Building the Universal Archive of Source Code

A global collaborative project for the benefit of all

By Jean-François Abramatic, Roberto Di Cosmo, Stefano Zacchiroli

Software is becoming the fabric that binds our personal and social lives, embodying a vast part of the technological knowledge that powers our industry, and fuels innovation. Software is a pillar of most scientific research activities in all fields, from mathematics to physics, from chemistry to biology, from finance to social sciences. Software is also an essential mediator for accessing any digital information.

In short, a rapidly increasing part of our collective knowledge is embodied in, or dependent on software artifacts. Our ability to design, use, understand, adapt, and evolve systems and devices on which our lives have come to depend relies on our ability to understand, adapt, and evolve the *source code of the software* that controls them.

Software source code is a precious, unique form of knowledge. It can be readily translated into a form executable by a machine, and yet it is *human readable*: Harold Abelson wrote *"Programs must be written for humans to read",*[1] and source code is the preferred form for modification of software artefacts by developers.[2] Quite differently from other forms of knowledge, we have grown accustomed to use version control systems that *trace source code development*, and provide precious insight on its evolution. As Len Shustek puts it, *"Source code provides a view into the mind of the designer"*.[3]

And yet, we have not been taking good care of this precious form of knowledge.

Source code is spread around a variety of platforms and infrastructures that we use to develop and/or distribute it, and software projects often migrate from one to another: there is no *universal catalog* that tracks it all.

Software can be deleted, corrupted or misplaced. What's even more worrying, in recent years we have seen major code forges shut down, endangering hundreds of thousands of publicly available software projects *at once.*[4]

We clearly need a *universal archive* of software source code.

The deep penetration of software in all aspects of our world brings along failures and risks whose potential impact is growing. Users now understand the need for an organized attention to software safety, security, reliability, and traceability. But unlike other scientific fields, we lack *large scale research instruments* for enabling massive analysis of all the available software source code.

1 Preface to Abelson, Sussman, and Sussman, "The Structure and Interpretation of Computer Programs", MIT Press, 1985
2 Free Software Foundation, Inc., "The GNU General Public License, Version 3", §1, 2007
3 Shustek, L. J. "What Should We Collect to Preserve the History of Software?", IEEE Annals of the History of Computing, 2006
4 Squire, M. "The Lives and Deaths of Open Source Code Forges", OpenSym 2017

As computer scientists and professionals, it is our duty, our responsibility, and our privilege, to build a *shared infrastructure* that answers these needs. Not just for our community, not just for the technical and scientific community, but *for society as a whole*.

Software Heritage[5] is an initiative launched at Inria precisely to take up this mission. While a full article detailing our approach is available online[6], we focus here on the challenges raised by the three main goals: *collecting, preserving, and sharing* the source code of all the software ever written.

Collection

There are various kinds of source code. Some is *current*, actively developed and technically easy to make available, some other is *legacy* that must be painfully digged out from offline media. Some is *open*, and *free* for all to read and reuse, some other is *closed* behind proprietary doors. Software Heritage's ambition is to collect it all.

For *current, open source code*, we need an *automated* process to harvest *all* software projects, with *all* the available development history, from the many places where development and distribution take place, like forges and package repositories. Yes, we really mean harvesting everything available, with no *a priori* filtering. Because the value of an active software project will only be known in the future, and because storing all present and future source code can be done at a reasonable cost.

The technical challenge is to build crawlers for each code hosting platform, as there is *no common protocol* available, and to develop adapters for all version control systems and package formats. It is a significant undertaking, but once a standard platform is available each of these crawlers and adapters *can be developed in parallel*.

For *legacy, open source code*, we need a *crowdsourcing* platform to empower the volunteers that are willing to help recover their preferred software artefacts. Guidelines must be offered to help properly reconstruct from the raw material the interesting history that lies behind it, like in the beautiful work that has been done for the history of Unix.[7]

Closed software contains precious knowledge that is more difficult to recover. For example, the Computer History Museum[8] and Living Computers[9] have shown, in the case of the mythical Alto system[10], that once the business need to keep software closed fades away, a *focused search* (that requires a costly and dedicated effort) can succeed in recovering and liberating its source code, growing our *software commons*.

5 Software Heritage, https://www.softwareheritage.org
6 Di Cosmo, R. and Zacchiroli, S. "Software Heritage: Why and How to Preserve Software Source Code", iPRES 2017
7 Spinellis, D. "A repository of Unix history and evolution". Empirical Software Engineering, 2017.
8 Computer History Museum, http://www.computerhistory.org/
9 Living Computers: Museum + Labs, http://www.livingcomputers.org/
10 See http://xeroxalto.computerhistory.org and http://www.livingcomputers.org/Discover/News/ContrAlto-A-Xerox-Alto-Emulator.aspx.

Finally, by providing a means to safely keep closed source software under *embargo*, much like what happens already with software escrow, we may succeed in collecting current and future closed source, and be ready to liberate it when time comes, dispensing altogether with costly technical recovery efforts.

Preservation

In the extensive literature on digital preservation, it is now well established that long term preservation requires full access to the source code of the tools used for the task. Software Heritage uses and develops exclusively free and open source software tools for building its archive.

Also, *replication* and *diversification* are best practices to mitigate the threats, from technical failures to legal and economic decisions, that endanger any long-term preservation initiative. Hence we want to foster a geographically distributed network of mirrors, implemented using a variety of storage technologies, in different administrative domains, controlled by a plurality of institutions, and located in different jurisdictions.

Finally, preserving *software source code* requires preserving also the *development history* of source code, that carries precious insights on the structure of programs and also track inter-project relationships. Software Heritage's unique approach is to store all available source code and its revisions into *a single* Merkle DAG (Directed Acyclic Graph), *shared among all software projects*. This data structure facilitates distribution and enables full deduplication (massively reducing storage costs), integrity checking and tracking of reuse across all software projects at the file level. But it also poses novel challenges when it comes to efficiently indexing and querying its contents.

Sharing

The raw material that Software Heritage collects must be properly organised to ease its fruition. On top of the information captured by version control systems, we need metadata describing the software and means to classify the millions of harvested projects, written in one of the thousands known programming languages[11]. We need to extract and reconcile existing information from many different sources, encoded in one of the many different software ontologies, and complete it with using either automatic tools or crowdsourcing.

We must also support the many use cases that it enables. Programmers may want to search for specific project versions or code snippets to reuse, and then *browse* them online or *download* history-full source code bundles. Companies may want to *access an API* to build applications that use the archive. Researchers may want to *access the whole corpus* to perform big data operations or train machine learning models.

We must carefully assess which functionalities are generic enough to be incorporated in the archive, and which are so specific that they are best implemented externally by third parties. And there are of course legal and ethical issues to be dealt with when redistributing parts, or all, of the contents of the archive.

11 See http://hopl.info/

Current status

Software Heritage is an active project that has already assembled the largest existing collection of software source code. At the time of writing the Software Heritage Archive contains more than 4 billion unique source code files and one billion individual commits, gathered from more than 80 million publicly available source code repositories (including a full and up-to-date mirror of GitHub) and packages (including a full and up-to-date mirror of Debian). Three copies are currently maintained, including one on a public cloud.

As a graph, the Merkle DAG underpinning the archive consists of 7 billion nodes and 60 billion edges; in terms of resources, the compressed and fully deduplicated archive requires some 200 TB of storage space. These figures grow constantly, as the archive is kept up to date by periodically crawling major code hosting sites and software distributions, adding new software artifacts, but never removing anything.

The contents of the archive can already be browsed online, or navigated via a REST API[12].

Next steps

We are at a unique turning point in the history of computer science and technology.

Looking at the past, we see many important pieces of historical software that are lost, misplaced or behind barriers. On the other hand, many of our founding fathers are still here. They have the knowledge and the will to share what is necessary to rebuild the full history of our discipline, a *unique opportunity* that no other field of science or technology has ever been offered.

Looking at the future, we see that software development is skyrocketing. It is urgent to build the missing infrastructure and put in place the good practices that are necessary to make sure our entire software commons will be properly collected and preserved. Every year that goes by without acting increases significantly the backlog.

By launching Software Heritage, Inria has done the initial effort, creating the archive infrastructure, establishing an agreement with UNESCO, and assembling an initial group of supporters[13] and committed sponsors, including Microsoft, Intel, Société Générale, Huawei, Google GitHub, Qwant, Nokia Bell Labs, DANS, FossID, UQAM and the University of Bologna. Now we need to move forward, and grow Software Heritage into an international common infrastructure.

Four ingredients are key to the success of our mission: raising awareness of the importance of source code as a first class citizen in our cultural heritage, gathering the resources needed to create the infrastructure, and leveraging the expertise from many fields of our discipline, building on a community that shares the vision.

12 https://archive.softwareheritage.org/
13 https://www.softwareheritage.org/support/testimonials/

As an open initiative, Software Heritage strives to act as a host and a catalyzer for this community, and we are now calling for contributors to join forces and tackle the issues highlighted in this article, and the many others that will arise along the way.

Let's recall here a few.

- For the collection phase, we need help recovering important software from the paste and building adaptors for the many hosting platforms and source code distribution formats.

- For the preservation phase, we need resources to host mirrors, as well as contributors willing to try different technologies for storing and mirroring the archive.

- For the sharing phase, help is needed to organize the contents, to build efficient indexing and querying mechanisms, and to develop applications for specific domains.

We, technologists, engineers, scientists, and IT professionals have a noble mission and a grand challenge: let's work together to deliver on it.