CVS-Vintage: A Dataset of 14 CVS Repositories of Java Software
Martin Monperrus, Matias Martinez

To cite this version:
Martin Monperrus, Matias Martinez. CVS-Vintage: A Dataset of 14 CVS Repositories of Java Software. 2012. hal-00769121

HAL Id: hal-00769121
https://hal.science/hal-00769121
Preprint submitted on 28 Dec 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
CVS-Vintage: A Dataset of 14 CVS Repositories of Java Software

Martin Monperrus and Matias Martinez


Abstract

This paper presents a dataset of 14 CVS repositories of Java applications. This dataset aims at supporting the replication of early papers in the field of software evolution and mining software repositories. By building this dataset, we saved some CVS repositories from a probable death by deletion.

1 Introduction

The history of version control systems (sometimes called “revision control system” or “software configuration management system”) is long (e.g. Tichy’s paper [22] was published in 1985). But CVS has a special place in this history, because it has been the most successful version control system in the early days of open-source [10]. As a result, the generalized use of CVS has produced rich software repositories, and the first researchers who explored such repositories (e.g. [18]) did use CVS data. However, CVS is no longer a mainstream version control system, and this poses two challenges for research on software evolution.

First, there is a need for scientific data archiving. In many projects, CVS has been replaced by Subversion (aka SVN) in the mid 2000’s, or by a distributed version control system (Git, Mercurial) more recently. When a project did not migrate the content of the CVS repository, the history before the migration date may not be publicly available anymore, i.e. the original CVS repository may have been lost forever.

Second, many projects have migrated the content of the original CVS repository to a newer version control system. For instance, there is a tool called cvs2svn that migrates the software history from CVS to SVN. However, certain early papers on software evolution published tools which only take CVS data as input. Consequently, to run those tools, it is not sufficient to find the migrated data to support replication of early mining papers. Those two concerns, “archiving” and “replication”, motivated us to create a dataset of CVS repositories.

To create a dataset of CVS repositories, we crawled “A survey and taxonomy of approaches for mining software repositories” [12] to identify which CVS repositories have been used, and tried to find this data again. Sometimes, the data was still publicly
available; sometimes we had to personally ask the project members for obtaining it. This dataset focuses on CVS repositories of Java software. We made this design choice because we are interested in replicating experiments on mining software repositories that uses analysis methods that are specific to a given programming language, in our case, Java. Our approach resulted in a dataset of 14 CVS repositories called CVS-Vintage.

The CVS-Vintage dataset is available as supplementary data on the open access archive HAL (http://hal.archives-ouvertes.fr/).

2 Methodology

This section presents the methodology that we devised to create a dataset of CVS repositories of Java applications.

2.1 Inclusion Criteria

We read carefully “A survey and taxonomy of approaches for mining software repositories” [12] to identify software packages used in previous research. For papers that are not described with sufficient details, we reviewed the papers as well to identify missing software packages that may be relevant for our dataset. This resulted in 39 candidates. Out of those 39 candidates, we selected the packages that are mostly written in Java (according to the description of their homepage). This resulted in 14 Java software packages. Then, for each of them, we tried to identify the corresponding repository on the Internet, which means answering to the following questions: Which organization hosted the project at the time of the CVS repository (e.g. Sourceforge, Apache, Tigris, OW2)? If the repository is not available anymore, who to contact? We answered to those questions by thoroughly searching and browsing the web.

To sum up, a CVS repository is included in a dataset if and only if: 1) it is used in a paper cited in “A survey and taxonomy of approaches for mining software repositories” [12] 2) it contains software that is mostly written in Java.

2.2 Data Acquisition

A CVS repository is a folder containing a directory called “CVSROOT” and a set of folders. Those top-level folders are called a “module” in the CVS terminology. Each module can contain an arbitrary deep hierarchy of folders. Those folders contain RCS files [22] ending in with the “.v” extension. For instance, the whole history of “Foo.java” consists of revisions that are contained in “Foo.java,v”.

To obtain the CVS repositories of the selected 14 Java software packages, we used one of the following techniques:
Copy  Certain open-source hosting services support direct copy of the CVS repositories, often using the rsync protocol\(^1\). The main difficulty consists of finding the server name and the absolute path of the repository (both are rarely documented)\(^2\). In the case of OSS projects of the Eclipse ecosystem, the Eclipse Foundation distributes a gzipped version of many complete CVS repositories at http://archive.eclipse.org/arch/.

Direct Query  When the hosting provider does not support direct extraction, we asked the administrators of the hosting forge under consideration for a copy of the repository.

2.3 Post-processing

As discussed in 2.2, we obtained either a single CVS module or a set of CVS modules. If the former case, we kept this data as is for the dataset. In the latter case, when we obtained multi-module CVS repositories, we only kept the “dominant” CVS module, the one that contain the core functionalities (for instance module “jboss” in the jboss repository). In other terms, we always include one single CVS module per CVS repository\(^3\).

A CVS repository of open-source software often contains large binary files such as libraries, images, etc. To save bandwidth and facilitate analysis of Java files only, we set up two flavors of the dataset: the “full” version contain all files (incl. binaries); the “light” version, only contains the history of Java files (i.e. only with files ending in “.java,v”).

3 CVS Repositories

Our inclusion criteria (see 2.1) yields 14 open-source software packages: Argouml, Columba, Jboss, Jhotdraw, Log4j, org.eclipse.ui.workbench, Struts, Carol, Dnsjava, Jedit, Junit, org.eclipse.jdt.core, Scarab and Tomcat. This section presents those repositories in alphabetical order.

ArgoUML  ArgoUML is a modeling tool that has always been hosted at tigris.org. The project migrated to SVN in September 2006. The CVS repository is not publicly available anymore. However, we asked Jack Repenning from tigris.org and fortunately he could find the original repository on the server and send it to. ArgoUML has been used in many papers including [24, 6].

\(^1\)e.g. “$ rsync -av rsync://columba.cvs.sourceforge.net/cvsroot/columba/ columba”

\(^2\)Since CVS is no longer used, the rsync support is doomed to disappear. For instance, for Apache’s Log4j, Struts and Tomcat, we found their CVS repositories by chance using the rsync protocol on the server “minotaur.apache.org” on Feb 8, 2012. As of May 2012, this data is not available anymore.

\(^3\)for Carol, Columba, JHotdraw, Jboss, Junit
**Carol** Carol is a middleware for Java to abstract over concrete implementations of remote method invocations. Carol used CVS from August 2002 to May 2007. They then switched to SVN and the CVS repository files are not publicly available anymore. We obtained the files\(^4\) by contacting Jérémie Casery, the IT administrator of OW2, the consortium that hosts the project. Carol has been used in [13, 4].

**Columba** Columba is an email client hosted at sourceforge.net. The project migrated to SVN in July 2006 without migrating the CVS history. Fortunately, we learned from the sourceforge.net support that they always keep old versioning data, even when project switch to a new system. We could download the CVS data at columba.cvs.sourceforge.net (using the rsync protocol)\(^5\). Columba has been used in [5, 14, 15].

**Dnsjava** Dnsjava is a DNS client hosted at sourceforge.net. The project migrated to SVN in August 2009. Dnsjava has been used in [13, 3, 1].

**Eclipse** Eclipse is a integrated development environment (IDE) mostly developed by IBM. It is legally and technically hosted by a consortium called the “Eclipse Foundation”. With respect to mining software repositories, Eclipse is a monster. First, it is one of the latest major open-source projects who is still using CVS. Second, their 14 CVS repositories are on the order of magnitude of Gigabytes (on Feb 8, 2012 “eclipse-cvs.tgz” is 7.6GB compressed!) and millions of file revisions. Consequently, we have to select a subset of those 14 repositories, otherwise the dataset would be completely biased towards Eclipse data (in terms of time span, domain, development process, developers).

We selected the oldest CVS repository (“eclipse-cvs.tgz”) which contains the core functionalities. Inside this repository, we chose to include two modules in the CVS-Vintage dataset: “org.eclipse.jdt.core” and “org.eclipse.ui.workbench”. The former contains the code to manipulate Java code (e.g. compiling to bytecode or refactoring), the latter contains the core user-interface of Eclipse. The rationales are as follows: first, they are top-level directories (hence modules in the sense of CVS), this reflects their central place in the project since the beginning; second; they still correspond to a compilable units of well-defined functionality (both are Java projects in the sense of Eclipse); third, they are orthogonal in terms of domain (code manipulation versus user-interface) fourth, previous work already used this subset (e.g. reference [16] used “org.eclipse.jdt.core”); fifth, the number of revisions of those two modules is comparable to other repositories of the dataset. Eclipse has been used in many papers including [16, 4].

**JBoss** JBoss is an application server that was hosted at sourceforge.net. The project stopped using CVS in August 2005. We downloaded the CVS data on the Sourceforge

---

\(^4\)The Carol repository contains 4 modules, we selected the main one: “carol”.

\(^5\)The Columba repository contains 10 modules, we selected the main one: “columba”.

4

4
server jboss.cvs.sourceforge.net. The JBoss repository contains 112 CVS modules. This repository has been used to host an ecosystem of related packages rather than a clearly focused application. According to our heuristics (see Sec. 2), we selected the CVS module jboss because it corresponds to the core of JBoss and contains the largest number of revisions of Java files. JBoss has been used in [28, 26, 21].

**JEdit**  JEdit is a text editor that was hosted at gjt.org. The project stopped using CVS in July 2006. While the jEdit history has been migrated using cvs2svn, the original CVS repository is not publicly available anymore. We got it by contacting Alan Ezust, a key project member. JEdit has been used in [17, 28, 11].

**JHotDraw**  JHotDraw is library for building drawing-based user interfaces. The project, hosted at sourceforge.net, stopped using CVS in April 2005 but the CVS data is still available at jhotdraw.cvs.sourceforge.net. JHotDraw has been used in [5, 2, 27].

**JUnit**  JUnit is a testing framework, hosted at sourceforge.net. We downloaded the original CVS data spanning from Dec 2000 to Jan 2009 at junit.cvs.sourceforge.net. The repository is composed of 3 CVS modules including the main one called junit. JUnit has been used in [21, 25, 20].

**Log4j**  Log4j is a testing framework, hosted at apache.org. The original CVS repository was abandoned in September 2005 but we found it using the rsync protocol on an Apache server (see Sec. 2). Log4j has been used in [7, 9, 19].

**Scarab**  Scarab is an issue tracker that is hosted at tigris.org whose CVS repository is not publicly available anymore. As for ArgoUML, Jack Repenning from tigris.org sent us the archive. Scarab has been used in [15] and many other Kim’s papers.

**Struts**  Struts is a web application framework that is hosted at apache.org. As for log4j, we were able to identify and download the original CVS history from Apache using the rsync protocol. This CVS history goes from June 2000 to September 2004. Struts has been used in [9, 8].

**Tomcat**  Tomcat is an application server. The project is hosted at apache.org. We could download and include this package in the dataset with the same protocol as log4j and struts. Tomcat has been used in [23, 25].

**Descriptive Statistics**  Table 1 shows descriptive statistics of the dataset. It gives the first and last revision date of the repository, the number of files per repository (all files and Java files only), the number of revisions (all files and Java revisions only).

---

Footnote: The JHotDraw repository contains 4 modules, we selected the main one: “jhotdraw6”.  

<table>
<thead>
<tr>
<th>Name</th>
<th>First Rev.</th>
<th>Last Rev.</th>
<th>#Files</th>
<th>#Java Files</th>
<th>%Java Files</th>
<th>#Rev.</th>
<th>#Java Rev.</th>
<th>%Java Rev.</th>
<th>Avail.</th>
<th>Migr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>argouml</td>
<td>1998/1/26</td>
<td>2006/9/28</td>
<td>10621</td>
<td>4542</td>
<td>42.7%</td>
<td>72356</td>
<td>51397</td>
<td>71%</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>carol</td>
<td>2002/8/6</td>
<td>2007/5/23</td>
<td>548</td>
<td>336</td>
<td>61.3%</td>
<td>2407</td>
<td>1439</td>
<td>59.8%</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>columba</td>
<td>2001/4/8</td>
<td>2006/7/28</td>
<td>7731</td>
<td>4503</td>
<td>58.2%</td>
<td>35142</td>
<td>27596</td>
<td>78.5%</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>dnsjava</td>
<td>1998/9/6</td>
<td>2009/8/8</td>
<td>376</td>
<td>354</td>
<td>94%</td>
<td>5763</td>
<td>5259</td>
<td>91.2%</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>eclipse.jdt.core</td>
<td>2001/6/5</td>
<td>2011/9/23</td>
<td>1911</td>
<td>1715</td>
<td>89.7%</td>
<td>80222</td>
<td>64976</td>
<td>80.9%</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>eclipse.ui.workbench</td>
<td>2002/9/24</td>
<td>2011/6/24</td>
<td>4217</td>
<td>3733</td>
<td>88.5%</td>
<td>40894</td>
<td>38701</td>
<td>94.6%</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>jboss</td>
<td>2000/4/22</td>
<td>2005/8/18</td>
<td>2516</td>
<td>1933</td>
<td>76.8%</td>
<td>23034</td>
<td>18818</td>
<td>81.7%</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>jedit</td>
<td>2001/9/2</td>
<td>2006/7/25</td>
<td>1503</td>
<td>593</td>
<td>39.4%</td>
<td>12949</td>
<td>7033</td>
<td>54.3%</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>jhotdraw</td>
<td>2000/10/12</td>
<td>2005/4/26</td>
<td>634</td>
<td>504</td>
<td>79.5%</td>
<td>3698</td>
<td>3227</td>
<td>87.2%</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>junit</td>
<td>2000/12/3</td>
<td>2009/1/28</td>
<td>1416</td>
<td>1198</td>
<td>84.6%</td>
<td>5833</td>
<td>5037</td>
<td>86.3%</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>log4j</td>
<td>2000/11/16</td>
<td>2005/9/8</td>
<td>2253</td>
<td>1069</td>
<td>47.4%</td>
<td>12260</td>
<td>7519</td>
<td>61.3%</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>scarab</td>
<td>2000/12/18</td>
<td>2005/7/04</td>
<td>3164</td>
<td>1073</td>
<td>33.9%</td>
<td>26393</td>
<td>10778</td>
<td>40.8%</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>struts</td>
<td>2000/6/1</td>
<td>2004/9/26</td>
<td>4062</td>
<td>1354</td>
<td>33.3%</td>
<td>17695</td>
<td>9088</td>
<td>51.3%</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>tomcat</td>
<td>1999/10/9</td>
<td>2005/9/13</td>
<td>2298</td>
<td>1134</td>
<td>49.3%</td>
<td>13528</td>
<td>8394</td>
<td>62%</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

Table 1: Descriptive Statistics of the CVS-Vintage Dataset.

and the relative frequency of Java files and Java revisions. It also indicates whether the CVS repository is still publicly available and whether the history was migrated to a newer version control system (for Eclipse JDT and Workbench UI, “-” means that they still use CVS). This table supports the following interpretation. First, most of the repositories have a similar time span, the projects started using CVS around 2000 and stopped using it around 2005. Second, the selected repositories actually mostly contain Java software with respect to the ratio of Java files and Java revisions. For instance, the “argouml” repository (first row) contains 42.7% of Java files and 71% of revisions concern java source code. Third, the size of the repositories in terms of Java revisions are commensurable, the biggest repository, “org.eclipse.jdt.core”, accounts for 24.8% of the dataset.

Finally, our initial goal of “archiving” has proved relevant: in the process of creating this dataset, we probably “saved” 4 repositories that were already not publicly available anymore.

4 Conclusion

We have presented a methodology to create a dataset of CVS repositories. The resulting dataset contains 14 CVS repositories of Java software and 352182 file revisions (in the sense of Tichy’s RCS). In the process of creating this dataset, we “saved” some data since certain repositories were already not publicly available anymore and planned for deletion by the administrators of the corresponding hosting forge.

The CVS-Vintage dataset is available as supplementary data on the open access archive HAL (http://hal.archives-ouvertes.fr/).
References


