A Source Code History Navigator

Alexander Bradley (awjb@cs.ubc.ca)

CPSC 533C Project Proposal
University of British Columbia

October 30, 2009

1 Problem Description

This project will address a problem drawn from the domain of software engineering: visualizing the history of source code. To gain understanding of the decisions and developmental steps that led to the current state of a piece of source code, a developer may need to consult several previous revisions of the code using a revision control system such as CVS. Current integrated development environments (IDEs) such as Eclipse make it easy to compare two revisions of a source code file. However, the developer may need to compare several past revisions to gain the desired understanding of the code's evolution. Many-revision comparison is not, to my knowledge, well supported by current IDEs. Research in software visualization systems has explored some techniques for displaying software evolution, and I hope to build on these existing approaches.

Any codebase with a significant revision history (and probably, though not necessarily, multiple contributors) is a potential dataset for this project. A particular dataset I may use is the open-source JQuery project from the UBC Software Practices Lab, with which I have some past experience as a developer.

2 Personal Expertise

I have 20 months of employment experience (16 co-op, 4 as an undergraduate RA) in software development positions. I am familiar with the Eclipse IDE and revision control systems such as CVS, Subversion and Perforce. The problem addressed in this project is one I personally encountered in some of my development work.

I have some experience developing GUIs using the Swing toolkit for Java. I have little experience developing GUIs directly with SWT, although I have some experience using SWT indirectly during Eclipse plugin development.

3 Proposed Solution

For my proposed solution to this problem, I am considering some variations on a small multiples approach. In its simplest form, this approach would display several revisions of the same file side by side (see Figure 1) or in a grid (Figure 2). The available screen real estate on current monitors would probably make it difficult to display more than three legible code panes horizontally and two panes vertically, but scroll bars could be used to permit access to a larger virtual space of code panes. (I envision the side-by-side display scrolling horizontally and the grid display scrolling vertically.) Each code pane could incorporate additional visual and textual information of use to the developer, e.g. marginal hints regarding the locations of differences, revision notes, revision numbers and authors, or source code colouring showing code age, statement type or authorship.

I am also considering a more complex focus+context visualization (see Figure 3) which incorporates elements of the TableLens of Rao and Card [2] and the "evolution view" from the Visual Code Navigator (VCN) of Lommerse et al. [1]. As in VCN, file revisions would be displayed as a "flow" of vertical stripes of pixels, with each horizontal pixel line in the stripe representing one or more lines of code from the revision. The horizontal pixel lines could be coloured according to code age, code authorship, etc. (see Figure 8 in [1] for examples.) At a few points, code panes showing the actual text of code from particular revisions could be embedded in the flow immediately to the right of the pixel stripes for those revisions. A black rectangle to the left of a code pane

could indicate where the code shown in the pane was located in the pixel-stripe representation of that revision. The lines of code in the pane could have a similar background colour scheme to the pixel stripe, to further emphasize the connection between the panes and the flow. The user would be able to move the panes to focus on different revisions, perhaps by dragging them or selecting revision stripes with the mouse.

The three visualizations described above would probably be useful in different and complementary ways. The focus+context view could be used to assess the overall patterns of evolution in the code and pick revisions of interest. The two small multiples views could be used for more detailed code comparison between selected revisions of interest.

Ideally, this system would be integrated with an IDE (e.g., as an Eclipse plugin), but this will not be considered an essential aspect of this course project. I will investigate Eclipse integration if I have extra time.

This tool will tentatively be called the *Source Code History Navigator* (SCHN).

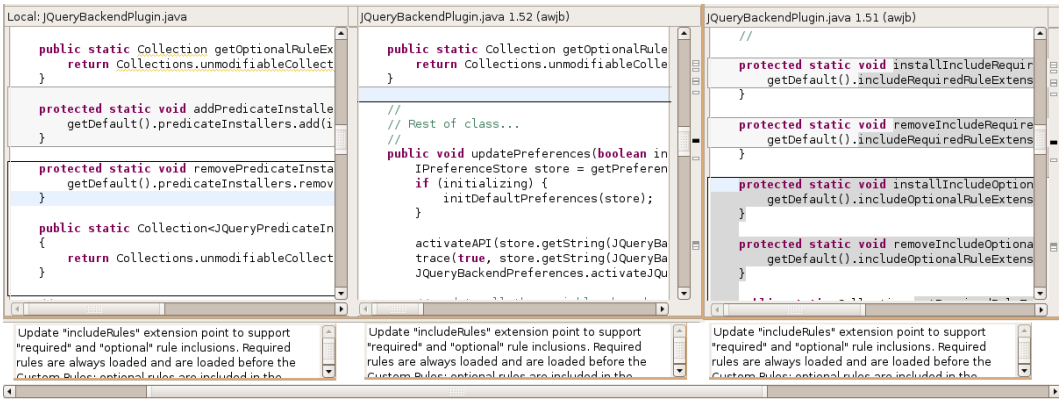

Figure 1: Side-by-side small multiples display (mockup).

4 Usage Scenario

A developer has discovered buggy code in source file `AnalyticalEngine.java` and wants to explore the file's history. The developer opens the file in SCHN. Initially, the focus+context view of the revision history is displayed with some arbitrarily selected code panes. The developer selects colouring by authorship and moves the code panes to points where significant changes seem to have occurred. The developer selects four interesting revisions and displays them in the grid view. After detailed comparison of the revisions, the developer might conclude that the code causing a buggy behaviour was introduced by user `menabrea` in revision 31 and significantly modified by users `alovelace` in revision 45 and `cbabbage` in revision 60. The developer might then contact those users to discuss the impact of a possible bug fix.

5 Implementation Approach

I plan to implement this system in Java using either Swing or Eclipse SWT. I am more familiar with using Swing for GUI development. However, using SWT would make subsequent integration

Figure 2: Grid small multiples display (mockup).

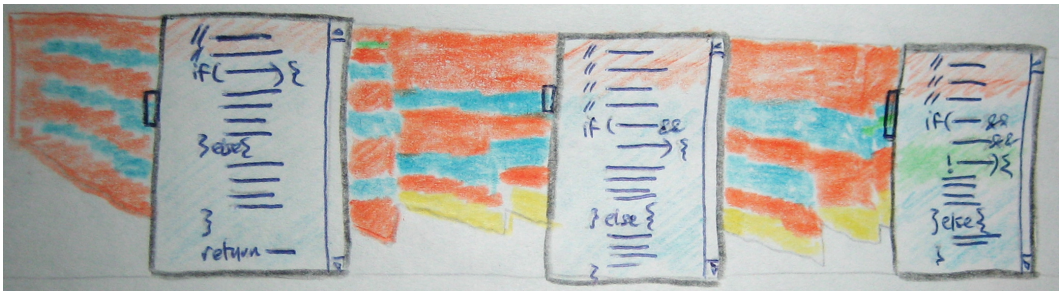

Figure 3: Historical evolution view with embedded code panes (sketch).

of my system as an Eclipse plugin easier, and might also permit me to reuse source code display components from Eclipse. My system will probably be designed to interact with the CVS revision control system.

6 Schedule

A tentative schedule for this project is as follows:

Friday, November 6:	Development environment set up. Prototype should be capable of accessing CVS repository and downloading code. Small-multiples prototype started.
Monday, November 16:	Prototype of small-multiples views complete. Prototype of focus+context view started.
Monday/Wednesday, November 16/18:	Project update presentation.
Friday, November 27:	Prototypes of all views complete.
Thursday, December 10:	Final implementation complete. Final presentation and report drafted.
Monday, December 14:	Final project presentations.
Wednesday, December 16:	Final report submitted.

7 Related Work

The code evolution "flow" visualization used in this project has its roots in a number of prior systems. Early work by Ball and Eick [3] explored the idea of representing large amounts of code by condensing source code lines to single-pixel horizontal line representations. The "flow" approach considered above was inspired by the "history flows" created by Viégas et al. [3] to visualize Wikipedia edit history. Voinea [4, 5, 6] has explored the domain of software evolution visualization extensively and has produced a variety of visualization techniques for exploring the history of source code files and entire codebases. The Visual Code Navigator [1] mentioned earlier incorporates his work. Voinea's CVSscan tool [5] includes a pixel-stripe revision visualization similar to "history flows", but has more sophisticated filtering techniques to accommodate the potentially large size of code files. CVSscan also includes a "details-on-demand" feature which allows the user to select a single revision from the flow and view the associated source code.

References

[1] G. Lommerse, F. Nossin, L. Voinea, and A. Telea. The visual code navigator: An interactive toolset for source code investigation. In *INFOVIS '05: Proceedings of the 2005 IEEE Symposium on Information Visualization*, page 4, Washington, DC, USA, 2005. IEEE Computer Society.

[2] R. Rao and S. K. Card. The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 318–322, New York, NY, USA, 1994. ACM.

[3] F. B. Viégas, M. Wattenberg, and K. Dave. Studying cooperation and conflict between authors with history flow visualizations. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 575–582, New York, NY, USA, 2004. ACM.

[4] L. Voinea and A. Telea. Visual querying and analysis of large software repositories. *Empirical Software Engineering*, 14(3):316–340, 2009.

[5] L. Voinea, A. Telea, and J. J. van Wijk. CVSscan: visualization of code evolution. In *SoftVis '05: Proceedings of the 2005 ACM symposium on Software visualization*, pages 47–56, New York, NY, USA, 2005. ACM.

[6] S. L. Voinea. *Software Evolution Visualization*. PhD thesis, Technische Universiteit Eindhoven, 2007.