ClonEvol: Visualizing Software Evolution with Code Clones

Avdo Hanjalić
Department of Computing Science
University of Groningen, the Netherlands
E-mail: a.hanjalic@student.rug.nl

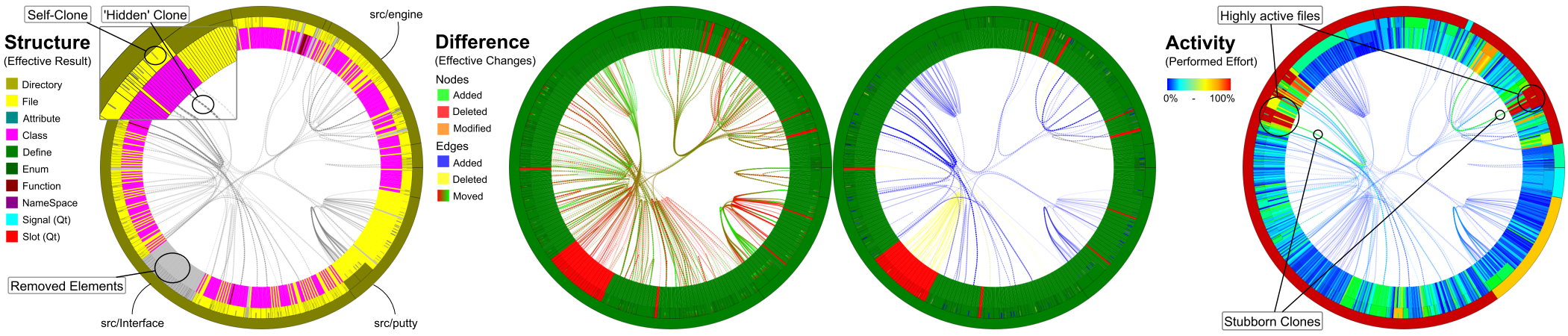

Figure 1: **Structure**, **Difference** and **Activity** color-maps shown for FileZilla trunk/src revision 0,001 to 5,165. Structure shows the state of the project at revision 5,165; Difference flattens all changes from revision 0,001 to 5,165 to show them as one change-set; Activity indicates how often a file/scope was involved in a change-set (relative to total amount of change-sets).

Abstract

ClonEvol is a visual analysis tool that assists in obtaining insight into the state and the evolution of a C/C++/Java source code base on project, file and scope level. It combines information obtained from the software versioning system and contents of files that change between versions; The tool operates as tool-chain of Subversion (SVN), Doxygen (applied as static analyzer) and Simian as code duplication detector. The focus lies on scalability (in time and space) concerning data acquisition, data processing and visualization, and ease of use. The visualization is approached by using a (mirrored) radial tree to show the file and scope structures, complemented with hierarchically bundled edges that show clone relations.

Mining changes in large projects

Revision control systems offer a vast amount of information that can help to understand (the evolution of) a source code base. The amount, location and span of code clones are a reliable measure when assessing quality of (the source code of) a software project. More interestingly, changes in code clones reveal the dynamics of a code repository, allowing us to obtain insight into the evolution of a project. We exploit these ideas in ClonEvol, a tool for analyzing code changes large software repositories using clone change data. The focus of our tool lies on

- **Scalability** (in time and space) for data acquisition, processing, and visualization;
- **Genericity** in terms of supported programming languages;
- **Simplicity** of use: point the tool to a repository URL and press 'Go';
- Utilization of third-party **open source** components.

Scalability is achieved by limiting data acquisition and fact extraction to only differences between code base versions. Where other tools take days or even weeks to mine a real-life project, ClonEvol can process hundreds of revisions per hour. ClonEvol supports all languages that are supported by its components (Doxygen and Simian), including C, C++ and Java.

Availability

A runnable version of ClonEvol can be found at: http://www.cs.rug.nl/svcg/SoftVis/ClonEvol. ClonEvol is subject of the author's MSc. thesis, that details the architecture and several usage scenarios. The thesis and source-code will be made available November 2013.

Tool-chain approach

Subversion change-logs, static analysis and code clone detection are combined to obtain information about clone evolution.

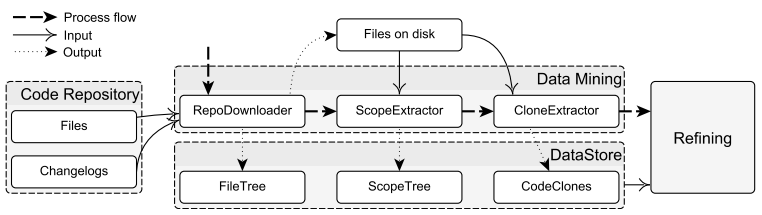

Figure 2: Data mining

1. Revisions are mined from a Subversion repository;
2. Code structure is extracted using lightweight static analysis provided by Doxygen;
3. Clones are searched between modified files in one revision and all files in the previous revision, using Simian;
4. Clones are lifted to structural level;

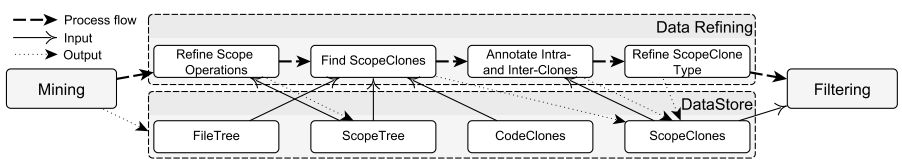

Figure 3: Data refinement

5. The following clone-related events are detected:

- **Addition**: software is modified or added, thereby yielding a new clone;
- **Deletion**: a software fragment is removed, thereby removing a clone;
- **Drift**: software code is moved around, thereby causing a clone to drift;
- **Merge**: two software fragments are moved to the same place;
- **Split**: a software fragment is cut into two parts which are next moved in different places.

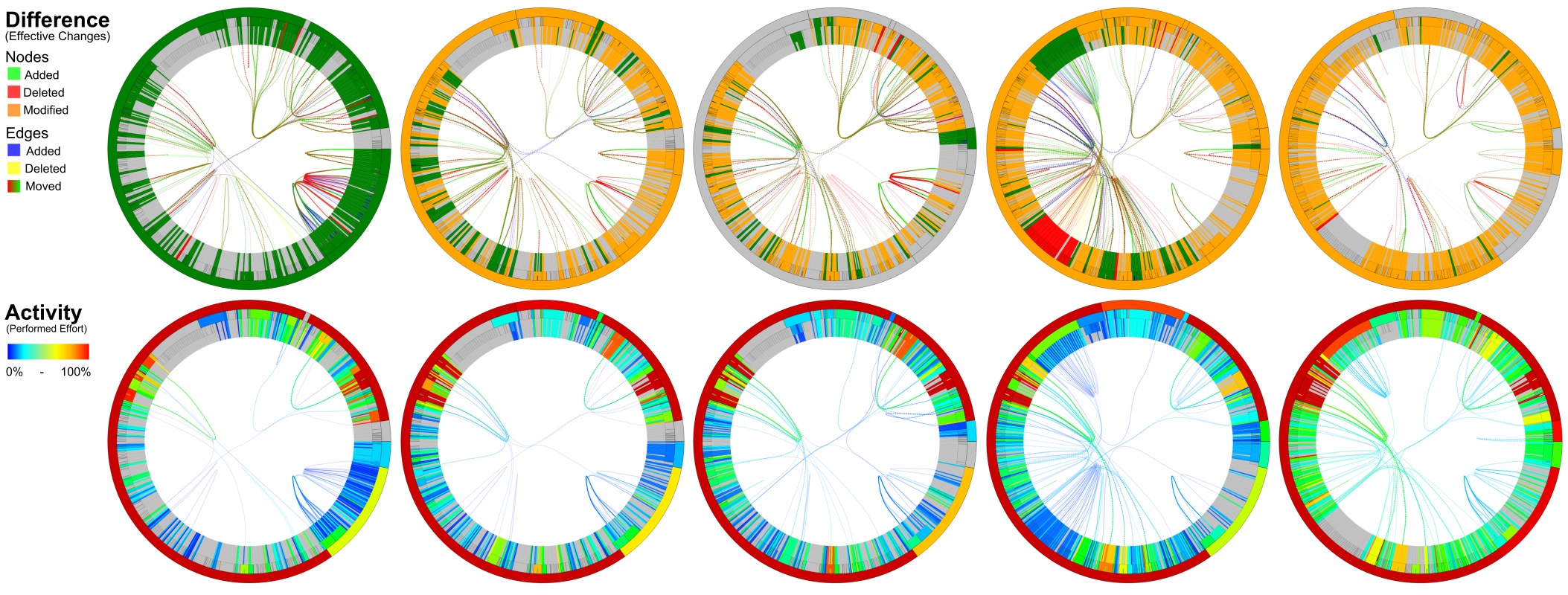

Figure 4: Evolution of FileZilla trunk/src from revision 1 to 5,165.