United Nations
Educational, Scientific and
Cultural Organization

Memory of
the World

Università di Pisa

Inria
inventors for the digital world

Version 1.1

The Software Heritage Acquisition Process

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

List of the authors

Short title	SWHAP
Full title	Software Heritage Acquisition Process
Authors:	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⟩
Date	October 6, 2025
Contact	Roberto Di Cosmo ⟨roberto@dicosmo.org⟩

Abstract

The source code of landmark legacy software is particularly important: it sheds insights in the history of the evolution of a technology that has changed the world, and tells a story of the humans that dedicated their lives to it.
Rescuing it is urgent, collecting and curating it is a complex task that requires significant human intervention.
This document presents the first version of SWHAP, the Software Heritage Acquisition Process: a protocol for the collection and preservation of software of historical and scientific relevance. SWHAP results from a fruitful collaboration of the University of Pisa with Software Heritage in this area of research, under the auspices of UNESCO, and has been validated on a selection of software source code produced in the Pisa area over the past 50 years.

Acknowledgments L. Bussi wants to acknowledge the Software Heritage Foundation for the scholarship that supported her work and the Department of Computer Science of the University of Pisa for hosting her while working on SWHAPPE.

Contents

1) Introduction

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 depends on software artifacts.

Software does not come out of the blue: it is written by humans, in the form of software Source Code, a precious, unique form of knowledge that, besides being readily translated into machine-executable form, should also "be written for humans to read" (Abelson and Julie Sussman [1]), and "provides a view into the mind of the designer" (Shustek [6]).

As stated in the Paris Call on Software Source code as Heritage for sustainable development (Report [5]), from the UNESCO-Inria expert group meeting, 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. To this end, Software Heritage designed specific strategies to collect software according to its nature (Abramatic, Di Cosmo, and Zacchiroli [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 September 2019, Software Heritage has already archived more than 6 billion unique source code files from over 90 million different origins, focusing in priority on popular software development platforms like GitHub and GitLab and rescuing software source code from legacy platforms, such as Google Code and Gitorious that once hosted more than 1.5 million projects.

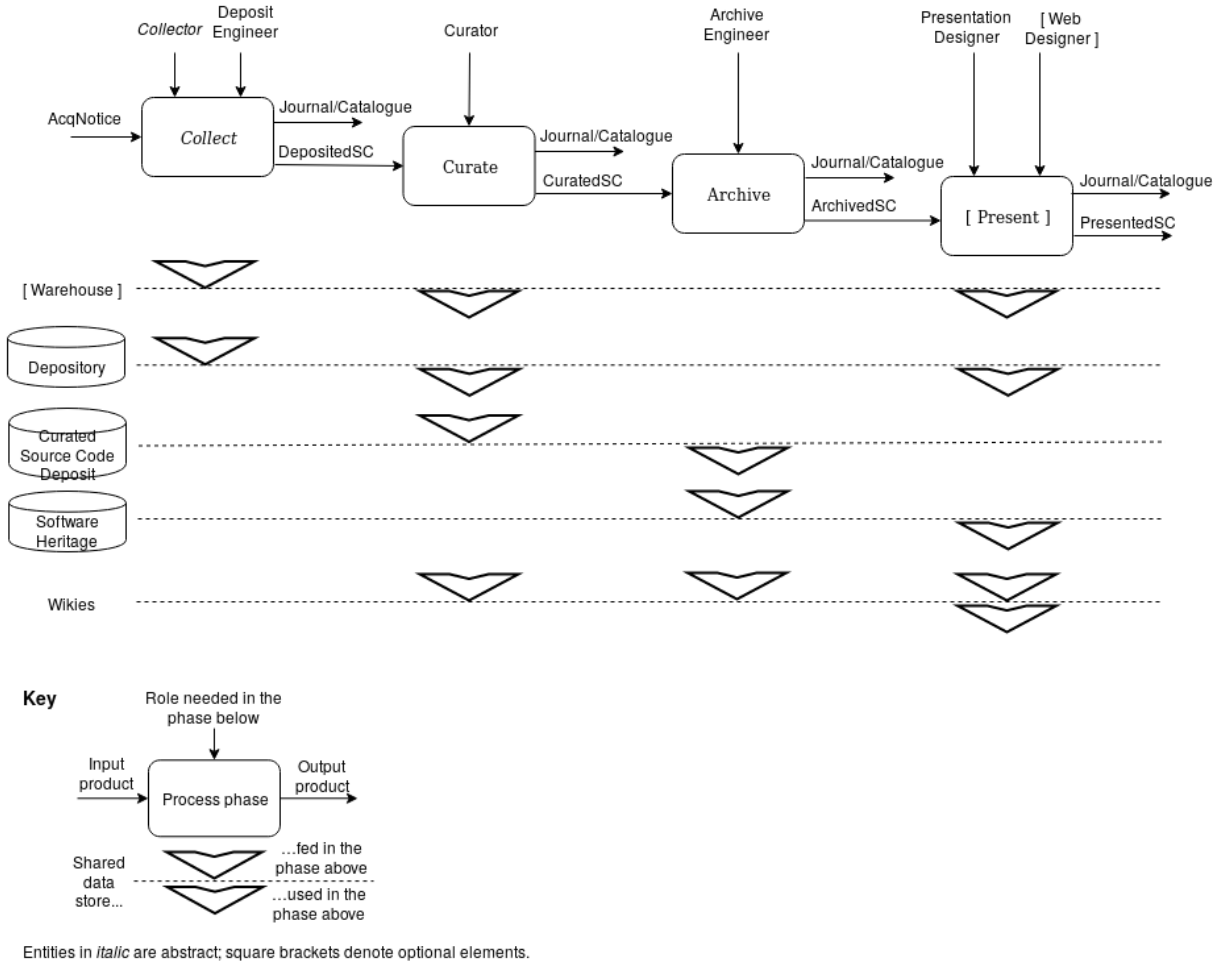
For source code that is not easily accessible online, a different approach is needed. It is necessary to cope with the variety of physical media where the source code may be stored, the multiple copies and versions that may be available, the potential input of the authors that are still alive, and the existence of ancillary materials like documentation, articles, books, technical reports, email exchanges. Such an approach shall be based on a focused search, involving a significant amount of human intervention, as demonstrated by the pioneering works reconstructing the history of Unix (Spinellis [7]) and the source code of the Apollo Guidance Computer (Burkey [4]).

This document presents the first version of SWHAP, the ***SoftWare Heritage Acquisition Process*** to rescue, curate and illustrate landmark legacy software source code, a joint initiative of Software Heritage and the University of Pisa, in collaboration with UNESCO.

Section {**sec:processabsempty citation**} provides an *abstract* view of SWHAP, its steps, documents and resources. No specific assumptions on the *tools, platforms and technologies* that may be used to enact it are made, but some requirements are made explicit. Section {**sec:processconcreteempty citation**} describes how the abstract process is implemented at the University of Pisa by leveraging the Git toolset and the GitHub collaborative development platform. This implementation is named SWHAPPE (SWH Acquisition Process Pisa Enactor) in this document. Finally, Section {**sec:walkthroughempty citation**} provides a walkthrough on an annotated example, using a real world medium-sized software project (Attardi and Flagella [3]).

2) The process, abstract view

This section describes SWHAP, the acquisition process for software artifacts at an *abstract* level, that is, without making specific assumptions on the *tools, platforms and technologies* that may be used to perform the various operations described here.



[

2.1) Phases

The activities involved in the acquisition process can be organized in the following four phases, of which the first one is *conservative*, i.e., it is devoted to save the raw materials that the other phases will build upon.

Figure {**fig:absprocessempty citation**} provides a pictorial view of the process, its phases, data stores and roles.

Collect

The purpose of this phase is *to find* the source code and related materials and *gather* it *as is* in a physical and/or logical place where it can be properly *archived* for later processing.

Various *strategies* are possible for collecting the raw materials: a dedicated team may proactively search for the artifact of specific software that has been identified as relevant (*pull approach*), or a crowdsourcing process may be set up to allow interested parties to submit software that has not been previously identified (*push approach*).

Source code can be provided in a *digital* or *physical* form. Typically, source code for old machines (such as the first Italian computer, CEP, now exposed in the Pisa museum of computing) is available only as paper printouts that may even include hand-written comments: all these materials deserve to be preserved.

Related materials can include research articles, pictures, drawings, user manuals: all of these are part of the software history and need to be preserved as well as the source code.

At this stage of elaboration of the process, this phase is better thought of as *abstract*, in the sense that several, more focussed descriptions should be provided to cater for the different situations identified. The same applies to the Curator role, which may need different capabilities in different scenarios.

Curate

The purpose of this phase is *to analyze, cleanup and structure* the raw materials that have been collected.

Preparing software source code for archival in **Software Heritage** requires special care: the source code needs to be *cleaned up*, different *versions* with their *production dates need to be ascertained*, and the *contributors need to be identified* in order to build a *faithful history of the evolution* of the software over time.

Also, proper *metadata* should be created and made available alongside the source code, providing all the key information about the software that is discovered during the curation phase. We recommend to use the vocabulary provided by **CodeMeta** as an extension to schema.org (see https://codemeta.github.io/terms/); this includes the software runtime platform, programming languages, authors, license, etc.

Particular care is required to *identify the owners* of the different artifacts, and *obtain if needed the necessary authorizations* to make these artifacts publicly available[1].

Archive

The purpose of this phase is to contribute the curated materials to the infrastructures specialized for each kind of materials: ***Software Heritage*** for the *source code*, ***Wikimedia*** for *images or videos*, ***open access repositories*** for *research articles*, ***Wikidata*** for *software descriptions and properties*, and so on.

Well established guidelines are available for contributing materials to Wikimedia (see https://commons.wikimedia.org/wiki/Co
and Wikidata (see https://www.wikidata.org/wiki/Wikidata:Data_donation), hence we will focus primarily on curating and contributing the software source code to Software Heritage, a process that is new and may require rather technical steps.

Present

The purpose of this phase is to create dedicated presentations of the curated materials.

Once the curated materials are made available in the dedicated infrastructures, it is possible to use it to create presentations for a variety of purposes: special events, virtual or physical expositions for museums or websites.

For this, the archived materials need to be referenced using the identifiers that each platform provides for its contents. Software Heritage provides intrinsic persistent identifiers that are fully documented at https://docs.softwareheritage.org/devel/swh-model/persistent-identifiers.html

The presentation phase is out of the scope of this document, and as such we are currently not providing a supporting implementation. Anyway, a good example of what can be done is the https://sciencestories.io

[1] This is a complex issue, that may need to be handled according to country-specific regulations and is out of the scope of the present document. In the United States, one may leverage the "fair use" doctrine, see for example the detailed analysis presented in https://www.softwarepreservationnetwork.org/bp-fair-use/

website.

2.2) An iterative process

New information may arise at any time: new raw materials may be discovered, refined information may be identified that needs to be added to the curation, and mistakes may need to be corrected. Hence, the overall process must be seen as *iterative*, in the sense that, when new data are available, the pertinent phase can be re-entered and the process enacted once more from there to update all the relevant information. This suggests that, whenever possible, the data stores should be fully versionable, not to loose historical information about the acquisition process itself.

2.3) Resources needed by the process

As any process supported digitally, SWHAP needs both human and technical resources to be enacted.

First of all, several data stores and working areas are needed, to save and make public the intermediate products, which are themselves of value, as already mentioned, and to pass the collected information across the phases. These are shown in the lower part of Figure 1, and are summarized here.

Warehouse

A physical location where physical raw materials are safely archived and stored, with the usual acquisition process[2].

Depository

A virtual space where digital raw materials are safely archived. The raw digital materials found in the Depository are used in the Curation phase to produce the source code that Software Heritage can ingest in the Archive phase.

The Depository holds also the related raw materials that may be elaborated and deposited in locations like WikiData, WikiMedia etc. - referred to as **Wikies** in fig. 1 - in the other phases.

Workbench

Any implementation of the process will need a virtual space and working environment where the activities can be carried out, with support for temporary storage and for logging the various operations in a journal.

Curated source code deposit

A fully versioned repository, holding the reconstructed development history of the source code, in view of its transfer to Software Heritage.

Catalogues and journals

As shown in fig. 1, according to the best practices of the archival sciences, each phase shall produce both a *Catalogue* of its products and a *Journal* recording its activities - *who did what, and when.* A list of the *Actors* involved in the process is also necessary. Provision to store all these information safely has to be foreseen in any supporting implementation.

[2]See for example https://collectionstrust.org.uk/spectrum/.

2.4) Roles in the process

With respect to the human resources, several roles are needed to enact the process, as indicated in the top part of fig. 1. Here is a short summary of the involved capabilities.

Collector

Searches and receives the raw materials. Identifies, classifies and separates source code and ancillary materials.

Deposit engineer

Masters the procedures to archive physical and digital materials, in the local context.

Curator

Prepares the version history, identifying the authors and other contributors. Provides a context to the source code, choosing among the ancillary materials.

Archive engineer

Masters the procedures to transfer the curated source code to SWH and to publish the context in the Wikies.

Presentation designer and Web engineer

These are out of the scope of this document, and are mentioned only to note that, though most of the presentations of the archived software will be on line, the abilities to design the contents of a presentation should be considered separately from the technical ones.

Remark the technical resources described above in abstract terms, may be implemented in a variety of ways. For example, one can imagine a single Depository for all the software projects that are collected, but it is also possible to use a separate Depository for each software project, and the same holds for all the other areas.

Remark the roles indicated above need not necessarily be played by different persons, e.g., Collector and Curator may be the same person, nor be played by a unique person, e.g., there can be several cooperating Curators, in case of large systems.

2.5) Implementation requirements

The abstract process may be implemented using different tools, platforms and technologies, as long as the following key requirements are satisfied.

Long term availability

The places where the artefact (both raw and curated) are stored must provide sufficient guarantees of availability over the long term. These places may be physical (warehouses), or logical (depositories).

Historical accuracy

Any supporting implementation should support the faithful recording of the authorship of the source code as well as of the reconstruction process, e.g., via a flexible versioning system.

Traceability

It must be possible to trace the origin of each of the artifacts that are collected, curated and deposited. For physical materials, we refer to common practice[3]. For digital artifacts, it is recommended to keep a *journal of all the operations* that are performed, and to automate them as much as possible, as the collection and curation process may require several iterations.

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.

[3]See for example in https://collectionstrust.org.uk/spectrum/.

3) The process, a concrete view

In order to implement SWHAP, the first step is to decide how to instantiate the needed storage and working areas: Warehouse, Depository, Curated source code deposit and Workbench.

The Warehouse is quite similar to the usual storage area where museums preserve their collections; it will need to be set up in a specific physical location, following the well established process for museums, so we will not cover it in this guide.

The other areas, which are virtual spaces, can very well be set up using distinct digital platforms, but it is also possible to instantiate all of them on a single platform.

Guidelines for effectively implement the process on different platforms. e.g. GitHub, GitLab.com or GitLab instances, etc. Will be progressively released.

They can be found in the documents with names starting with the `SwhapGuide–` prefix.

Bibliography

[1] Harold Abelson and Gerald J. Sussman with Julie Sussman. *Structure and Interpretation of Computer Programs*. Cambridge, MA: The MIT Press and McGraw-Hill, 1985, pp. xx + 542. ISBN: 0-262-01077-1 (MIT Press), 0-07-000422-6 (McGraw-Hill).

[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. URL: http://doi.acm.org/10.1145/3183558.

[3] Giuseppe Attardi and Tito Flagella. "Memory Management in the PoSSo Solver". In: *J. Symb. Comput.* 21.3 (1996), pp. 293–311. DOI: 10.1006/jsco.1996.0013. URL: https://doi.org/10.1006/jsco.1996.0013.

[4] Ronald Burkey. *Virtual AGC - Changelog*. Available at http://ibiblio.org/apollo/changes.html. Spans years 2003 to 2019.

[5] Expert Group Report. *Paris Call: Software Source Code as Heritage for Sustainable Development*. Available from https://unesdoc.unesco.org/ark:/48223/pf0000366715. 2019.

[6] Leonard J. Shustek. "What Should We Collect to Preserve the History of Software?" In: *IEEE Annals of the History of Computing* 28.4 (2006), pp. 110–112. DOI: 10.1109/MAHC.2006.78. URL: http://dx.doi.org/10.1109/MAHC.2006.78.

[7] 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. URL: https://doi.org/10.1007/s10664-016-9445-5.