*. Ball and Eick [1] developed visualizations for showing changes that appear in the source code.

Gulla [4] proposes multiple visualization of C code, but to our knowledge there was no implementation. Collberg *et* *al.* used graph-based visualizations to display the changes authors make to class hierarchies. However, they do not give any representation of the dimension of the effort and of the removals of entities.

Rysselberghe and Demeyer [13] propose a simple visualization of a system history from a CVS repository by creating a matrix where the columns represent files ordered by names and lines represent the time. Wu, Holt and Hassan [18] introduce a spectrograph visualization to render software evolution. These approaches offer visualizations of software repositories, but no interaction with the visualizations are possible.

A major drawback of the above mentioned and many other approaches is that many of them are either implemented in stand-alone tools (thus hindering widespread usage) or in IDEs like eclipse (thus being targeted towards a specialized audience, *i.e.*, developers). One of the advantages of our approach is that the visualizations are displayed within a web browser, thus being accessible by anyone (*e.g.*, developers, managers ...).

6. Conclusion

In this paper we presented the 3D web visualizer White Coats, which gives the ability to the developer to visualize the evolution of a software repository in a web-site. We presented some visualizations of the project Azureus. We discussed our work, and related it to other tools.

We plan to add a 2D visualization using the same polymetric views. It will make the visualization process easier for large projects where rendering half a million boxes begin to be difficult for the VRML plugin. In fact a 2D visualization will make the web user interface easier to navigate. We are currently also planning to apply data mining techniques to mine the relevant entities.

Acknowledgments

We would like to thank Romain Robbes for his comments on this paper. We also thank CHOOSE (Swiss Group for Object-Oriented Systems and Environments) for their financial support.

References

[1] T. Ball and S. Eick. Software visualization in the large. *IEEE Computer*, pages 33–43, 1996.

[2] M. Fischer, M. Pinzger, and H. Gall. Populating a release history database from version control and bug tracking systems. In *ICSM '03: Proceedings of the International Conference on Software Maintenance*, page 23, Washington, DC, USA, 2003. IEEE Computer Society.

[3] D. Grune. Concurrent Versions system, a method for independent cooperation. Technical report, Vrije Universiteit, Amsterdam, Netherlands, 1986.

[4] B. Gulla. Improved maintenance support by multi-version visualizations. In *Proceedings of the 8th International Conference on Software Maintenance (ICSM 1992)*, pages 376–383. IEEE Computer Society Press, Nov. 1992.

[5] M. Jazayeri. On architectural stability and evolution. In *Reliable Software Technlogies-Ada-Europe 2002*, pages 13–23. Springer Verlag, 2002.

[6] M. Lanza. The evolution matrix: Recovering software evolution using software visualization techniques. In *Proceedings of IWPSE 2001 (International Workshop on Principles of Software Evolution)*, page to be published, 2001.

[7] M. Lanza and S. Ducasse. Understanding software evolution using a combination of software visualization and software metrics. In *Proceedings of LMO 2002 (Langages et Modèles à Objets*, pages 135–149, 2002.

[8] M. Lanza and S. Ducasse. Polymetric views — a lightweight visual approach to reverse engineering. *IEEE Transactions on Software Engineering*, 29(9):782–795, Sept. 2003.

[9] K. M. K. Laven, S. Roweis, and G. Wilson. Mining student cvs repositories for performance indicators. In *MSR 2005: International Workshop on Mining Software Repositories*, 2005.

[10] M. M. Lehman and L. Belady. *Program Evolution – Processes of Software Change*. London Academic Press, 1985.

[11] I. Neamtiu, J. S. Foster, and M. Hicks. Understanding source code evolution using abstract syntax tree matching. In *MSR 2005: International Workshop on Mining Software Repositories*, 2005.

[12] M. J. Rochkind. The Source Code Control System. *Transactions on Software Engineering*, 1(4):364–370, 1975.

[13] F. V. Rysselberghe and S. Demeyer. Studying software evolution information by visualizing the change history. In *ICSM '04: Proceedings of the 20th IEEE International Conference on Software Maintenance (ICSM'04)*, pages 328–337, Washington, DC, USA, 2004. IEEE Computer Society.

[14] J. Spacco, J. Strecker, D. Hovemeyer, and W. Pugh. Software repository mining with marmoset: An automated programming project snapshot and testing system. In *MSR 2005: International Workshop on Mining Software Repositories*, 2005.

[15] C. M. B. Taylor and M. Munro. Revision towers. In *Proceedings of the 1st International Workshop on Visualizing Software for Understanding and Analysis*, pages 43–50. IEEE Computer Society, 2002.

[16] W. F. Tichy. RCS — a system for version control. *Software — Practice and Experience*, 15(7):637–654, 1985.

[17] C. C. Williams and J. K. Hollingworth. recovering system rules from software repositories. In *MSR 2005: International Workshop on Mining Software Repositories*, 2005.

[18] J. Wu, R. C. Holt, and A. E. Hassan. Exploring software evolution using spectrographs. In *11th Working Conference on Reverse Engineering (WCRE'04)*, pages 80–89, November 2004.