Visualization to Support Version Control Software: Suggested Requirements

Xiaomin Wu
University of Victoria
xwu@uvic.ca

Margaret-Anne Storey
University of Victoria
mstorey@uvic.ca

Adam Murray
University of Ottawa
amurray@site.uottawa.ca

Rob Lintern
University of Victoria
rlintern@uvic.ca

Abstract

Many version control systems have been developed to manage both software version history and associated human activities with the intent of producing higher quality software. To better understand and explore the vast information these version control systems portray, several approaches have been conducted to apply visualization techniques in this domain, resulting in a variety of tools. However, these tools have rarely been evaluated and hence we are unable to tell how successful these information visualization techniques are for understanding and exploring version control information. Moreover, there is lack of requirements for how such a visualization tool can support version control activities. This paper describes a set of requirements for visualization support in a version control tool. We also present a tool called Xia, which was developed for the navigation and exploration of software version history and associated human activities. Moreover, we conducted an exploratory user study to test if the functionality of the Xia tool meets these requirements and if there are requirements we missed -- the results are documented. This position paper ends with a question – how should we proceed next with our research? We intend to refine the requirements and seek directions for future exploration.

1. Introduction

Version control systems are becoming an increasingly important tool for software development projects, especially when the development tasks are performed in a team environment. Presently, most medium to large-scale software projects are developed in association with a version control tool. A large amount of information is generated and stored in the repositories of these version control tools. What does this information mean to the software development process? Can this information be used in a meaningful way to help with team work? If so, how does the presentation of this information assist software development? To answer these questions, we conducted a preliminary survey of five version control systems in the spring of 2002. In this survey, we posed questions related to the functionality and ease of use of version control systems, as well as what data is important to support team collaboration. The results of this survey highlighted that although the features of version control systems are considered adequate; the interfaces of these systems are not satisfactory for users to understand and explore version control information. Also, our survey demonstrated that the most prominent concerns related to a team development task include:

- What happened since I last worked on the project (types of events, such as new file added, file modified, etc.)?
- Who made this happen?
- Where did this take place (location of the new file, change, deletion, etc.)?
- When did this happen?
- Why were these changes made (what is the rationale of the designer(s) who made the change)?
- How has a file changed (exact details of the change, as well as relationship to other files)?
- What is the history of a particular file?

We name this problem the "5W+2H" problem for brevity, referring to the 5W's, what, who, where, when, why, and 2H, how, history, above. When many of the 5W+2H questions remain unresolved, developers may feel like they are working in a void, and progress will be greatly hindered. More importantly, if the 5W+2H problem is not properly addressed, we cannot explore how people work in teams on software projects. When this problem is related to the entire software project, participants in our survey stated that they would like to have an overall view of the entire dataset. They believe that an overall view showing data related to all people's work would enable them to collaborate even better. This is because they will have more awareness of each other's activities and how other people's work may be relevant with their own work.

On the basis of these findings, we conjectured that applying information visualization techniques to a version control system might resolve these problems. Consequently, we investigated related research and noticed that some approaches have been deployed as in the following projects: Seesoft [1, 7], Beagle [17], CVS Activity Viewer [4], and others. These tools used some kind of visualization and query mechanisms to display and explore data from version control systems**.** However, these tools don't provide requirement analysis and have rarely been evaluated and hence we are unable to tell how successful these visualization techniques can be for assisting people in understanding and exploring version control information.

In this position paper, we present Xia, a version control visualization tool, which is tightly integrated with a full-featured IDE, Eclipse [5]. In Xia, advanced visualization techniques can be used for browsing and interactively exploring the data in a CVS repository. A preliminary user study was also conducted to evaluate both the requirements we identified through the survey and to test if the tool satisfactorily meets these requirements

Section 2 introduces our approach to the design and implementation of Xia. The details and results of our user study are described in Section 3. In Section 4, we outline future work and pose questions about how to improve our requirements and further evaluate our tool.

2. Approach

In our approach, we elected to focus our tool on the version control system known as CVS [3]. CVS is freely available open-source software that is widely used. We believe the widespread user base will make it easier to evaluate the effectiveness of our tool, as users will be easier to find. Our previous experience [8, 10] of plugging a visualization tool, SHriMP [14], into the Eclipse platform [6], encouraged us towards an approach of using the Eclipse platform as a framework for the integration of:

(1) The Eclipse CVS plug-in, a CVS interface plugged into the Eclipse platform, through which the CVS repository information could be accessed and retrieved;

(2) The Eclipse JDT (Java Development Tools) plug-in, which provides the workspace information for a particular Java project and;

(3) The SHriMP visualization engine, a domain-independent information visualization tool developed at the University of Victoria.

Xia is the result of an integration and customization of these components. The Eclipse CVS plug-in [9] and the JDT plug-in serve as data backends for Xia. The SHriMP visualization tool is customized and used by Xia as a visual front-end for the back-end data. Figure 1 illustrates the architecture of Xia.

In the following subsections, we look into the data we obtained from a CVS repository, and describe how we design the visualization in our tool to help answer the questions we raised before.

2.1. Data acquisition and analysis

The CVS repository is a good resource of information for helping to resolve the 5W+2H questions. We believe that pertinent information can be obtained and visualized. For instance, the log message in CVS contains the record of each commitment, including:

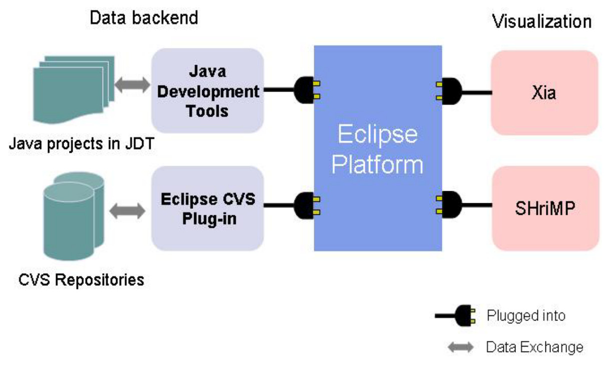


Figure 1. The architecture of Xia

- The author *who* made the commitment;
- The comments made by the author of *what* was changed and hopefully *why* it was changed; and
- The time and date *when* the file revision was created.

To understand *how* a change occurs, we propose the diff function of CVS helps. The location of the changed file in the repository hierarchy helps determine *where* the change takes place. In our tool, the information we required was retrieved directly from both the CVS repository via the Eclipse CVS plug-in, and the JDT in Eclipse, or the information was calculated from the retrieved data.

By analyzing data from the CVS repositories, we classified data into two categories: the software artifacts, including files, folders, and other code-level entities, and associated revision attributes. We attached the following attributes (see Table 1) to each of the file revision.

Table 1. File revision attributes

Attribute Name	**Data Resource**	**Data type**
File revision number	Retrieved	Ordinal
File revision tags	Retrieved	Nominal
Date of last commitment of a file revision	Retrieved	Ordinal
Author who changed the file most recently	Retrieved	Nominal
Author who changes the file most times in a particular time period	Calculated	Nominal
Comments associated with each commitment	Retrieved	Nominal
Number of changes associated with a file revision	Calculated	Ordinal
History of a file	Retrieved	Nominal

These attributes reflect human activities that concern people in answering the 5W+2H questions. They have been classified into two categories according to their data types, nominal and ordinal. Nominal attributes are strings whereas ordinal attributes have numeric or ordinal values.

2.2. Visualization

In this section, we describe the visualization of software artifacts and associated version attributes. Then we outline our method of interactively exploring this information. Finally, we summarize how our visualization techniques were designed to answer the 5W+2H questions.

2.2.1. Visual representation of CVS artifacts. A single file revision in the CVS repository is mapped to a single node in SHriMP. Likewise, a folder containing file revisions is mapped to a parent node of file revision nodes.

In the Eclipse CVS plug-in, software in the repository is displayed in a tree-like hierarchical structure of folders and file revisions. This structure corresponds well to nested graphs in SHriMP, as illustrated in a screen shot of the CVS data in Fig. 2. In Fig. 2, parent folder nodes (shown in purple) encompass file revision nodes (shown in yellow). The outmost blue nodes represent two versions of the same project. Node size relates to the size of the content within the node, so the version on the left (which is graphically larger) contains more sub-nodes than the version on the right..

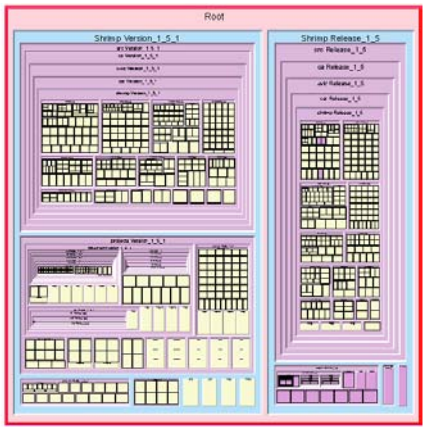


Figure 2. A nested graph showing two versions (in blue) of a software project.

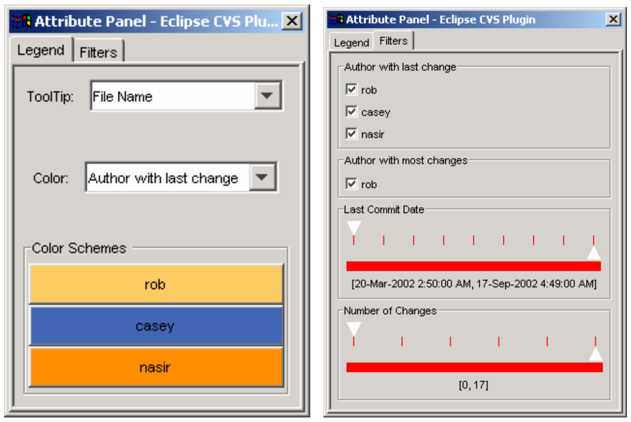
2.2.2. Visual representation of attributes. In addition to the pre-existing SHriMP visualization techniques, we developed an **Attribute Panel** for showing and querying attributes associated with file revisions. The attribute panel concept was originally developed at the University of Maryland, and combined with the Treemap visualization tool [11].

Appropriate visual variables are used to display the attributes [18]. For instance, using color, intensity, tool tips, size, and position to highlight nodes and accentuate their differences. Visual variable values are triggered through the **Attribute Panel**, as illustrated in Fig. 3.

Tool tips provide instant messages that are easily perceived during browsing. In Xia, all attributes in the CVS domain can be viewed with tool tips. The user selects which of the attributes to show in the tool tips.

Colors may be used for both nominal and ordinal attributes, though different color schemes are necessary for each type [2]. For nominal attributes, each of the values may be assigned a distinct color; whereas ordinal attributes may use color intensity instead of different colors (see Fig. 4). In Xia, the date of last commitment and number of changes are the two ordinal attributes that can be visualized using color intensity. These ordinal attributes are sorted in an old-to-new and few-to-more order respectively, and then each value is assigned an intensity of green (the default color). In our example, a more recent date is assigned a brighter green color. Figure 4 shows two screen shots of coloring nodes according to their nominal and ordinal attributes.

The arcs between nodes enable people to focus on a specific task and keep track of its relationship to other files in the project. This kind of awareness is very important for teams to collaborate on work effectively, especially if there are many dependencies between the different artifacts that are being worked on.


Attribute Panel - Eclipse CVS Plu...
Legend
Filters
Attribute Panel - Eclipse CVS Plugin

ToolTip: File Name
Color: Author with last change
Color Schemes
rob
casey
nasir

Author with last change
☑ rob
☑ casey
☑ nasir
Author with most changes
☑ rob
Last Commit Date
[20-Mar-2002 2:50:00 AM, 17-Sep-2002 4:49:00 AM]
Number of Changes
[0, 17]

(a) (b)

Figure 3. In (a), the user can change colors for the developers, and change how the tool tips appear; In (b), checkbox and double slider filters are used to filter nodes by their attribute values

2.2.3. Interactive exploration. The **Attribute Panel** also supports dynamic exploration using filters. Two kinds of filtering widgets have been developed for different attribute types. A checkbox filter, as illustrated in Fig. 3b is created for each of the nominal attributes. A checkbox filter consists of a set of checkboxes associated with each of the attribute values in the domain. An unchecked checkbox results in the corresponding nodes having equal attribute values being filtered from the screen. The other filtering widget, a double slider, is designed for ordinal attributes and is especially useful for dynamic queries. A double slider allows the user to select a range of values for query by adjusting the minimum and maximum value of the slider, and can also be used to select a single value by setting the minimum and maximum value of the slider to the same value. We implemented the double slider in Xia to filter two ordinal attribute values: Date of last commitment and Number of Changes. These two sliders could be used together to perform a multi-variable query. For example, if a programmer wants to look at the file that changed most frequently in the past week, he/she would be able to get the result by setting the Date of Last Commitment slider to the corresponding range, and setting the Number of Change slider to its minimum and maximum value.

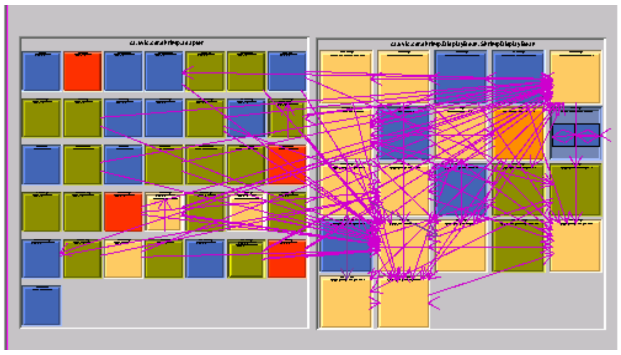


Figure 4(a). The color of each node represents the author who made the latest change

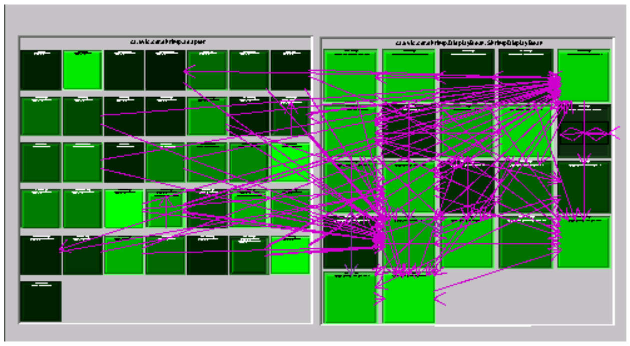
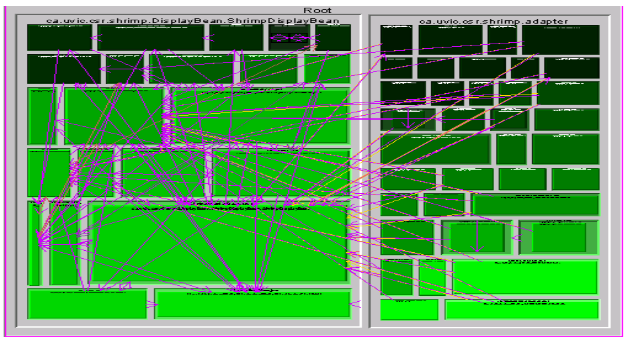


Figure 4(b). The color intensity of each node is determined by the date of the latest commitment.

2.2.4. Ordered Treemap layout. To provide a view that addresses the 5W+2H problem at the entire project level, we adopted the Ordered Treemap algorithm created by the HCI lab at the University of Maryland [12]. Two distinct features of the Treemap layout are the variation of the node size according to the associated numerical attribute and the repositioning of nodes according to their associated ordinal value. Figure 5 demonstrates a screen shot in which node size was adjusted according to the


Root
ca.uvic.csr.shrimp.DisplayBean.ShrimpDisplayBean
ca.uvic.csr.shrimp.adapter
Figure 5. An ordered Treemap layout.

number of changes and the position of nodes are ordered by their last commit date. This ordering feature provides a comparable view for files in a project, hence answering the *When* question at the project level.

2.2.5. Summary of visualization features for CVS. Xia provides various ways to visualize data or derived data from the CVS repository. In addition, relationships between files can be determined using information extracted from the Eclipse JDT plug-in.

Table 2. Map of visualization techniques to questions of interest when working with CVS

Question	Visualization techniques
What	The name of the changed file can be shown using labels on the nodes (which are visible when you zoom in), or they can be shown using tool tips when the user brushes over nodes in the graph with a mouse.
Who	Can be distinguished using different node colors; filter by name using a checkbox. Tool tips could also be used to show the author's name.
Where	Nested within relevant folders in the layouts
When	Date can be shown using color intensity, tool tips. File revisions can also be filtered by date
Why	Rapid access to the code, CVS comments (in the attribute table) and documentation (by right clicking on the nodes, or by zooming in to an embedded view)
How	Access to the code, Javadoc and CVS comments by zooming on a file revision node. Tool tips, intensity, size and location could also be used to show number of changes to a file. Relationships between files are shown using arcs, which could be used to trace the impact of changes.
History	Access to a history panel by zooming on a file revision node and the use of right mouse menu. A table contains the revision history of the file is displayed.

We conjecture that the 5W+2H questions can be answered by interacting with the features in Xia which includes the double slider filters, the different layout algorithms (such as the Treemap layout) effecting size and order of the nodes, the checkbox filters, tool tips, color and intensity. In addition to these features, Xia provides easy access to the source code, documentation (Javadoc) and comments in the CVS repository. The user can zoom into a node representing a file revision and switch between these different views. In Table 2 we summarize how these different features can be used to answer the 5W+2H questions.

3. Evaluation

We conducted an exploratory user study to test both the initial requirements we discovered through the survey and the functionality and usability of Xia. As very few studies have been conducted in the field of version control visualization [19], we consider this study novel.

A Java project with four versions was chosen as the dataset for the study. Five graduate students from the Department of Computer Science at the University of Victoria participated in the study. Each participant had experience on a team software project, working with at least one version control tool.

Following the pre-study questionnaire (to determine their previous programming and version control experience etc.), a fifteen-minute orientation on Xia was provided to introduce the basic tool operations and the tool's core features to each participant. Then, a task list was administered to participants. Further inquiry into the user's opinion of the tool was gathered through a post-study questionnaire.

The following subsections describe in detail the tasks users performed and our general observations.

3.1. Tasks

Two sets of similar tasks were assigned to participants corresponding to two different data resources: the data in the CVS repository and the data in the programmer's own workspace. These two sets of data constitute a programmer's working data in the real world.

The tasks involved exploring the information space and answering questions related to team work and software history, including the 5W+2H questions. For example, one of the tasks asked the participant to name all programmers that have been working on the project. Another task asked the participant to find out who was the last person working on a particular file. These two tasks correspond to the "Who" question on the project and file levels. In regards to the "What" question, we asked the user to determine what kind of changes to a particular file have been made. As per the "When" question, we encouraged the user to establish which file was changed most recently. With respect to the "How" question, participants were asked to discover how a particular file was changed in the latest commitment. The "Why" question was explored by asking for the rationale behind a particular change, and the "History" of a file was explored by request too.

3.2. General observations

Participants successfully resolved most tasks. Some general observations were as follows:

- The visualization and exploration techniques provided by the **Attribute Panel** were used frequently to resolve the tasks. Also, participants pointed out in their post-study questionnaires that they would like to use features of the **Attribute Panel** in their everyday work.
- The tool appeared to be easy to learn and use. Although only fifteen minutes of orientation was provided, participants used the tool effectively to perform the tasks. They were aware of the possible ways to use the tool to solve problems and did not require additional assistance or note any significant difficulties.
- Participants considered the tool informative, from both the project manager and programmer perspective. Candidates indicated they believe the Xia tool could prove helpful in helping them solve many problems they encounter in a work environment.
- The tool was also used to answer more sophisticated questions by making use of a combination of features. For example, one of the tasks asked the participants to find out which file is most stable and which file is most active. The participants defined "stable" and "active" in a similar way: a file that has not been changed for a long time and to which very few changes were made was considered stable; the opposite held true for an active file. To answer this question, participants chose both the *last commit date* double slider and *number of change* double slider to narrow down the range of candidate nodes, and analyzed the candidate nodes.
- The visualization features in Xia helped the users gain more awareness of their teammates activities.
- The participants were impressed by the immediate feedback the visualizations provided when they posed a new query. Some of them had special interests in color schemes while others used filters more often.

Though the positive feedback is encouraging, we also noticed some deficiencies of the tool:

- Some participants were confused when working with different revisions of the same file. They suggested that some kind of mapping between different revisions of the same file would be helpful.
- The file revision organization requires a more elegant display. Currently, the file revisions are organized by software versions. However, revisions not belonging to a particular version will not be considered or displayed in the tool. This may lead to the loss of information.
- Some participants also suggested a time-line arrangement of project versions as time is a very important attribute in version control.
- Visualization of other attributes was also anticipated by some of the users. For example, one of the users was interested in who originally created a particular file.
- Participants considered the "diff" function – a comparison of two different file revisions very important in their everyday work. We considered displaying the CVS plug-in's diff view within Xia, however, Xia does not currently support this feature on account of difficulties embedding Xia's Java Swing [15] GUI inside of Eclipse's SWT [16] GUI (a problem discussed by Rayside *et al.* [10]). Further investigation is required for this technical issue.

3.3. Justification of the study design

The study we conducted is a very preliminary step to provide feedback on the use of a tool such as Xia and to help us in our requirements gathering process. Although the number of users was small, we were more interested in finding out if the requirements we had were correct, and if we were missing any. The preliminary study also allowed us to study how the tool can be used to help with version control activities. Our intention at this point is not to perform statistical analysis of results, as we believe research in this area is still too new. The size of the code studied was also small but the study required that the users gain some knowledge of the code in a relatively short time. We also did not compare our tool to others as firstly, there are no other prototypes available that use visualization for version control activities. Secondly, we considered comparing our tool to CVS, but this would be a biased study as our tool provides answers to questions which cannot be easily answered by CVS even in a textual way.

4. Future Work

Based on our observations from the user study we found more research questions that need to be explored. For example, we believe more version attributes beyond the 5W+2H questions (e.g. the creator of an artifact) and visualization of finer granularity of version control should be investigated. We are currently making improvements to our tool – the question we now face, is how should we proceed in the next evaluation phase? Our proposed approach is to do an introspective case study by applying the tool in our own research group's programming activities. We also believe the requirements are evolving with the availability of various tools. We are interested in feedback on our tool and on our requirements at the Vissoft workshop.

References

[1] Ball, T. A. and Eick, S. G. 1996. Software visualization in the large. *IEEE Computer*, vol. 29, no. 4, pp. 33-43.

[2] Card, S. K., Mackinlay, J. D., and Shneiderman, B. 1999. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann.

[3] CVS 2003. The CVS website: http://www.cvshome.org/

[4] Dourish, P. 2002. "Visualizing Software Development Activity". URL: http://www.ics.uci.edu/~jpd/research/seesoft.html

[5] Eclipse 2003. Eclipse Homepage: http://www.Eclipse.org

[6] Eclipse Platform, 2003. The Eclipse Platform Subproject Webpage: http://www.eclipse.org/platform/index.html

[7] Eick, S. G., Steffen, J. L., and Summer, E. E. 1992. Seesoft – A tool for visualizing line oriented software statistics. *IEEE Trans. Software Engineering*, vol. 18, no. 11, pp. 957-968.

[8] Lintern, R., Michaud, J., Storey, M.-A., and Wu, X. 2003. Plugging-in Visualization: Experiences Integrating a Visualization Tool with Eclipse. In *Proceedings of Software Visualization 2003.*

[9] McGuire, K. 2002. VCM 2.0 Story (article in Eclipse website: http://www.eclipse.org/platform/index.htm)

[10] Rayside, D., Litoiu, M., Storey, M.-A., Best, C. and Lintern, R. 2002. Visualizing Flow Diagrams in Websphere Studio Using SHriMP Views (Visualizing Flow Diagrams). *Information Systems Frontiers: A Journal of Research and Innovation,* vol. 4 (4)

[11] Shneiderman, B. 1992. Tree Visualization with Tree-maps: A 2-d space-filling approach. *ACM Trans. Graphics*, vol. 11, no. 1, pp. 92-99.

[12] Shneiderman, B. and Wattenberg, M. 2001. Ordered Treemap Layouts. In *Proc. IEEE Symposium on Information Visualization 2001*, 73-78. Los Alamitos, CA

[13] Storey, M.-A., Best, C., Michaud, J., Rayside, D., Litoiu, M. and Musen, M. 2002. SHriMP views: an interactive environment for information visualization and navigation. In *Proceedings of CHI 2002 Conference,* Minneapolis, Minnesota, USA, pp. 520-521.

[14] SHriMP 2003. SHriMP Website: www.shrimpviews.com

[15] Swing 2003. The Swing Connection, http://java.sun.com /products/jfc/tsc/

[16] SWT 2003. SWT: The Standard Widget Toolkit, http://www.Eclipse.org/articles/Article-SWT-Design-1/SWT-Design-1.html

[17] Tu, Qiang and Godfrey, Michael 2002. An Integrated Approach for Studying Software Architectural Evolution. In *Proc. of 2002 Intl. Workshop on Program Comprehension (IWPC-02)*.

[18] Ware, C. 2000. *Information Visualization, perception for design.* Morgan Kaufmann

[19] Weinberg, Z. 2002. Novel Methods of Displaying Source History: A Preliminary User Study. http://www.panix.com/~zackw/cs260/novel-methods-of-displaying-source-history.pdf