2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)

Half-Century of Unix: History, Preservation, and Lessons Learned

Diomidis Spinellis
Department of Management Science and Technology
Athens University of Economics and Business
Athens, Greece
dds@aueb.gr

***Abstract*—The Unix operating system stands out as a major engineering achievement due to its exemplary design, technical contributions, impact, development model, and widespread use. Its evolution is made available as a revision management repository, covering the period from its inception in 1970 until today. The data set is being used for empirical research in the evolution of code, programming practices, systems, and software architecture.**

***Index Terms*—Unix; software evolution; software preservation; software archeology; Git; GitHub**

By the end of this decade Unix will be turning 50 years old. The Unix operating system stands out as a major engineering achievement with a design that has been characterized as one offering unusual simplicity, power, and elegance. A large community contributed software to Unix from its early days, grew immensely over time, and worked using what we now term open source software development methods. Unix and its intellectual descendants have also helped the spread of many key computing technologies and form a large part of the modern internet infrastructure and the web. Given the system's stellar pedigree, extraordinary impact, maturity, and widespread use, preserving and studying its history can help us understand, appreciate, and learn from long-term software evolution.

The evolution of the Unix operating system is made available as a revision management repository,[1] covering the period from its inception in 1970 as a 2.5 thousand line kernel with 26 commands, to 2017 as a widely-used 27 million line system [1]. It can then be used for empirical research in software engineering, information systems, and software archeology. The 1.1GB repository contains half a million commits, employing the commonly used Git version control system for its storage and the popular GitHub archive for its hosting. It has been created by synthesizing with custom software 25 snapshots of systems developed at Bell Labs, the University of California at Berkeley, and the 386BSD team, as well as hundreds of subsequent patches, two legacy repositories, and the modern repository of the open source FreeBSD system. Early parts of the system are integrated from digitized versions of surviving printouts restored through community efforts, demonstrating the difficulty and importance of software artefact preservation. In total, about one thousand individual contributors are identified, the early ones through primary research. These include 29 from the Bell Labs staff, 158 from Berkeley's Computer Systems Research Group (CSRG), 79 contributors of the 386BSD patch kit, and about 700 from the FreeBSD Project. The creation involved techniques for maintaining links between snapshots, merging related commits, and integrating decades-old commits with their current GitHub authors.

Based on the Unix history repository, we formulated seven hypotheses associated with the long term evolution of C programming, and examined them by extracting, aggregating, and synthesising metrics from 66 snapshots obtained from it [2]. We found that over the years developers of the Unix operating system appear to have evolved their coding style in tandem with advancements in hardware technology, promoted modularity to tame rising complexity, adopted valuable new language features, allowed compilers to allocate registers on their behalf, and reached broad agreement regarding code formatting. The progress we have observed appears to be slowing or even reversing prompting the need for new fruitful programming practices to be discovered and followed.

Currently the Unix history repository is being extended with a homogenized database of its evolving facilities covering all major releases. This is being used to study the long-term evolution of software architecture by identifying the early appearance of patterns, components, and connectors. These include the application of layering, the use of interpreters and command files, the definition of a stable, binary, language-independent API and a reference architecture, the abstraction of I/O and device drivers, the promotion of interoperability through documented file formats, the reuse of data structure definitions, and the adoption automatic code generation. Quantitative metrics of its evolution demonstrate that over the past half century the Unix system grew in similar proportion in the number of all components and connectors.

[1] https://github.com/dspinellis/unix-history-repo

References

[1] D. Spinellis, "A repository of Unix history and evolution," *Empirical Software Engineering*, 2016.

[2] D. Spinellis, P. Louridas, and M. Kechagia, "The evolution of C programming practices: A study of the Unix operating system 1973–2015," in *ICSE '16: Proceedings of the 38th International Conference on Software Engineering*, W. Visser and L. Williams, Eds. New York: Association for Computing Machinery, May 2016, pp. 748–759.

978-1-5386-1544-7/17 $31.00 © 2017 IEEE
DOI 10.1109/MSR.2017.1