Large-Scale Reuse in Open Source Software

Audris Mockus

audris@avaya.com

Avaya Labs Research
Basking Ridge, NJ 07920
http://mockus.org/

Open Source Innovations

- Fundamentally different model of software development
  - Built by large numbers of volunteers without physical contact
  - Work is not assigned but chosen
  - Design controlled by a few architects
- Resulting properties of software and process [2]
  - Small core team controlling code submission and new features with an order of magnitute wider bug fix community and two orders of magnitude larger problem reporting community
  - Low post-feature-test defect density
  - Large developer productivity
  - Rapid response to user problems

Research Goals

- A key premise of open source is that the code can be used in other projects
  - Reduces risks of project's code being no longer available or supported
  - Provides social value by encouraging innovation (no need to reimplement existing functionality)
- These suggest the following research questions:
  - What is the extent of reuse?
  - What are properties of highly reused code?
  - How to evaluate reuse potential for a component?
  - How to to find code most suitable for reuse?
  - How to produce code that is more likely to be reused?

Experimental approach

- Sample a large set of open source projects
- Identify and quantify instances of large-scale reuse
  - not a copy and paste in an editor
  - not a case of reuse where another project is reused as-is through libraries without copying the code
- Identify common patterns of reuse
- Quantify quality and other properties of the reused code

Sample selection and retrieval

- Sample
  - Important projects: Apache, Gnome, KDE, Mozilla, OpenSolaris, Postgres, and W3C
  - Large distributions: Fedora 6, Gentoo, Slackware, FreeBSD, NetBSD, and OpenBSD
  - Development portals: Savannah, SourceForge, and Tigris
  - Random or language specific: FreshMeat, CPAN, RpmForge, and Gallery of Free Software Packages
- Retrieval
  - SVN/CVS, wget, and page scraping (FreshMeat)
  - $13.2M$ files from $49.9K$ bundles
  - $5.3M$ source code files and $38.7K$ bundles after normalization (removing package versions, binary files, ...)

Quantify large-scale reuse

- Method
  - Identify pairs of directories with a large fraction of filenames that are shared between them [1] as reused directories
  - Consider files with the same names in reused directories to be reused
- Measures
  - Overall reuse — a fraction of files that are in more than one project
  - Component reuse — a number of projects in which the component is present

Results

- Results using different parameter values for the minimal fraction of shared filenames between two directories

	(30%)	(50%)	(80%)
File count	$2,837,233$	$2,782,339$	$2,654,977$
Overall reuse	.53	.52	.49

Table 1: Reused files in open source projects.

Scenarios of reuse

- Most reused (numbers are based on 80% cutoff)
  - Text template: 657 projects using language translations, “po” directory with almost 50 files: “am.po”, ..., “zh_TW.po”
  - Functional template: 576 projects using install module for Perl
  - Verbatim copy: 547 projects using C functions for internationalization
- Largest components reused at least 50 times
  - 701 include files for Linux kernel
  - System dependent configuration: glibc/sysdeps/generic with 750 files

Validity

- Sampling process to increase the representativeness of project sample
- The definition of large-scale reuse
  - not a copy and paste in an editor
  - not a case of reuse where another project is reused as-is through libraries without copying the code
- No substantial changes to filenames or directory structure
- The instances of reuse are underestimated (no cases of mistaken identification of reuse were found)

Summary and future work

- Findings
  - The three most common patterns of reuse do not suggest immediate ways to increase reuse but point out less intuitive avenues for reuse
  - The reuse is, indeed, massive and, therefore, has to facilitate innovation and to ensure that reused code lives on even if some projects die or vegetate
  - The amount of OSS code is not that vast
- Future
  - Better sample, identification of reuse, classification of patterns
  - Reconstructing authorship and implicit collaborations via universal version history
  - Quantifying quality and other properties of highly reused code
  - Quantifying benefits to society

References

[1] Hung-Fu Chang and Audris Mockus. Constructing universal version history. In *ICSE'06 Workshop on Mining Software Repositories*, pages 76–79, Shanghai, China, May 22-23 2006.

[2] Audris Mockus, Roy T. Fielding, and James Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):1–38, July 2002.