RepoVis: A Tool for Assisting Project Management by Visualizing Source Code Versioning Systems

Matthew Lueder
Grand Valley State University
CIS 671 – Information Visualization

Abstract

Version control systems (VCSs) are vital to project management because they contain important information related to the history of a software engineering team's progress. VCSs alone are limited in the ways they can present this information to the user, especially when it comes to allowing users to discern patterns at higher levels of abstraction. We developed RepoVis to assist project management by providing a visualization based approach to the exploration of VCS metadata. RepoVis uses a dual-layer graph which shows relationships between developers and projects. Double-clicking nodes in this graph shows users more detailed information on developers and projects. Users can filter out nodes that are not of interest.

Introduction

It is estimated that software engineers spend 70% of their time on cooperative activities [1]. Collaboration is essential for the software design process. Version control systems (VCSs) have become an essential part of collaboration within software development teams. VCSs assist with the task of keeping software systems consisting of many versions, configurations, and contributors well organized [2]. VCSs allow developers to merge changes they have made to a local version of a file with a repository. The action of doing so is often referred to as a 'commit'. Other developers can then see the changes made and 'pull' the updated project to their local machines. VCSs store information about changes made to a repository including information about commits made (what was changed, who changed it, etc.).

Because of the wealth of information that these systems provide, VCSs have become vital to project management. For example, a project manager could use information obtained by a VCS to: assign new tasks to the relevant developers, to find out who is responsible for a bug, and to monitor the productivity of developers [3]. However, there are limitations to using VCSs as a project management tool. VCSs often contain a very large amount of information. Finding information relevant to the project manager's need can be difficult because most VCSs do not provide tools to easily filter out irrelevant information. Also, VCSs are typically good with presenting low level information such as what lines of code have been changed with a commit or on what date a commit took place, but are not effective at outlining higher level structure.

A well-developed visualization is capable of effectively showing higher level structure of large data sets while simultaneously allowing for users to filter out irrelevant information and drill down on points of interest. Because of this, we decided to create a tool which visualizes information parsed from a VCS.

ReposVis is a tool meant to assist project management by providing a visualization based approach to the exploration of source code versioning system metadata. RepoVis uses information parsed from a VCS and provides a variety of views. The main view is an overview of all projects in the repository and uses a connected graph to represent relationships between developers and projects. At the side of this view are controls which allow the user to filter the information displayed in this view. Details on demand are provided when a user double-clicks on a node in the graph view. If the user double-clicks on a node representing a developer, a dialog containing the developer view appears. The developer view shows a list of the projects a developer has worked on. If a user double-clicks on a node representing a project or project cluster, a dialog containing the project cluster view appears. The project cluster view shows the relative levels that each developer contributed to the projects in the cluster.

Related Works

Tools to visualize source code revision history have existed for over two decades [4]. Seesoft, created in 1992, was one of the first projects to attempt VCS visualization. Seesoft represents source code through pixel and file maps. It can display up to 50,000 lines of code simultaneously and uses color to visualize features of interest. For example, it uses the color red to indicate that a line of code has been recently changed, and blue to indicate a line of code has not been changed recently [5]. Seesoft is a good tool for seeing patterns in a small project but is limited in the fact that it is only able to display 50,000 lines of code at a time. In most situations, a repository will contain a code base many magnitudes larger in size.

Code_swarm and Gource are two more recent VCS visualizations that attempt to show the entire history of a project using an animation. In Gource, the visualization starts with a single node which represents the root directory of the project. As time progresses, a tree structure is built, with new nodes representing files added to the root directory [6]. In code_swarm, the names of developers are shown on screen and files are represented as orbs which circle the name of the developer working on them. The color of the orb represents the document type and the darkness of the color represents how recent a file was committed [7]. Both code_swarm and Gource provide visually appealing animations, however taking any useful information out of these visualizations is difficult. Because of this, these projects do not make a good project management tool.

The use of graphs is common among existing VCS visualization software. This is because graphs can reveal the hierarchical evolution of software. Xia/Creole shows differences between artifacts of two versions using a hierarchical graph view [8].

Extracting Data from the CVS

RepoVis was created to work with any version control system but is most easily compatible with Apache Subversion (SVN). This is because it comes with a script to automatically parse log files created with the ‘svn log’ command into a CSV file which is the

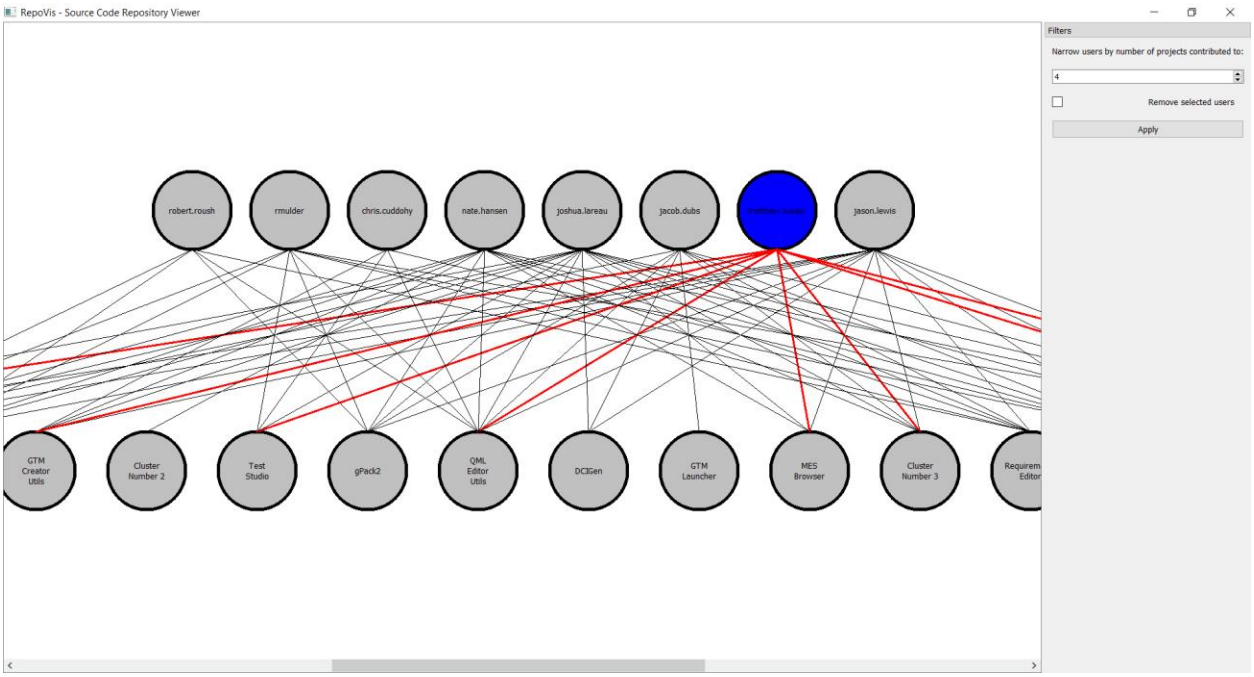

Figure 1. This figure shows the graph view, which acts as a main overview for RepoVis. User nodes can be seen aligned horizontally across the top, and project cluster nodes can be seen aligned at the bottom. At the right is the filtering pane. Currently, all users who have contributed to less than four projects have been filtered out. The user node 'matthew.lueder' has been selected resulting in all connected edges being highlighted.

main input of the program. The CSV used as input has three columns: project name, employee username, and timestamp. Each row in the CSV represents a commit.

Graph View

The Graph view is the main view of RepoVis. This view uses a connected graph to show relationships between developers and projects within the repository. The graph consists of two rows of nodes. The top row contains developer nodes, while the bottom row contains project cluster nodes (Figure 1).

Each developer node represents a developer who has made changes to at least one project in the repository. In the center of each developer node, the represented developer's username is displayed. Project cluster nodes represent project clusters. A project cluster is a collection of projects which have been worked on by the same developers. A project cluster may contain only one project if this project has a unique combination of developers who have worked on it. Project clusters were created to help reduce visual clutter. Project cluster nodes that represent project clusters which contain more than one project are numbered, and this number is shown in the center of the node. Project cluster nodes that represent project clusters containing only one project, have the name of this project in the center of the node.

Project cluster nodes and developer nodes are connected using undirected edges. A connection between a developer and a project cluster means that the developer has worked on all the projects in the project cluster.

Left-clicking on a node highlights the node and all edges connected to it. Left-clicking on the node again will un-highlight the node and all of its edges. This feature allows users to

easily see what nodes are connected by edges to a specific node of interest.

Filtering

On the right side of the graph view there is a pane with controls to allow the user to filter out the information presented in the graph view (Figure 1). Currently, there are two ways that a user can filter information.

The first way allows users to filter out users who have not contributed to some minimum number of projects. This minimum number is selected in the combo box at the top of the filtering pane.

The second way is to remove all highlighted users. To do this, you must mark the checkbox in the filtering pane that is next to the label saying "Remove selected users".

In order to initiate filtering, the user must click the apply button at the bottom of the filtering pane. To unset all filters, set the combo box to 1, uncheck the remove selected users checkbox, and press apply.

Developer View

The developer view appears in a dialog after the user double-clicks on a developer node. This view consists of a table which shows the name of each project that the user has worked on in the first column, and the date of the last commit the user had made to that project in the second column. Projects are sorted by the second column, so the project that the user has worked on most recently will appear at the top of the table (Figure 2).

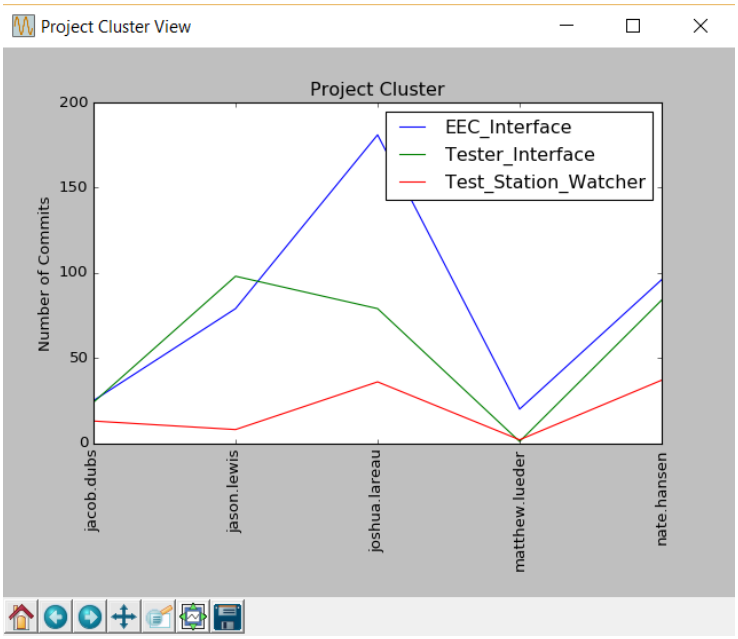

Figure 2. Project cluster view. A dialog showing a showing a project cluster containing 3 projects which have been worked on by 5 developers.

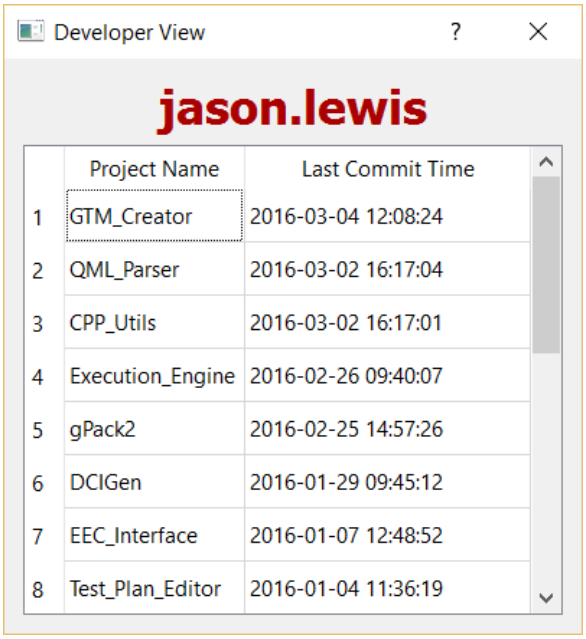

Figure 3. Developer view. At the top of this dialog is a label showing the developer's name in red. Below this is a table containing the names of all the projects that this developer has worked on and the last time this developer made changes to it.

Project Cluster View

The project cluster view appears in a dialog when a user double-clicks on a project cluster node. This view uses a line graph to show relative levels that each developer contributed to the projects within the project cluster. Users that have contributed to the projects in the cluster are listed across the x-axis. The y-axis represents number of commits. Each project in the cluster is represented by a line in the graph. When there is more than one project being represented, a different color is used for each line (Figure 3).

Implementation

RepoVis was created using Python 3.4.4 for Windows. PyQt5, a python port of the Qt 5 library, was used to create the graph view, filtering pane, and the developer view. Matplotlib 1.5.1 was used to create the project cluster view.

Future Goals

RepoVis has many useful features to allow project managers to explore VCS information. However, there are a number of improvements that could be made to it in order to increase user performance and functionality.

Currently, nodes in the graph view are not placed in any specific order. If nodes were placed in an order which would reduce edge-crossing, visual clutter would be reduced and user performance could be improved.

Also, the current tools given to allow for filtering are limited. There is no way to filter the project cluster nodes. This could be a problem when visualizing repositories with a large number of projects. A way to filter project cluster nodes could improve performance when visualizing these types of repositories.

References

1. I. Vessey and A. P. Sravanapudi, "CASE tools as collaborative support technologies", Commun. ACM, vol. 38 (1995): 83-95
2. Tichy, Walter F. "Rcs — a System for Version Control." Softw: Pract. Exper. Software: Practice and Experience 15.7 (1985): 637-54
3. Xie, Xinrong, Denys Poshyvanyk, and Andrian Marcus. "Visualization of CVS Repository Information." 2006 13th Working Conference on Reverse Engineering (2006)
4. Liu, Chang, Xin Ye, and En Ye. "Source Code Revision History Visualization Tools: Do They Work and What Would It Take to Put Them to Work?" IEEE Access 2 (2014): 404-26
5. Eick, S.c., J.l. Steffen, and E.e. Sumner. "Seesoft-a Tool for Visualizing Line Oriented Software Statistics." IEEE Transactions on Software Engineering IIEEE Trans. Software Eng. 18.11 (1992): 957-68
6. Caudwell, Andrew H. "Gource." Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion - SPLASH '10 (2010)
7. Ogawa, Michael, and Kwan-Liu Ma. "Code_swarm: A Design Study in Organic Software Visualization." IEEE Transactions on Visualization and Computer Graphics 15.6 (2009): 1097-104.
8. Wu, Xiaomin, A. Murray, M.-A. Storey, and R. Lintern. "A Reverse Engineering Approach to Support Software Maintenance: Version Control Knowledge Extraction." 11th Working Conference on Reverse Engineering (2004): 80-89.