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
- Example applications
- Techniques critical for operational data
- Need research
Version Control Data

- Developers use VCS to track changes

Operational Data from VCS

**Code Before**

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

**Code After**

```c
//print n integers iff n \geq 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**

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

**Code After**

```c
//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**

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

**Code After**

```c
//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

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

Code After

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

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 ≤ 0"
Version Control Data

- Developers use VCS to track changes

Operational Data from VCS

Code Before

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

Code After

```c
//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 ≤ 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, Reddit
- **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 businesses, 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

Universal Version History of Clear-Light

VCS for project Clear

VCS for project Light

Identical/Similar Content
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

<table>
<thead>
<tr>
<th></th>
<th>URL pattern</th>
<th>Clone repository</th>
<th>List revisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVS</td>
<td>d:pserver:<a href="mailto:user@cvs.repo.org">user@cvs.repo.org</a>:/</td>
<td>rsync</td>
<td>cvs log</td>
</tr>
<tr>
<td>Subversion</td>
<td>{svn,http}://PRJ.repo.org/</td>
<td>svm sync URL</td>
<td>svn log -v URL</td>
</tr>
<tr>
<td>Git</td>
<td>git://git.repo.org/</td>
<td>git clone URL PRJ</td>
<td>git log OPTIONS</td>
</tr>
<tr>
<td>Mercurial</td>
<td>hg://hg.repo.org/</td>
<td>hg clone URL</td>
<td>hg log -v</td>
</tr>
<tr>
<td>Bazaar</td>
<td><a href="http://bzr.repo.org/">http://bzr.repo.org/</a></td>
<td>bzr branch URL</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Extract content</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CVS</td>
<td>rcs -pREV FILE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subversion</td>
<td>svn cat -rREV URL/FILE@REV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Git</td>
<td>git show REV:FILE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mercurial</td>
<td>hg cat -rREV FILE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bazaar</td>
<td>bzr cat -rREV FILE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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)?

<table>
<thead>
<tr>
<th>Forge</th>
<th>Type</th>
<th>VCSs</th>
<th>Files</th>
<th>File/Ver.</th>
<th>Uniq.</th>
<th>Space</th>
<th>From</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large cmpny.</td>
<td>Var.</td>
<td>&gt;200</td>
<td>3,272K</td>
<td>12,585K</td>
<td>4,293K</td>
<td>remote</td>
<td>1988</td>
</tr>
<tr>
<td>SourceForge</td>
<td>CVS</td>
<td>121K</td>
<td>26,095K</td>
<td>81,239K</td>
<td>39,550K</td>
<td>820GB</td>
<td>1998</td>
</tr>
<tr>
<td>code.google</td>
<td>SVN</td>
<td>42K</td>
<td>5,675K</td>
<td>14,368K</td>
<td>8,584K</td>
<td>remote</td>
<td>1996</td>
</tr>
<tr>
<td>github.com</td>
<td>Git</td>
<td>29K</td>
<td>5,694K</td>
<td>18,986K</td>
<td>7,076K</td>
<td>154GB</td>
<td>1988</td>
</tr>
<tr>
<td>repo.or.cz</td>
<td>Git</td>
<td>1,867</td>
<td>2,519K</td>
<td>11,068K</td>
<td>5,115K</td>
<td>43GB</td>
<td>1986</td>
</tr>
<tr>
<td>gitorious.org</td>
<td>Git</td>
<td>1,098</td>
<td>1,229K</td>
<td>4,896K</td>
<td>1,749K</td>
<td>20GB</td>
<td>1988</td>
</tr>
<tr>
<td>Debian</td>
<td>Git</td>
<td>1,662</td>
<td>1,058K</td>
<td>4,741K</td>
<td>1,863K</td>
<td>19GB</td>
<td>1988</td>
</tr>
<tr>
<td>Savannah</td>
<td>CVS</td>
<td>2,946</td>
<td>852K</td>
<td>3,623K</td>
<td>2,345K</td>
<td>25GB</td>
<td>1985</td>
</tr>
<tr>
<td>objectweb.org</td>
<td>CVS</td>
<td>93</td>
<td>1,778K</td>
<td>2,287K</td>
<td>528K</td>
<td>17GB</td>
<td>1999</td>
</tr>
<tr>
<td>SourceWare</td>
<td>CVS</td>
<td>65</td>
<td>213K</td>
<td>1,459K</td>
<td>761K</td>
<td>10GB</td>
<td>1989</td>
</tr>
<tr>
<td>netbeans</td>
<td>Hg</td>
<td>57</td>
<td>185K</td>
<td>23,847K</td>
<td>492K</td>
<td>69GB</td>
<td>1999</td>
</tr>
<tr>
<td>rubyforge.org</td>
<td>SVN</td>
<td>3,825</td>
<td>456K</td>
<td>807K</td>
<td>256K</td>
<td>4.9GB</td>
<td>2001</td>
</tr>
<tr>
<td>Mozilla</td>
<td>Hg</td>
<td>14</td>
<td>58K</td>
<td>210K</td>
<td>105K</td>
<td>1.6GB</td>
<td>2000</td>
</tr>
<tr>
<td>Forge</td>
<td>Type</td>
<td>VCSs</td>
<td>Files</td>
<td>File/Ver.</td>
<td>Unique F/V</td>
<td>Space</td>
<td>From</td>
</tr>
<tr>
<td>----------------</td>
<td>------</td>
<td>------</td>
<td>--------</td>
<td>-----------</td>
<td>------------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>git.kernel.org</td>
<td>Git</td>
<td>595</td>
<td>12,974K</td>
<td>97,585K</td>
<td>856K</td>
<td>205GB</td>
<td>1988</td>
</tr>
<tr>
<td>OpenSolaris</td>
<td>Hg</td>
<td>98</td>
<td>77K</td>
<td>1,108K</td>
<td>91K</td>
<td>9.7GB</td>
<td>2003</td>
</tr>
<tr>
<td>FreeBSD</td>
<td>CVS</td>
<td>1</td>
<td>196K</td>
<td>360K</td>
<td>75K</td>
<td>2.5GB</td>
<td>1993</td>
</tr>
<tr>
<td>Kde</td>
<td>SVN</td>
<td>1</td>
<td>2,645K</td>
<td>10,162K</td>
<td>527K</td>
<td>50GB</td>
<td>1997</td>
</tr>
<tr>
<td>gnome.org</td>
<td>SVN</td>
<td>566</td>
<td>1,284K</td>
<td>3,981K</td>
<td>1,412K</td>
<td>1GB</td>
<td>1997</td>
</tr>
<tr>
<td>Freedesktop</td>
<td>CVS</td>
<td>75</td>
<td>139K</td>
<td>784K</td>
<td>375K</td>
<td>4GB</td>
<td>1994</td>
</tr>
<tr>
<td>Gcc</td>
<td>SVN</td>
<td>1</td>
<td>3,758K</td>
<td>4,803K</td>
<td>395K</td>
<td>14GB</td>
<td>1989</td>
</tr>
<tr>
<td>Eclipse</td>
<td>CVS</td>
<td>9</td>
<td>729K</td>
<td>2,127K</td>
<td>575K</td>
<td>11GB</td>
<td>2001</td>
</tr>
<tr>
<td>OpenJDK</td>
<td>Hg</td>
<td>392</td>
<td>32K</td>
<td>747K</td>
<td>60K</td>
<td>15GB</td>
<td>2008</td>
</tr>
<tr>
<td>Mysql-Server</td>
<td>Bazaar</td>
<td>1</td>
<td>10K</td>
<td>523K</td>
<td>133K</td>
<td>6GB</td>
<td>2000</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>CVS</td>
<td>1</td>
<td>6K</td>
<td>108K</td>
<td>105K</td>
<td>0.5GB</td>
<td>1994</td>
</tr>
<tr>
<td>ruby-lang</td>
<td>SVN</td>
<td>1</td>
<td>163K</td>
<td>271K</td>
<td>56K</td>
<td>0.6GB</td>
<td>1998</td>
</tr>
<tr>
<td>Perl</td>
<td>Git</td>
<td>1</td>
<td>11,539</td>
<td>103K</td>
<td>42K</td>
<td>0.2GB</td>
<td>1988</td>
</tr>
<tr>
<td>Python</td>
<td>SVN</td>
<td>1</td>
<td>8K</td>
<td>89K</td>
<td>76,454</td>
<td>0.8GB</td>
<td>1991</td>
</tr>
</tbody>
</table>
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
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

- **Approaches for data**
  - contextualization,
  - augmentation (missing links),
  - correction
Laura A. Dabbish, H. Colleen Stuart, Jason Tsay, and James D. Herbsleb.
Social coding in github: transparency and collaboration in an open software repository.

Audris Mockus.
Missing data in software engineering.
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.

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.