Census of public source code history

Audris Mockus

[2016-01-21]

Outline

- Background
- Motivation: why census?
- How to get it done?
    - Steps: Discover, Retrieve, Store, Update
    - Resources: Network, Computing, Storage
    - Qualitative: Interviews, Surveys :noexport:
- Example applications
- Techniques critical for operational data
- Need research

Version Control Data

- Developers use VCS to track changes

Operational Data from VCS

Code Before

```
int i = n;
while (i−−)
   printf (” %d”, i);
```

Code After

```
//print n integers iff n≥ 0
int i = n;
while (−−i > 0)
   printf (” %d”, i);
```

Version Control Data

- Developers use VCS to track changes

Operational Data from VCS

Code Before

```

int i = n;
while (i--)
   printf (" %d", i);
```

Code After

```
//print n integers iff n≥ 0
int i = n;
while (--i > 0)
   printf (" %d", i);
```

one line deleted

Version Control Data

- Developers use VCS to track changes

Operational Data from VCS

Code Before

```

int i = n;
while (i−−)
   printf (” %d”, i);
```

Code After

```
//print n integers iff n≥ 0
int i = n;
while (−−i > 0)
   printf (” %d”, i);
```

two lines added

Version Control Data

- Developers use VCS to track changes

Operational Data from VCS

Code Before

```

int i = n;
while (i−−)
   printf (” %d”, i);
```

Code After

```
//print n integers iff n≥ 0
int i = n;
while (−−i > 0)
   printf (” %d”, i);
```

two lines
unchanged

Version Control Data

- Developers use VCS to track changes

Operational Data from VCS

Code Before

```
int i = n;
while (i−−)
   printf (" %d", i);
```

Code After

```
//print n integers iff n≥ 0
int i = n;
while (−−i > 0)
   printf (" %d", i);
```

one line deleted

two lines added

two lines unchanged

Other attributes: date: 2014-05-29 01:25:30, developer id: audris, branch: master, Comment: “Fix bug 3987 - infinite loop if n $\leq$ 0”

Software Tools Producing/Consuming OD

- Version control systems (VCS)
  - SCCS, CVS, ClearCase, SVN, Bzr, Hg, Git
- Issue tracking and customer relationship mgmt
  - Bugzilla, JIRA, ClearQuest, Siebel
- Code editing
  - vi, emacs, Eclipse, Sublime
- Communication
  - Twitter, IM, Forums
- Documentation
  - StackOverflow, Wikies, Redit
- Execution
  - GoogleAnalytics, AB testing, performance logs

Why Census Of Public Source Code?

- It allows us to compare different groups of software projects and technologies because the same information is recorded in the same way throughout all version control systems.
- The census provides information on the current state and trends that busineses, organizations, and government need to develop policies, plan and run private and public services, and allocate funding.

Why now?

- Finally feasible with the software tools residing not on premises but in the cloud: GitHub, BitBucket, . . .

Why Code?

- Code is Functional knowledge
  - Scholarly and literary works need a subject to interpret/perform them
  - Code just needs a computer to be executed
- Open source code
  - A vehicle for innovation through reuse (build on existing knowledge)
  - A common platform for everyone to express themselves (contribute their knowledge)
  - Critical (inter)national infrastructure
- Codebase for legacy systems encodes millions of person-years of tacit knowledge on:
  - the practices of producing the code and
  - the market (value the software provides to users)

Theoretical frameworks

- Developer activity traces as (biological) signaling [1]
- Source code reuse as a supply chain
- Source code reuse as a innovation engine [5]
- Code recommenders

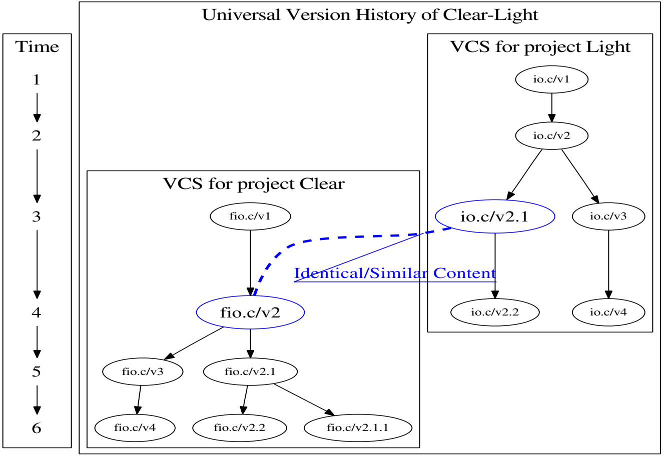
Why global properties of code?

- How much code? What is that code? How old, of what type, where?
  - Extent of code transfer/reuse: study patterns or reuse and innovation
- What types of projects are there, what types of technologies and practices are used, what are the outcomes?
- Full population is needed
  - To identify network structure, context, trends
  - Authorship (succession): Find Adam&Eve of code or identify original authors
  - License compliance: verify that code is not borrowed from public domain

Approach: Version Control Census

- Discover VCS repositories
- Copy/clone repositories
- Obtain complete metadata (commit dates/authors/comments)
- Extract and index all versions of each file
- Establish links across project repositories to create Universal Version History
  - Unlike people, files and their version histories can be and very often are copied
  - To avoid double-count for census and other analysis we thus need to create each file's "passport" or provenance

Identity/provenance of the code



How to construct Universal Version History?

- Establish links among files across multiple VCS
  - identical content: the closure of files sharing at least one identical version
  - Also: identical AST, Trigram, other ways to establish identity or similarity

Discovery strategy

- Sites with many projects: e.g., GitHub, BitBucket, dozens of other forges
- Ecosystems: e.g., Gnome, KDE, NetBeans, Mozilla, . . .
- Famous projects: e.g., Mysql, Perl, Wine, Postgres, and gcc
- In wide use: e.g., git.debian.org
- Published surveys of projects

How to automate VCS discovery?

- Create a spider utilizing a search engine, and seeded by project directories to grab these URLs from projects' home page
    - Search for VCS-specific URL patterns
        - cvs[:.], svn[:.], git[:.], hg[:.], bzr[:.]
        - Entice projects themselves to submit a pointer to their VCS by providing a compelling service (licensing, origin, quality)
- Example discovery/update challenge
    - 5% of top forges now defunct
    - Projects massively move towards Git as VCS

Copy, log, extract

	URL pattern	Clone repository	List revisions
CVS	d:pserver:user@cvs.repo.org:/	rsync	cvs log
Subversion	{svn,http}://PRJ.repo.org/	svm sync URL	svn log -v URL
Git	git://git.repo.org/	git clone URL PRJ	git log OPTIONS
Mercurial	hg://hg.repo.org/	hg clone URL	hg log -v
Bazaar	http://bzr.repo.org/	bzr branch URL	

	Extract content
CVS	rcs -pREV FILE
Subversion	svn cat -rREV URL/FILE@REV
Git	git show REV:FILE
Mercurial	hg cat -rREV FILE
Bazaar	bzr cat -rREV FILE

Current setup

- Four servers 396GB RAM and 24 cores each with
- 8TB SSD
- 200Tb online disk
- 100TB offline
- 1Gbit connection to outside

Workflow

- Chunks of 2TB
    - Clone (network bound)
        - Retrieval/cloning: No more than one(three) process per forge/repository (ethics)
    - Extract metadata: cpu/disk bound
        - Extraction: as some cloned repositories become available use all available processors (processing time), store content in intermediate hashtables to avoid bottleneck of a single table
    - Extract content: cpu/disk bound
    - Index content: disk (ssd) bound, can be distributed via pre-hashing
    - Further processing: trigrams, AST, do in parallel on all available servers after the main hashtable (composed of 100 tables) is complete

What is out there (2010)?

Forge	Type	VCSs	Files	File/Ver.	Uniq.	Space	From
Large cmpny.	Var.	>200	3,272K	12,585K	4,293K	remote	1988
SourceForge	CVS	121K	26,095K	81,239K	39,550K	820GB	1998
code.google	SVN	42K	5,675K	14,368K	8,584K	remote	1996
github.com	Git	29K	5,694K	18,986K	7,076K	154GB	1988
repo.or.cz	Git	1,867	2,519K	11,068K	5,115K	43GB	1986
gitorious.org	Git	1,098	1,229K	4,896K	1,749K	20GB	1988
Debian	Git	1,662	1,058K	4,741K	1,863K	19GB	1988
Savannah	CVS	2,946	852K	3,623K	2,345K	25GB	1985
objectweb.org	CVS	93	1,778K	2,287K	528K	17GB	1999
SourceWare	CVS	65	213K	1,459K	761K	10GB	1989
netbeans	Hg	57	185K	23,847K	492K	69GB	1999
rubyforge.org	SVN	3,825	456K	807K	256K	4.9GB	2001
Mozilla	Hg	14	58K	210K	105K	1.6GB	2000

...

Forge	Type	VCSs	Files	File/Ver.	Unique F/V	Space	From
git.kernel.org	Git	595	12,974K	97,585K	856K	205GB	1988
OpenSolaris	Hg	98	77K	1,108K	91K	9.7GB	2003
FreeBSD	CVS	1	196K	360K	75K	2.5GB	1993
Kde	SVN	1	2,645K	10,162K	527K	50GB	1997
gnome.org	SVN	566	1,284K	3,981K	1,412K	1GB	1997
Freedesktop	CVS	75	139K	784K	375K	4GB	1994
Gcc	SVN	1	3,758K	4,803K	395K	14GB	1989
Eclipse	CVS	9	729K	2,127K	575K	11GB	2001
OpenJDK	Hg	392	32K	747K	60K	15GB	2008
Mysql-Server	Bazaar	1	10K	523K	133K	6GB	2000
PostgreSQL	CVS	1	6K	108K	105K	0.5GB	1994
ruby-lang	SVN	1	163K	271K	56K	0.6GB	1998
Perl	Git	1	11,539	103K	42K	0.2GB	1988
Python	SVN	1	8K	89K	76,454	0.8GB	1991

What is out there (2015)?

30+M projects on GitHub (double from 2014)
0.7M projects on BitBucket (80% up from 2014)

How to use VCS census?

- Example applications
  - Software supply chain
    - Reduce risks by tracking down vulnerable code
    - Measure Truck Factor [4]
    - Evaluate/Identify Commercial Involvement [11]
    - Market research: what to use, where to contribute
  - Provide software engineering research base (most of the current SE work relies on such data)
    - Code/Expertise search
    - Automatic documentation
    - Universal defect predictors
    - Risk models

What are the main challenges?

- Operational data are treacherous - unlike experimental data [3]
  - Multiple contexts [9, 7, 8]
  - Missing events [2]
  - Incorrect, filtered, or tampered with [6, 10]
- Continuously changing
  - Systems and practices are evolving
- Challenges measuring or defining accuracy
- Potential for misinterpretation

OD: Multi-context, Missing, and Wrong

- Example issues with commits in VCS
  - Context:
    - Why: merge/push/branch, fix/enhance/license
    - What: e.g, code, documentation, build, binaries
    - Practice: e.g., centralized vs distributed
  - Missing: e.g., private VCS, no links to defect
  - Incorrect: tangled commits, misleading comments
  - Filtered: import from CVS/SVN
  - Tampered with: `git rebase`

Whats needed

- Hardware resources
  - To construct and analyze large graphs
  - To store 1PB of data
- Aproaches for data
  - contextualization,
  - augmentation (missing links),
  - correction

References

Laura A. Dabbish, H. Colleen Stuart, Jason Tsay, and James D. Herbsleb.
Social coding in github: transparency and collaboration in an open software repository.
In *CSCW*, pages 1277–1286, 2012.

Audris Mockus.
Missing data in software engineering.
In J. Singer et al., editor, *Guide to Advanced Empirical Software Engineering*, pages 185–200.
Springer-Verlag, 2008.

Audris Mockus.
Engineering big data solutions.
In *ICSE'14 FOSE*, 2014.

Peter Rigby, Yue Cai Zhu, Samuel M. Donadelli, and Audris Mockus.
Quantifying and mitigating turnover-induced knowledge loss: Case studies of chrome and a project at avaya.
In *ICSE'16*, Austin, Texas, May 2016. ACM.
Accepted.

Eric von Hippel and Georg von Krogh.
Open source software and the "private-collective" innovation model: Issues for organization science.
Organization Science, 14(2):209–223, March/April 2003.

Jialiang Xie, Qimu Zhengand, Minghui Zhou, and Audris Mockus.
Product assignment recommender.
In *ICSE'14 Demonstrations*, 2014.

Feng Zhang, Audris Mockus, Iman Keivanloo, and Ying Zou.
Towards building a universal defect prediction model.
In *11th IEEE Working Conference on Mining Software Repositories*, May 30–31 2014.