Saving the Software Heritage: the process

Extended Abstract

Laura Bussi
Dept. of Computer Science
University of Pisa
l.bussi1@studenti.unipi.it

Roberto Di Cosmo
Software Heritage, Inria and University of Paris
roberto@dicosmo.org

Carlo Montangero
Dept. of Computer Science
University of Pisa
carlo@montangero.eu

Guido Scatena
Dept. of Computer Science
University of Pisa
guido.scatena@unipi.it

May 10, 2019

Software is everywhere, binding our personal and social lives, embodying a vast part of the technological knowledge that powers our industry, supports modern research, mediates access to digital content and fuels innovation. In a word, a rapidly increasing part of our collective knowledge is embodied in, or dependent on software artifacts.

Software does not come out of the blue: it is written by humans, in the form of software Source Code (SC), is a precious, unique form of knowledge that, besides being readily translated into machine executable form, should also "'be written for humans to read"' [1], and "'provides a view into the mind of the designer"' [7]. It is essential to preserve this precious technical, scientific and cultural heritage over the long term.

Software Heritage is a non profit, multi-stakeholder initiative, launched by Inria in partnership with Unesco, that has taken over this challenge. Its stated mission is to collect, preserve, and make readily accessible all the software source code ever written, in the Software Heritage *Archive*.

Software Heritage designed specific strategies to collect software according to its nature [2].

For software that is easily accessible online, and that can be copied without specific legal authorizations, the approach is based on automation. This way, as of May 2019, Software Heritage has already archived almost 6 billion unique SC files from over 89 million different origins, focusing in priority on popular

software development platforms like GitHub and GitLab and rescuing SC from Google Code and Gitorious that hosted more than 1.5 million projects, and now shutdown.

For source code that is not easily accessible online or requires curation, we take a different approach, based on a focused search, and with significant human intervention. The DIPISASWHinitiative is the first major step in this direction: its goal is to design and evaluate a process, that we call Software Heritage Acquisition Process (SWHAP), to *rescue, curate and illustrate* landmark legacy SC. It needs to cope with the variety of physical media where the source code may be stored, the multiple copies and versions that may be available, the potential input of the authors that are still alive, and the existence of ancillary material like documentation, articles, books, technical reports, email exchanges.

Our purpose is to present the approach we are taking with the definition, pilot implementation and experimental usage of SWHAP, in order to get feedback from the HaPoC community on our goals and methods.

Requirements First of all, we identified the following high-level requirements, for the activities of *preservation, curation and illustration.*

Preservation of the artifacts is a primary concern: we need a place where to save the original raw material that will be later studied and manipulated. Depending on the nature of the raw material, these places may be physical or virtual, as follows:

Warehouse : a physical location where physical raw material is safely archived and stored, with the usual acquisition process (for example, see [6])

Depository : a virtual space where is safely archived digital raw materials, either obtained directly in digital form, or extracted from physical supports; a proper acquisition process must be put in place to ensure that this material is traceable.

Curation of the raw digital material found in the Depository will produce clean and historified SC that can be ingested in the SWH Archive, and ancillary material that may be deposited in locations like WikiData.

Workbench The curation process for source code can be complex, as can be seen in the great example of the history of Unix [8], and we recommend to have a standard environment where it can be carried out, with support for logging the various operations.

Illustration of the history of relevant source code may require access to all of the above locations: the Warehouse, the Depository, the Workbench, the Software Heritage Archive, WikiData, as well as to other external resources.

We also agreed on the following supporting principles

Convergence The activities related to offline (legacy) SC should converge as early as possible to those related to online SC.

Openness Any supporting implementation should be based on open and free tools and standards.

Interoperability Any supporting implementation should provide support for the cooperation and coordination of the many actors playing the many roles of the acquisition process.

We are currently experimenting with a particular implementation of the process where the two virtual locations, Depository and Workbench, use GitHub as a common technological backbone.

GitHub is a platform that offers not only a Version Control System (VCS), extremely useful for tracking various version of digital artifacts, but also a wealth of tools to facilitate collaboration among a variety of users that seems very well suited for supporting the multi-users acquisition process we aim at.

Moreover, GitHub is already automatically archived by Software Heritage, which means that all the artifacts stored in the Depository and the steps performed in the Workbench will be safely archived alongside the results of the curation process: we believe this will facilitate convergence.

To facilitate the access to the artifacts, we need to adopt a standard ontology, and for this we are evaluating the CodeMeta Project [5], that provides a standard vocabulary for describing software artifacts.
To archive the ancillary documents, Wikimedia [9] is a natural choice, given its openness and freeness.

Over the next months, we will run the process on a representative sample of SC developed in Pisa, including i) a representative offline digital software of the '90s, the CMM system [3], developed using a in house VCS, and ii) a representative offline software of the '70s, available only as paper listing, the TAUMUS system for music generation [4].

We hope to demonstrate the feasibility of a process that, leveraging the powerful infrastructures available today, ranging from Software Heritage to Wikimedia, through GitHub, will be easy to be adopted worldwide by all the passionate people and institutions that want to contribute to rebuilding the history of computer programming.

References

[1] Harold Abelson and Gerald J. Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, 1985.

[2] Jean-François Abramatic, Roberto Di Cosmo, and Stefano Zacchiroli. "Building the Universal Archive of Source Code." In: *Commun. ACM* 61.10 (Sept. 2018), pp. 29–31. ISSN: 0001-0782. DOI: 10.1145/3183558.

[3] G. Attardi and T. Flagella. "A customisable memory management framework." In: *Proc. of USENIX C++ Conference 1994*. (Cambridge, MA). 1994, pp. 123–142.

[4] G. Bertini, M. Chimenti, and F. Denoth. "TAU2 – An Audio Terminal for Computer Music Experiments." In: *Int. Symposium on Technology for Selective Dissemination of Information*. (San Marino). New York, NY: IEEE Computer Society, 1976, pp. 143–149.

[5] CodeMeta. *The CodeMeta Project*. https://codemeta.github.io/index.html.

[6] Edited Gordon Mckenna, Efthymia Patsatzi, and Cb Jp. *The UK Museum Documentation Standard SPECTRUM*. 2009.

[7] L. Shustek. "What Should We Collect to Preserve the History of Software?" In: *IEEE Annals of the History of Computing* 28.04 (Oct. 2006), pp. 110–112.

[8] Diomidis Spinellis. "A Repository of Unix History and Evolution." In: *Empirical Software Engineering* 22.3 (2017), pp. 1372–1404. DOI: 10.1007/s10664-016-9445-5.

[9] Wikimedia. *The Wikemedia Foundation*. https://wikimediafoundation.org/.