Should We Preserve the World's Software History, And Can We?

Roberto Di Cosmo(✉)

Software Heritage, Inria and Université Paris Cité, Paris, France
roberto@dicosmo.org

Abstract. Cultural heritage is the legacy of physical artifacts and intangible attributes of a group or society that a re inherited from past generations, maintained in the present and bestowed for the benefit of future generations.

What role does software play in it? We claim that software source code is an important product of human creativity, and embodies a growing part of our scientific, organisational and technological knowledge: it is a part of our cultural heritage, and it is our collective responsibility to ensure that it is not lost.

Preserving the history of software is also a key enabler for reproducibility of research, and as a means to foster better and more secure software for society. This is the mission of Software Heritage, a non-profit organization dedicated to building the universal archive of software source code, catering to the needs of science, industry and culture, for the benefit of society as a whole.

In this keynote talk we survey the principles and key technology used in the archive that contains over 12 billion unique source code files from some 180 millions projects worldwide.

1 Introduction

Software is incorporating an important part of our scientific, technical and industrial heritage.

If one looks closely, it is easy to see that the real knowledge that is contained in software is not in the executable programs, but in the "source code", which according to the definition used in the GPL, is "*the preferred form for a developer to make a change to a program*"[1]. Source code is a special form of knowledge: it is made to be understood by a human being, the developer, and can be mechanically translated into a form to be executed directly on a machine. The very terminology used by the computing community is telling: "programming languages" are used to "write" software. As Harold Habelson wrote as early as 1985, "programs must be written for humans to read, and only accessorily for machines to execute"[2]

[1] Gnu general public license, version 2, 1991. Retrieved September 2015.

[2] Preface to Abelson, H., and Sussman, G. J. S. (1985). Structure and Interpretation of Computer Programs. The MIT Press.

G. Silvello et al. (Eds.): TPDL 2022, LNCS 13541, pp. 3–7, 2022.
https://doi.org/10.1007/978-3-031-16802-4_1

The *source code* of software is therefore a human creation in the same way as other written documents, and software developers deserve the same respect as other creators [5].

Software source code is therefore a valuable heritage, and it is essential to work on its preservation.

This is one of the missions of Software Heritage, an initiative launched in 2015 with the support of Inria and in partnership with UNESCO, to *collect*, *organize*, *preserve* and *make easily accessible* all publicly available source code on the planet, regardless of where and how it was developed or distributed.

2 A Complex Task

Archiving all available source code is a complex task: as detailed in the article [1], different strategies are needed depending on whether one seeks to collect open or proprietary source code, and one does not treat source code that is readily available online in the same way as source code that resides on older physical media.

For open source code that is readily available online, the most appropriate approach is to build a harvester that automatically collects content from a wide variety of collaborative development platforms, such as GitHub, GitLab.com, or BitBucket, or from software package distribution platforms, such as Debian, NPM. CRAN or Pypi.

For the source code of old software, a real process of computer archaeology must be set up, and we have already started this work in a collaboration with the University of Pisa and UNESCO that has resulted in the SWHAP process that has been used to find, document and archive software that is important in the history of computing in Italy (see https://www.softwareheritage.org/swhap), and which has recently been extended with the Software Stories project, which aims to highlight all the historical elements around a software whose source code has been found (see https://stories.softwareheritage.org).

3 A Universal Mission

The founding principles of Software Heritage are [1]: systematic use of open source software to build the Software Heritage infrastructure, so that its operation can be understood, and replicated if necessary; construction of a global network of independent mirrors of the archive, because a large number of copies is the best protection against loss and attack; choice of a non-profit, international, multi-stakeholder structure, to minimize the risk of having single points of failure, and to ensure that Software Heritage will indeed serve all.

For such a mission, institutional legitimacy is required, as well as openness to federate a broad consensus. This is why the framework agreement signed between Inria and UNESCO on April 3, 2017, and renewed in November 2021, is essential.

4 Past, Present, Future: Much More Than an Archive

Software Heritage is now an infrastructure that grows day by day: the bulk of the archive's content is the result of automatic harvesting, but pearls are beginning to be introduced through the patient work of recovering significant historical software, following the SWHAP acquisition process that has been developed in collaboration with the University of Pisa and UNESCO.

Exhaustiveness is still far from being achieved, but the archive already contains the largest corpus of source code available on the planet, with more than 180 million archived origins, for over 12 billion unique source files, each equipped with an intrinsic identifier based on cryptographic hashes [2,3] (Fig. 1).

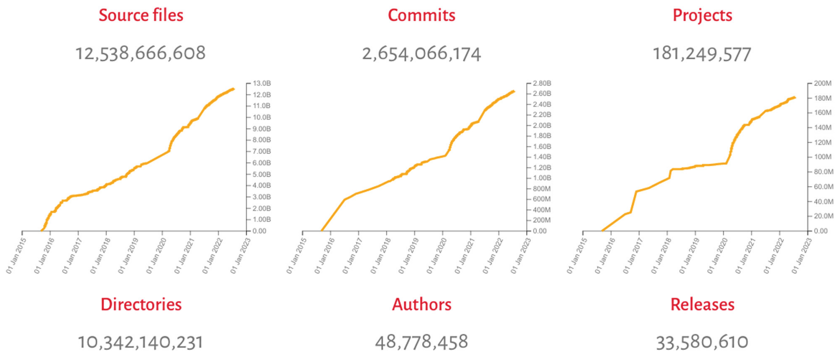

Fig. 1. Number of projects, source files, and versions archived in Sofware Heritage as of June 2022 (see `https://www.softwareheritate.org/archive`)

This unique infrastructure has a multiple mission: of course, it is about preserving for future generations the source codes of the past that made the history of Computer Science and the Information Society, but also, and above all, we are trying to build a very large telescope that will allow us to explore the present evolution of the software development galaxy, in order to better understand it, and to improve it, in order to build a better technological future.

5 A Peek Under the Hood

In Fig. 2 one can find a high level overview of the architecture of the Software Heritage crawler, and of the key data structure that is used to store all the source code with its development history. Software development and distribution platforms abound, and there is no single standard protocol to interact with them, so Software Heritage is building, with the help of a growing community, a large set of connectors, called *listers*, to identify the available software projects. There is no single standard for keeping track of versions of software either, nor for distributing packaged software, so we are also building a large set of converters, called *loaders*, that are able to encode all these different formats in a single, universal data structure, that is based on the Merkle tree construction [4]. This uniform data structure contains today over 30 billion nodes and 350 billion edges, one of the largest public graphs available.

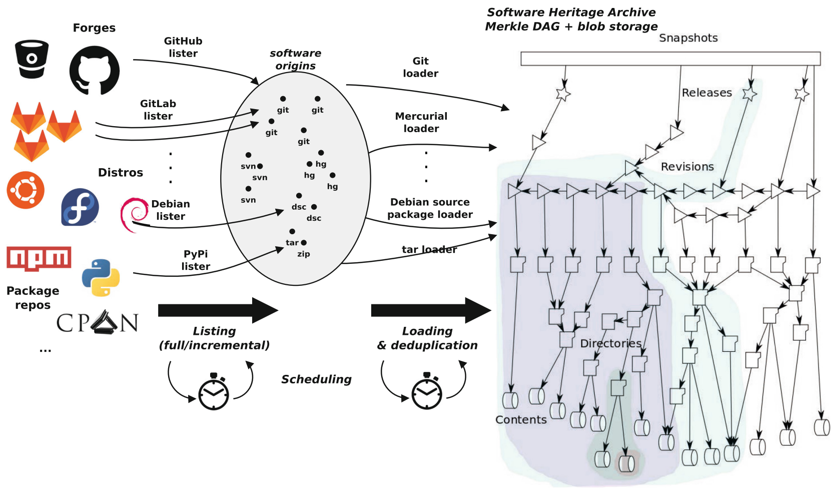

Fig. 2. The Software Heritage architecture: crawling and data structure

In this giant Merkle graph, every software artefact is identified by a cryptographic intrinsic persistent identifier, called SWHID, that can be used to check integrity of the artifact without relying on any central registry. A key side effect is the fact that duplicate artifacts can be immediately spotted (they have the same identifier!), and the Software Heritage archive only keeps one copy of each of them, while keeping track of all their occurrences.

6 Beyond Preservation, A Strategic Issue

The Software Heritage archive is already the most important source code collection in the world, but there is still a long way to go: a wide range of players, from cultural heritage to industry, from research to public administration, must be brought together around it. To do this, we are counting on a growing network of ambassadors (see https://softwareheritage.org/ambassadors/). It is clear that software has become an essential component of all human activity, and therefore unrestricted access to publicly available software source codes is becoming a digital sovereignty issue for all nations. The unique infrastructure that Software Heritage is building, and its universal approach, is an essential element to meet this challenge of digital sovereignty while preserving the common good dimension of the archive. It is therefore of the utmost importance that institutional, industrial, academic and civil society actors grasp the importance of these issues, providing the necessary resources to make Software Heritage grow and last, by taking their place alongside other international actors who are already committed, and by supporting the creation of an international non-profit institution that will carry out this mission over the long term.

7 For More Information

You can learn more about the project by visiting www.softwareheritage.org, annex.softwareheritage.org and docs.softwareheritage.org. It is possible to easily explore the source codes contained in Software Heritage on archive.softwareheritage.org.

References

1. Abramatic, J.-F., Di Cosmo, R., Zacchiroli, S.: Building the universal archive of source code. Commun. ACM **61**(10), 29–31 (2018)
2. Di Cosmo, R.: Archiving and referencing source code with software heritage. In: Bigatti, A.M., Carette, J., Davenport, J.H., Joswig, M., de Wolff, T. (eds.) ICMS 2020. LNCS, vol. 12097, pp. 362–373. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-52200-1_36
3. Di Cosmo, R., Gruenpeter, M., Zacchiroli, S.: Referencing source code artifacts: a separate concern in software citation. Comput. Sci. Eng. **22**(2), 33–43 (2020)
4. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 369–378. Springer, Heidelberg (1988). https://doi.org/10.1007/3-540-48184-2_32
5. Shustek, L.J.: What should we collect to preserve the history of software? IEEE Ann. Hist. Comput. **28**(4), 110–112 (2006)