2015 12th Working Conference on Mining Software Repositories

A Repository with 44 Years of Unix Evolution

Diomidis Spinellis
Department of Management Science and Technology
Athens University of Economics and Business
Patision 76, GR-104 34 Athens, Greece
Email: dds@aueb.gr

***Abstract*—The evolution of the Unix operating system is made available as a version-control repository, covering the period from its inception in 1972 as a five thousand line kernel, to 2015 as a widely-used 26 million line system. The repository contains 659 thousand commits and 2306 merges. The repository employs the commonly used Git system for its storage, and is hosted on the popular GitHub archive. It has been created by synthesizing with custom software 24 snapshots of systems developed at Bell Labs, Berkeley University, and the 386BSD team, two legacy repositories, and the modern repository of the open source FreeBSD system. In total, 850 individual contributors are identified, the early ones through primary research. The data set can be used for empirical research in software engineering, information systems, and software archaeology.**

I. Introduction

The Unix operating system stands out as a major engineering breakthrough due to its exemplary design, its numerous technical contributions, its development model, and its widespread use. The design of the Unix programming environment has been characterized as one offering unusual simplicity, power, and elegance [1]. On the technical side, features that can be directly attributed to Unix or were popularized by it include [2]: the portable implementation of the kernel in a high level language; a hierarchical file system; compatible file, device, networking, and inter-process I/O; the pipes and filters architecture; virtual file systems; and the shell as a user-selectable regular process. A large community contributed software to Unix from its early days [3], [4, pp. 65–72]. This community grew immensely over time and worked using what are now termed open source software development methods [5, pp. 440-442]. Unix and its intellectual descendants have also helped the spread of the C and C++ programming languages, parser and lexical analyzer generators (*yacc*, *lex*), document preparation tools (*troff*, *eqn*, *tbl*), scripting languages (*awk*, *sed*, *Perl*), TCP/IP networking, and configuration management systems (*SCCS*, *RCS*, *Subversion*, *Git*), while also forming a large part of the modern internet infrastructure and the web.

Luckily, important Unix material of historical importance has survived and is nowadays openly available. Although Unix was initially distributed with relatively restrictive licenses, the most significant parts of its early development have been released by one of its right-holders (Caldera International) under a liberal license. Combining these parts with software that was developed or released as open source software by the University of California, Berkeley and the FreeBSD Project provides coverage of the system's development over a period ranging from June 20th 1972 until today.

Curating and processing available snapshots as well as old and modern configuration management repositories allows the reconstruction of a new synthetic Git repository that combines under a single roof most of the available data. This repository documents in a digital form the detailed evolution of an important digital artefact over a period of 44 years. The following sections describe the repository's structure and contents (Section II), the way it was created (Section III), and how it can be used (Section IV).

II. Data Overview

The 1GB Unix history Git repository is made available for cloning on GitHub.[1] Currently[2] the repository contains 659 thousand commits and 2306 merges from about 850 contributors. The contributors include 23 from the Bell Labs staff, 158 from Berkeley's Computer Systems Research Group (CSRG), and 660 from the FreeBSD Project.

The repository starts its life at a tag identified as *Epoch*, which contains only licensing information and its modern README file. Various tag and branch names identify points of significance.

- *Research–VX* tags correspond to six research editions that came out of Bell Labs. These start with *Research–V1* (4768 lines of PDP-11 assembly) and end with *Research–V7* (1820 mostly C files, 324kLOC).
- *Bell-32V* is the port of the 7th Edition Unix to the DEC/VAX architecture.
- *BSD–X* tags correspond to 15 snapshots released from Berkeley.
- *386BSD–X* tags correspond to two open source versions of the system, with the Intel 386 architecture kernel code mainly written by Lynne and William Jolitz.
- *FreeBSD–release/X* tags and branches mark 116 releases coming from the FreeBSD project.

In addition, branches with a *–Snapshot–Development* suffix denote commits that have been synthesized from a time-ordered sequence of a snapshot's files, while tags with a –*VCS–Development* suffix mark the point along an imported version control history branch where a particular release occurred.

[1]https://github.com/dspinellis/unix-history-repo
[2]Updates may add or modify material. To ensure replicability the repository's users are encouraged to fork it or archive it.

978-0-7695-5594-2/15 $31.00 © 2015 IEEE
DOI 10.1109/MSR.2015.64

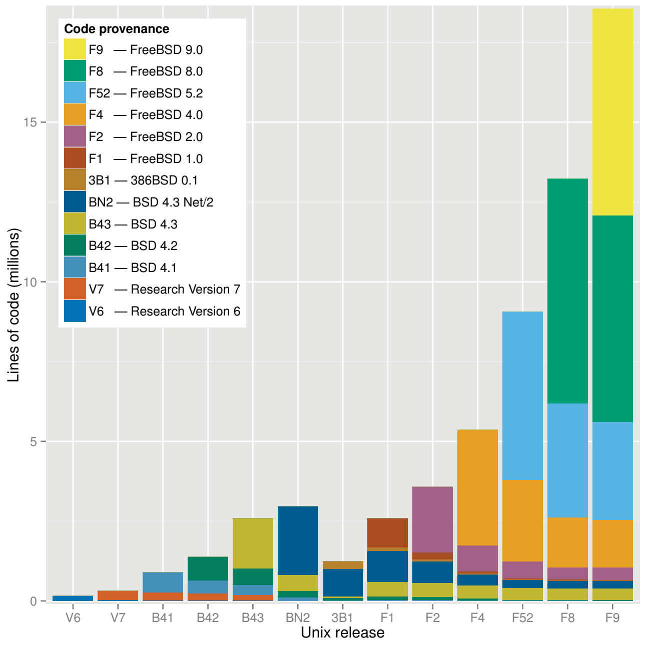

Fig. 1. Code provenance across significant Unix releases.

The repository's history includes commits from the earliest days of the system's development, such as the following.

```
commit c9f643f59434f14f774d61ee3856972b8c3905b1
Author: Dennis Ritchie <research!dmr>
Date:   Mon Dec 2 18:18:02 1974 -0500
    Research V5 development
    Work on file usr/sys/dmr/kl.c
```

Merges between releases that happened along the system's evolution, such as the development of BSD 3 from BSD 2 and Unix 32/V, are also correctly represented in the Git repository as graph nodes with two parents.

More importantly, the repository is constructed in a way that allows *git blame*, which annotates source code lines with the version, date, and author associated with their first appearance, to produce the expected code provenance results. For example, checking out the *BSD–4* tag, and running *git blame* on the kernel's *pipe.c* file will show lines written by Ken Thompson in 1974, 1975, and 1979, and by Bill Joy in 1980. This allows the automatic (though computationally expensive) detection of the code's provenance at any point of time.

As can be seen in Figure 1, a modern version of Unix (FreeBSD 9) still contains visible chunks of code from BSD 4.3, BSD 4.3 Net/2, and FreeBSD 2.0. Interestingly, the Figure shows that code developed during the frantic dash to create an open source operating system out of the code released by Berkeley (386BSD and FreeBSD 1.0) does not seem to have survived. The oldest code in FreeBSD 9 appears to be an 18-line sequence in the C library file `timezone.c`, which can also be found in the 7th Edition Unix file with the same name and a time stamp of January 10th, 1979 — 36 years ago.

III. Data Collection and Processing

The goal of the project is to consolidate data concerning the evolution of Unix in a form that helps the study of the system's evolution, by entering them into a modern revision repository. This involves collecting the data, curating them, and synthesizing them into a single Git repository.

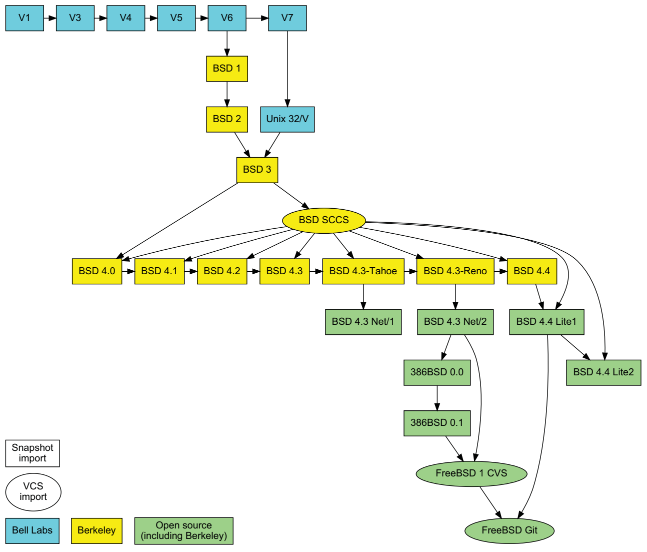

Fig. 2. Imported Unix snapshots, repositories, and their mergers.

The project is based on three types of data (see Figure 2). First, **snapshots of early released versions**, which were obtained from the Unix Heritage Society archive,[3] the CD-ROM images containing the full source archives of CSRG,[4] the OldLinux site,[5] and the FreeBSD archive.[6] Second, **past and current repositories**, namely the CSRG SCCS [6] repository, the FreeBSD 1 CVS repository, and the Git mirror of modern FreeBSD development.[7] The first two were obtained from the same sources as the corresponding snapshots.

The last, and most labour intensive, source of data was **primary research**. The release snapshots do not provide information regarding their ancestors and the contributors of each file. Therefore, these pieces of information had to be determined through primary research. The authorship information was mainly obtained by reading author biographies, research papers, internal memos, and old documentation scans; by reading and automatically processing source code and manual page markup; by communicating via email with people who were there at the time; by posting a query on the Unix *StackExchange* site; by looking at the location of files (in early editions the kernel source code was split into `usr/sys/dmr` and `/usr/sys/ken`); and by propagating authorship from research papers and manual pages to source code and from one release to others. (Interestingly, the 1st and 2nd Research Edition manual pages have an "owner" section, listing the person (e.g. *ken*) associated with the corresponding system command, file, system call, or library function. This section was not there in the 4th Edition, and resurfaced as the "Author"

[3]http://www.tuhs.org/archive_sites.html
[4]https://www.mckusick.com/csrg/
[5]http://www.oldlinux.org/Linux.old/distributions/386BSD
[6]http://ftp-archive.freebsd.org/pub/FreeBSD-Archive/old-releases/
[7]https://github.com/freebsd/freebsd

section in BSD releases.) Precise details regarding the source of the authorship information are documented in the project's files that are used for mapping Unix source code files to their authors and the corresponding commit messages. Finally, information regarding merges between source code bases was obtained from a BSD family tree maintained by the NetBSD project.[8]

The software and data files that were developed as part of this project, are available online,[9] and, with appropriate network, CPU and disk resources, they can be used to recreate the repository from scratch. The authorship information for major releases is stored in files under the project's `author-path` directory. These contain lines with a regular expressions for a file path followed by the identifier of the corresponding author. Multiple authors can also be specified. The regular expressions are processed sequentially, so that a catch-all expression at the end of the file can specify a release's default authors. To avoid repetition, a separate file with a `.au` suffix is used to map author identifiers into their names and emails. One such file has been created for every community associated with the system's evolution: Bell Labs, Berkeley, 386BSD, and FreeBSD. For the sake of authenticity, emails for the early Bell Labs releases are listed in UUCP notation (e.g. `research!ken`). The FreeBSD author identifier map, required for importing the early CVS repository, was constructed by extracting the corresponding data from the project's modern Git repository. In total the commented authorship files (828 rules) comprise 1107 lines, and there are another 640 lines mapping author identifiers to names.

The curation of the project's data sources has been codified into a 168-line `Makefile`. It involves the following steps.

a) Fetching: Copying and cloning about 11GB of images, archives, and repositories from remote sites.

b) Tooling: Obtaining an archiver for old PDP-11 archives from 2.9 BSD, and adjusting it to compile under modern versions of Unix; compiling the 4.3 BSD *compress* program, which is no longer part of modern Unix systems, in order to decompress the 386BSD distributions.

c) Organizing: Unpacking archives using *tar* and *cpio*; combining three 6th Research Edition directories; unpacking all 1 BSD archives using the old PDP-11 archiver; mounting CD-ROM images so that they can be processed as file systems; combining the 8 and 62 386BSD floppy disk images into two separate files.

d) Cleaning: Restoring the 1st Research Edition kernel source code files, which were obtained from printouts through optical character recognition, into a format close to their original state; patching some 7th Research Edition source code files; removing metadata files and other files that were added after a release, to avoid obtaining erroneous time stamp information; patching corrupted SCCS files; processing the early FreeBSD CVS repository by removing CVS symbols assigned to multiple revisions with a custom Perl script, deleting CVS *Attic* files clashing with live ones, and converting the CVS repository into a Git one using *cvs2svn*.

An interesting part of the repository representation is how snapshots are imported and linked together in a way that allows *git blame* to perform its magic. Snapshots are imported into the repository as sequential commits based on the time stamp of each file. When all files have been imported the repository is tagged with the name of the corresponding release. At that point one could delete those files, and begin the import of the next snapshot. Note that the *git blame* command works by traversing backwards a repository's history, and using heuristics to detect code moving and being copied within or across files. Consequently, deleted snapshots would create a discontinuity between them, and prevent the tracing of code between them.

Instead, before the next snapshot is imported, all the files of the preceding snapshot are moved into a hidden look-aside directory named `.ref` (reference). They remain there, until all files of the next snapshot have been imported, at which point they are deleted. Because every file in the `.ref` directory matches exactly an original file, *git blame* can determine how source code moves from one version to the next via the `.ref` file, without ever displaying the `.ref` file. To further help the detection of code provenance, and to increase the representation's realism, each release is represented as a merge between the branch with the incremental file additions (*–Development*) and the preceding release.

For a period in the 1980s, only a subset of the files developed at Berkeley were under SCCS version control. During that period our unified repository contains imports of both the SCCS commits, and the snapshots' incremental additions. At the point of each release, the SCCS commit with the nearest time stamp is found and is marked as a merge with the release's incremental import branch. These merges can be seen in the middle of Figure 2.

The synthesis of the various data sources into a single repository is mainly performed by two scripts. A 780-line Perl script (`import-dir.pl`) can export the (real or synthesized) commit history from a single data source (snapshot directory, SCCS repository, or Git repository) in the *Git fast export* format. The output is a simple text format that Git tools use to import and export commits. Among other things, the script takes as arguments the mapping of files to contributors, the mapping between contributor login names and their full names, the commit(s) from which the import will be merged, which files to process and which to ignore, and the handling of "reference" files. A 450-line shell script creates the Git repository and calls the Perl script with appropriate arguments to import each one of the 27 available historical data sources. The shell script also runs 30 tests that compare the repository at specific tags against the corresponding data sources, verify the appearance and disappearance of look-aside directories, and look for regressions in the count of tree branches and merges and the output of *git blame* and *git log*. Finally, *git* is called to garbage-collect and compress the repository from its initial 6GB size down to the distributed 1GB.

[8]http://ftp.netbsd.org/pub/NetBSD/NetBSD-current/src/share/misc/bsd-family-tree

[9]https://github.com/dspinellis/unix-history-make

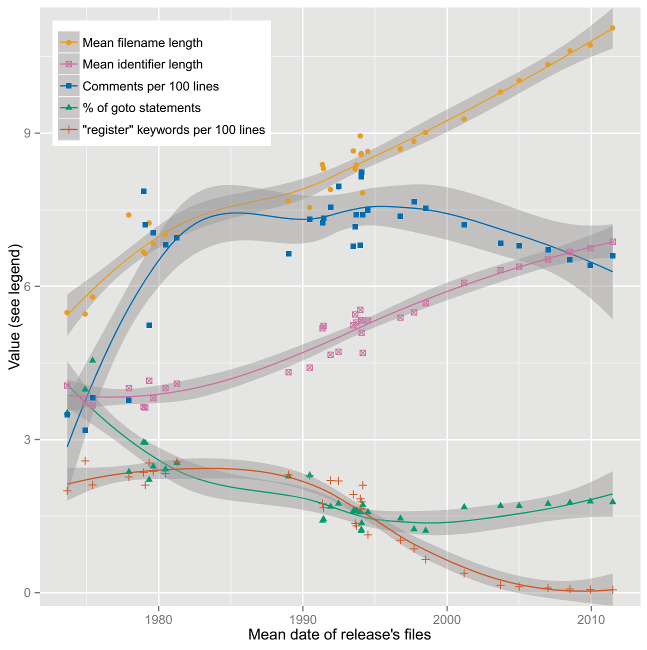

Fig. 3. Code style evolution along Unix releases.

IV. Data Uses

The data set can be used for empirical research in software engineering, information systems, and software archeology. Through its unique uninterrupted coverage of a period of more than 40 years, it can inform work on software evolution and handovers across generations. With thousandfold increases in processing speed and million-fold increases in storage capacity during that time, the data set can also be used to study the co-evolution of software and hardware technology. The move of the software's development from research labs, to academia, and to the open source community can be used to study the effects of organizational culture on software development. The repository can also be used to study how notable individuals, such as Turing Award winners (Dennis Ritchie and Ken Thompson) and captains of the IT industry (Bill Joy and Eric Schmidt), actually programmed. Another phenomenon worthy of study concerns the longevity of code, either at the level of individual lines, or as complete systems that were at times distributed with Unix (Ingres, Lisp, Pascal, Ratfor, Snobol, TMG), as well as the factors that lead to code's survival or demise. Finally, because the data set stresses Git, the underlying software repository storage technology, to its limits, it can be used to drive engineering progress in the field of revision management systems.

Figure 3, which depicts trend lines (obtained with R's local polynomial regression fitting function) of some interesting code metrics along 36 major releases of Unix, demonstrates the evolution of code style and programming language use over very long timescales. This evolution can be driven by software and hardware technology affordances and requirements, software construction theory, and even social forces. The dates in the Figure have been calculated as the average date of all files appearing in a given release. As can be seen in it, over the past 40 years the mean length of identifiers and file names has steadily increased from 4 and 6 characters to 7 and 11 characters, respectively. We can also see less steady increases in the number of comments and decreases in the use of the *goto* statement, as well as the virtual disappearance of the *register* type modifier.

V. Further Work

Many things can be done to increase the repository's faithfulness and usefulness. Given that the build process is shared as open source code, it is easy to contribute additions and fixes through GitHub pull requests. The most useful community contribution would be to increase the coverage of imported snapshot files that are attributed to a specific author. Currently, about 90 thousand files (out of a total of 160 thousand) are getting assigned an author through a default rule. Similarly, there are about 250 authors (primarily early FreeBSD ones) for which only the identifier is known. Both are listed in the build repository's `unmatched` directory, and contributions are welcomed. Furthermore, the BSD SCCS and the FreeBSD CVS commits that share the same author and time-stamp can be coalesced into a single Git commit. Support can be added for importing the SCCS file comment fields, in order to bring into the repository the corresponding metadata. Finally, and most importantly, more branches of open source systems can be added, such as NetBSD OpenBSD, DragonFlyBSD, and *illumos*. Ideally, current right holders of other important historical Unix releases, such as System III, System V, NeXTSTEP, and SunOS, will release their systems under a license that would allow their incorporation into this repository for study.

Acknowledgements

The author thanks the many individuals who contributed to the effort. Brian W. Kernighan, Doug McIlroy, and Arnold D. Robbins helped with Bell Labs login identifiers. Clem Cole, Era Eriksson, Mary Ann Horton, Kirk McKusick, Jeremy C. Reed, Ingo Schwarze, and Anatole Shaw helped with BSD login identifiers. The BSD SCCS import code is based on work by H. Merijn Brand and Jonathan Gray.

This research has been co-financed by the European Union (European Social Fund — ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) — Research Funding Program: Thalis — Athens University of Economics and Business — Software Engineering Research Platform.

References

[1] M. D. McIlroy, E. N. Pinson, and B. A. Tague, "UNIX time-sharing system: Foreword," *The Bell System Technical Journal*, vol. 57, no. 6, pp. 1899–1904, July-August 1978.
[2] D. M. Ritchie and K. Thompson, "The UNIX time-sharing system," *Bell System Technical Journal*, vol. 57, no. 6, pp. 1905–1929, July-August 1978.
[3] D. M. Ritchie, "The evolution of the UNIX time-sharing system," *AT&T Bell Laboratories Technical Journal*, vol. 63, no. 8, pp. 1577–1593, Oct. 1984.
[4] P. H. Salus, *A Quarter Century of UNIX*. Boston, MA: Addison-Wesley, 1994.
[5] E. S. Raymond, *The Art of Unix Programming*. Addison-Wesley, 2003.
[6] M. J. Rochkind, "The source code control system," *IEEE Transactions on Software Engineering*, vol. SE-1, no. 4, pp. 255–265, 1975.