EOC-Pillar UC 6.4: Software Heritage
Software source code preservation, reference and access.

David Douard
Software engineer
Inria, Software Heritage
david.douard@softwareheritage.org

September 30th, 2021
1. Introduction
2. Software Heritage: the universal source code archive
3. SWHID: Software Heritage Persistent Identifiers
4. Archiving source code in the Software Heritage Archive
5. Retrieving Source Code from the Archive
6. Use Case Status
7. Conclusion
Leveraging the experience of Software Heritage, this task aimed at design and pilot a solution for the preservation of massive collections of software source code (billions of files with links to publications) into EOSC eTDR service (European Trusted Digital Repository). More specifically, this task will:

1. standardize a persistent identifier schema (PID) for referencing billions of archived software artifacts,
2. develop a deposit API for research software archival,
3. develop an access API for retrieval of archived software,
4. provide access and help EOSC partners to integrate the above APIs and Services,
5. develop a pilot to fully store the software archive onto existing EOSC infrastructure.
Main Objectives

- Collect  Save all the (open source) software source code
- Preserve  Keep all the (archived) software source code forever
- Share  Make every piece of source code identifiable and freely available
Software is all around EOSC-Pillar

Open Science cannot exist without:
- open source software
- accessible source code
- identifiable source code
- FAIR source code

Source is everywhere in EOSC:
- data production
- data processing
- HPC / compute as a service

David Douard
www.softwareheritage.org September 30th, 2021 4 / 41
Software is all around EOSC-Pillar

Open Science cannot exist without:
- open source software
- accessible source code
- identifiable source code
- FAIR source code

Source is everywhere in EOSC:
- data production
- data processing
- HPC / compute as a service

Is every piece of this software clearly identified?
Software is all around EOSC-Pillar

Open Science cannot exist without:
- open source software
- accessible source code
- identifiable source code
- FAIR source code

Source is everywhere in EOSC:
- data production
- data processing
- HPC / compute as a service

Is every piece of this software:
- clearly identified?
- accessible?
Software is all around EOSC-Pillar

<table>
<thead>
<tr>
<th>Open Science cannot exists without</th>
</tr>
</thead>
<tbody>
<tr>
<td>- open source software</td>
</tr>
<tr>
<td>- accessible source code</td>
</tr>
<tr>
<td>- identifiable source code</td>
</tr>
<tr>
<td>- FAIR source code</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source is everywhere in EOSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>- data production</td>
</tr>
<tr>
<td>- data processing</td>
</tr>
<tr>
<td>- HPC / compute as a service</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Is every piece of this software</th>
</tr>
</thead>
<tbody>
<tr>
<td>- clearly identified?</td>
</tr>
<tr>
<td>- accessible?</td>
</tr>
<tr>
<td>- today?</td>
</tr>
</tbody>
</table>
Software is all around EOSC-Pillar

Open Science cannot exist without:
- open source software
- accessible source code
- identifiable source code
- FAIR source code

Source is everywhere in EOSC:
- data production
- data processing
- HPC / compute as a service

Is every piece of this software:
- clearly identified?
- accessible?
- today?
- in the future?
“Software, instructions that tell a computer what to do. Software comprises the entire set of programs, procedures, and routines associated with the operation of a computer system. The term was coined to differentiate these instructions from hardware—i.e., the physical components of a computer system.”
“Software, instructions that tell a computer what to do. Software comprises the entire set of programs, procedures, and routines associated with the operation of a computer system. The term was coined to differentiate these instructions from hardware—i.e., the physical components of a computer system.”
So/f_tware definition

Encyclopædia Britannica

“Software, instructions that tell a computer what to do. Software comprises the entire set of programs, procedures, and routines associated with the operation of a computer system. The term was coined to differentiate these instructions from hardware—i.e., the physical components of a computer system.”

Software as a concept

- software project / entity
- the creators and the community around it
Encyclopædia Britannica

“Software, instructions that tell a computer what to do. Software comprises the entire set of programs, procedures, and routines associated with the operation of a computer system. The term was coined to differentiate these instructions from hardware—i.e., the physical components of a computer system.”
Software definition

Encyclopædia Britannica

“Software, instructions that tell a computer what to do. Software comprises the entire set of programs, procedures, and routines associated with the operation of a computer system. The term was coined to differentiate these instructions from hardware—i.e., the physical components of a computer system.”

Software as a concept

- software project / entity
- the creators and the community around it
- the software idea / algorithms / solutions

Software artifact

- the executable (or binary) of each version for a specific environment
“Software, instructions that tell a computer what to do. Software comprises the entire set of programs, procedures, and routines associated with the operation of a computer system. The term was coined to differentiate these instructions from hardware—i.e., the physical components of a computer system.”

Software as a concept
- software project / entity
- the creators and the community around it
- the software idea / algorithms / solutions

Software artifact
- the executable (or binary) of each version for a specific environment
- the software source code for each revision
Software is complex

- **Structure**: monolithic/composite; self-contained/external dependencies
- **Lifetime**: one-shot/long term
- **Community**: one man/one team/distributed community
- **Authorship**: complex set of roles
- **Authority**: institutions/organizations/communities/single person

Much more complex than it seems
Much more complex than it seems

Software is complex

- **Structure**: monolithic/composite; self-contained/external dependencies
- **Lifetime**: one-shot/long term
- **Community**: one man/one team/distributed community
- **Authorship**: complex set of roles
- **Authority**: institutions/organizations/communities/single person

Various granularity

- **Exact status of the source code** for reproducibility, e.g.
  “you can find at swh:1:cnt:cdf19c4487c43c76f3612557d4dc61f9131790a4;lines=146-187 the core algorithm used in this article”

  -(Major) release “This functionality is available in OCaml version 4”

  - Project “Inria has created OCaml and Scikit-Learn”
The knowledge is in the source code!

“The source code for a work means the preferred form of the work for making modifications to it.”

GPL Licence
"The source code for a work means the preferred form of the work for making modifications to it."

GPL Licence

Hello World
The knowledge is in the source code!

"The source code for a work means the preferred form of the work for making modifications to it."

---

**Hello World**

---

### Program (excerpt of binary)

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4004e6</td>
<td>55</td>
</tr>
<tr>
<td>4004e7</td>
<td>48 89 e5</td>
</tr>
<tr>
<td>4004ea</td>
<td>bf 84 05 40 00</td>
</tr>
<tr>
<td>4004ef</td>
<td>b8 00 00 00 00</td>
</tr>
<tr>
<td>4004f4</td>
<td>e8 c7 ff ff</td>
</tr>
<tr>
<td>4004f9</td>
<td>90</td>
</tr>
<tr>
<td>4004fa</td>
<td>5d</td>
</tr>
<tr>
<td>4004fb</td>
<td>c3</td>
</tr>
</tbody>
</table>
The knowledge is in the source code!

"The source code for a work means the preferred form of the work for making modifications to it."

---

**Hello World**

**Program (source code)**

```c
/* Hello World program */

#include<stdio.h>

void main()
{
    printf("Hello World");
}
```

**Program (excerpt of binary)**

4004e6: 55
4004e7: 48 89 e5
4004ea: bf 84 05 40 00
4004ef: b8 00 00 00 00
4004f4: e8 c7 fe ff ff
4004f9: 90
4004fa: 5d
4004fb: c3
Source code is special

Executable and human readable knowledge

“Programs must be written for people to read, and only incidentally for machines to execute.”

Harold Abelson
Source code is *special*

**Executable and human readable knowledge**

“Programs must be written for people to read, and only incidentally for machines to execute.”

Harold Abelson

**Software evolves over time**

- projects may last decades
- the *development history* is key to its *understanding*
Source code is special

Executable and human readable knowledge

“Programs must be written for people to read, and only incidentally for machines to execute.”
Harold Abelson

Software evolves over time

- projects may last decades
- the development history is key to its understanding

Complexity

- millions of lines of code
- large web of dependencies
  - easy to break, difficult to maintain
- sophisticated developer communities
So/f_tware Source Code human readable and executable knowledge

THE MASTER IGNITION ROUTINE IS DESIGNED FOR USE BY THE FOLLOWING LEM PROGRAMS: P12, P40, P42, P51, P63.
# IT PERFORMS all FUNCTIONS IMMEDIATELY ASSOCIATED WITH APS OR DPS IGNITION: IN PARTICULAR, EVERYTHING LYING
# BETWEEN THE PRE-IGNITION TIME CHECK -- ARE WE WITHIN 45 SECONDS OF TIG? -- AND TIG + 20 SECONDS, WHEN DPS
# PROGRAMS THROTTLE UP.
#
# VARIATIONS AMONG PROGRAMS ARE ACCOMODATED BY MEANS OF TABLES CONTAINING CONSTANTS (FOR AVEGEEXIT, FOR
# WAITLIST, FOR PINBALL) AND TCF INSTRUCTIONS. USERS PLACE THE ADDRESS OF THE HEAD OF THE APPROPRIATE TABLE
# (OF POTABLE FOR POLIM, FOR EXAMPLE) IN ERASABLE REGISTER 'WHICH' (E4). THE IGNITION ROUTINE THEN INDEXES BY
# WHICH TO OBTAIN OR EXECUTE THE PROPER TABLE ENTRY. THE IGNITION ROUTINE IS INITIATED BY A TCF BURNBABY,
# THROUGH BANKJUMP IF NECESSARY. THERE IS NO RETURN.
#
# THE MASTER IGNITION ROUTINE WAS CONCEIVED AND EXECUTED, AND (NOTA BENE) IS MAINTAINED BY ADLER AND EYLES.
#
# HONI SOIT QUI MAL Y PENSE
#
# *******************
# TABLES FOR THE IGNITION ROUTINE
#
# *******************
#
# NOLI SE TANGERE

<table>
<thead>
<tr>
<th>P12TABLE</th>
<th>VN</th>
<th>0074</th>
<th># (0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCF</td>
<td>ULLL0T</td>
<td># (1)</td>
<td></td>
</tr>
<tr>
<td>TCF</td>
<td>COMFAIL3</td>
<td># (2)</td>
<td></td>
</tr>
<tr>
<td>TCF</td>
<td>GCUT0FF</td>
<td># (3)</td>
<td></td>
</tr>
<tr>
<td>TCF</td>
<td>TASKOVER</td>
<td># (4)</td>
<td></td>
</tr>
<tr>
<td>TCF</td>
<td>P12SPOT</td>
<td># (5)</td>
<td></td>
</tr>
<tr>
<td>DEC</td>
<td>0</td>
<td># (6)</td>
<td>NO ULLAGE</td>
</tr>
<tr>
<td>EDBANK:</td>
<td>WHICH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2CAMBR:</td>
<td>SERVEXIT</td>
<td># (7)</td>
<td></td>
</tr>
<tr>
<td>TCF</td>
<td>DISCHNG</td>
<td># (11)</td>
<td></td>
</tr>
<tr>
<td>TCF</td>
<td>WAITBTT</td>
<td># (12)</td>
<td></td>
</tr>
<tr>
<td>TCF</td>
<td>P12IGN</td>
<td># (13)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P40TABLE</th>
<th>VN</th>
<th>0048</th>
<th># (0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCF</td>
<td>ULLL0T</td>
<td># (1)</td>
<td></td>
</tr>
</tbody>
</table>
Software in research has different roles

Multiple facets, it can be seen as:

- a tool
- a research outcome or result
- the object of research
Software in research has different roles

Multiple facets, it can be seen as:
- a tool
- a research outcome or result
- the object of research

By identifying the software role
we can decide how to treat it
What is at stake in increasing order of difficulty

Archival

Research software artifacts must be properly archived
make it sure we can retrieve them (reproducibility)
What is at stake in increasing order of difficulty

**Archival**
Research software artifacts must be properly **archived**
make it sure we can **retrieve** them (**reproducibility**)

**Identification**
Research software artifacts must be properly **referenced**
make it sure we can **identify** them (**reproducibility**)
### What is at stake in increasing order of difficulty

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Action</th>
<th>Reproducibility</th>
<th>Visibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archival</td>
<td>Research software artifacts must be properly archived</td>
<td>make it sure we can <strong>retrieve</strong> them</td>
<td><strong>reproducibility</strong></td>
<td></td>
</tr>
<tr>
<td>Identification</td>
<td>Research software artifacts must be properly referenced</td>
<td>make it sure we can <strong>identify</strong> them</td>
<td><strong>reproducibility</strong></td>
<td></td>
</tr>
<tr>
<td>Metadata</td>
<td>Research software artifacts must be properly described</td>
<td>make it easy to <strong>discover</strong> them</td>
<td><strong>visibility</strong></td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Archival</td>
<td>Research software artifacts must be properly archived and retrieved.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identification</td>
<td>Research software artifacts must be properly referenced and easily identified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metadata</td>
<td>Research software artifacts must be properly described for easy discovery.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Citation</td>
<td>Research software artifacts must be properly cited to give credit to authors.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

1. Introduction
2. Software Heritage: the universal source code archive
3. SWHID: Software Heritage Persistent Identifiers
4. Archiving source code in the Software Heritage Archive
5. Retrieving Source Code from the Archive
6. Use Case Status
7. Conclusion
Collect, preserve and share *all* software source code

Preserving our heritage, enabling better software and better science for all
Collect, preserve and share *all* software source code

Preserving our heritage, enabling better software and better science for all

Reference catalog

*find* and *reference* all software source code
Collect, preserve and share *all* software source code

Preserving our heritage, enabling better software and better science for all

Reference catalog

- find and reference all software source code

Universal archive

- preserve all software source code
Collect, preserve and share all software source code

Preserving our heritage, enabling better software and better science for all

Reference catalog
- find and reference all software source code

Universal archive
- preserve all software source code

Research infrastructure
- enable analysis of all software source code
Our principles

Cultural Heritage

Industry

Research

Education

Software Heritage
Software Heritage

Size

As of today the archive already contains and keeps safe for you the following amount of objects:

- **Source files**: 11,311,753,576
- **Commits**: 2,390,337,241
- **Projects**: 165,640,910
- **Directories**: 9,415,589,293
- **Authors**: 45,750,890
- **Releases**: 19,047,072
Under the hood: Automation, and storage

Global development history permanently archived in a uniform data model over 11 billion unique source files from over 160 million software projects ~650 TB (uncompressed) blobs, ~25 B nodes, ~300 B edges

David Douard
Under the hood: Automation, and storage

Software Heritage Archive
Merkle DAG + blob storage

Loading & deduplication

Scheduling

Listing (full/incremental)

Forges

GitHub lister

GitLab lister

PyPi lister

Debian lister

Distros

Package repos

CPAN

Packages

Forges

Git

Mercurial

Debian source
package loader

tar loader

Software Heritage Archive

Snapshots

Releases

Revisions

Directories

Contents

Global development history permanently archived in a uniform data model

over 11 billion unique source files from over 160 million software projects
~650 TB (uncompressed) blobs, ~25 B nodes, ~300 B edges

David Douard

www.softwareheritage.org

September 30th, 2021

14 / 41
Global development history permanently archived in a uniform data model

- over 11 billion unique source files from over 160 million software projects
- ~650 TB (uncompressed) blobs, ~25 B nodes, ~300 B edges
Merkle tree (R. C. Merkle, Crypto 1979)

Combination of
- tree
- hash function

Classical cryptographic construction
- fast, parallel signature of large data structures
- widely used (e.g., Git, blockchains, IPFS, ...)
- built-in deduplication
Typical properties of systems of identifiers

- uniqueness
- non ambiguity
- persistence
- abstraction (opacity)
Our challenges in the PID landscape

**Typical properties of systems of identifiers**

- uniqueness, non ambiguity, persistence, abstraction (opacity)

**Key needed properties from our use cases**

- **gratis** identifiers are free (billions of objects)
- **integrity** the associated object cannot be changed (sw dev, reproducibility)
- **no middle man** no central authority is needed (sw dev, reproducibility)
Our challenges in the PID landscape

Typical properties of systems of identifiers

- uniqueness, non ambiguity, persistence, abstraction (opacity)

Key needed properties from our use cases

- gratis  identifiers are free (billions of objects)
- integrity  the associated object cannot be changed (sw dev, reproducibility)
- no middle man  no central authority is needed (sw dev, reproducibility)

- we could not find systems with both integrity and no middle man!
- Intrinsic, decentralised, cryptographically strong identifiers = SWHIDs

David Douard
The SWHID schema

- **swc**: Prefix
- **cnt**: Content
- **object_id**: 41db23118f92d7218099a5e7a990cf58f1d07fa
- **schema_version**: Version
- **object_type**: Type
The SWHID schema

```
swh:1:cnt:41ddb23118f92d7218099a5e7a990cf58f1d07fa
```

- `prefix` indicates the namespace of the identifier.
- `object_id` is the unique identifier within the namespace.

- `object_type` can be:
  - "snp" - snapshot
  - "rel" - release
  - "rev" - revision
  - "dir" - directory
  - "cnt" - content

David Douard
www.softwareheritage.org  September 30th, 2021
The SWHID schema
Apollo 11 source code

P63SP0T3
CA BIT6 # IS THE LR ANTENNA IN POSITION 1 YET
EXTEND
RAND CHAN33
EXTEND
BZF P63SP0T4 # BRANCH IF ANTENNA ALREADY IN POSITION 1
CAF CODE500 # ASTRONAUT: PLEASE CRANK THE
TC BANKCALL # SILLY THING AROUND
CADR GOPERF1
TCF GOTOP00H # TERMINATE
TCF P63SP0T3 # PROCEED SEE IF HE'S LYING

P63SP0T4
TC BANKCALL # ENTER INITIALIZE LANDING RADAR
CADR SETPOS1
TC POSTJUMP # OFF TO SEE THE WIZARD ...
CADR BURNBABY
(Quick) Demo time

Apollo 11 source code

<table>
<thead>
<tr>
<th>P63SP0T3</th>
<th>CA</th>
<th>BIT6</th>
<th># IS THE LR ANTENNA IN POSITION 1 YET</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EXTEND</td>
<td>RAND</td>
<td>CHAN33</td>
</tr>
<tr>
<td></td>
<td>BZF</td>
<td>P63SP0T4</td>
<td># BRANCH IF ANTENNA ALREADY IN POSITION 1</td>
</tr>
<tr>
<td></td>
<td>CAF</td>
<td>CODE500</td>
<td># ASTRONAUT: PLEASE CRANK THE</td>
</tr>
<tr>
<td></td>
<td>TC</td>
<td>BANKCALL</td>
<td># SILLY THING AROUND</td>
</tr>
<tr>
<td></td>
<td>CADR</td>
<td>GOPERF1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TCF</td>
<td>GOTOP00H</td>
<td># TERMINATE</td>
</tr>
<tr>
<td></td>
<td>TCF</td>
<td>P63SP0T3</td>
<td># PROCEED SEE IF HE'S LYING</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P63SP0T4</th>
<th>TC</th>
<th>BANKCALL</th>
<th># ENTER INITIALIZE LANDING RADAR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CADR</td>
<td>SETPOS1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TC</td>
<td>POSTJUMP</td>
<td># OFF TO SEE THE WIZARD ...</td>
</tr>
<tr>
<td></td>
<td>CADR</td>
<td>BURNBABY</td>
<td></td>
</tr>
</tbody>
</table>

* swh:1:cnt:64582b78792cd6c2d67d35da5a11bb80886a6409
(Quick) Demo time

Apollo 11 source code

```
P63SP0T3     CA    BIT6               # IS THE LR ANTENNA IN POSITION 1 YET
              EXTEND
              RAND   CHAN33
              EXTEND
              BZF     P63SP0T4       # BRANCH IF ANTENNA ALREADY IN POSITION 1
              CAF     CODE500          # ASTRONAUT: PLEASE CRANK THE
              TC      BANKCALL         # SILLY THING AROUND
              CADR    GOPERF1
              TCF     GOTOP00H          # TERMINATE
              TCF     P63SP0T3          # PROCEED   SEE IF HE'S LYING
P63SP0T4     TC      BANKCALL       # ENTER       INITIALIZE LANDING RADAR
              CADR    SETPOS1
              TC      POSTJUMP        # OFF TO SEE THE WIZARD ...
              CADR    BURNBABY
```

* swh:1:cnt:64582b78792cd6c2d67d35da5a11bb80886a6409
  ;origin=https://github.com/virtualagc/virtualagc;lines=245-261
Outline

1. Introduction
2. Software Heritage: the universal source code archive
3. SWHID: Software Heritage Persistent Identifiers
4. Archiving source code in the Software Heritage Archive
5. Retrieving Source Code from the Archive
6. Use Case Status
7. Conclusion
Ingestion in SWH

- 3 sources of source code software ingestion

- listers and loader
- pull based
- best effort
- responsibility of SWH
- save code now
- push based
- accessible to everyone
- (light) curation from SWH
- deposit
- push based
- authenticated
- responsibility of the user/partner
- comes with metadata
Ingestion in SWH

- 3 sources of source code software ingestion

**Listers and loader**
- pull based
- best effort
- reponsibility of SWH
Ingestion in SWH

- 3 sources of source code software ingestion

<table>
<thead>
<tr>
<th>Listers and Loader</th>
<th>Save Code Now</th>
</tr>
</thead>
<tbody>
<tr>
<td>• pull based</td>
<td>• push based</td>
</tr>
<tr>
<td>• best effort</td>
<td>• accessible to everyone</td>
</tr>
<tr>
<td>• responsibility of SWH</td>
<td>• (light) curation from SWH</td>
</tr>
</tbody>
</table>
Ingestion in SWH

- 3 sources of source code software ingestion

**listers and loader**
- pull based
- best effort
- responsibility of SWH

**save code now**
- push based
- accessible to everyone
- (light) curation from SWH

**deposit**
- push based
- authenticated
- responsibility of the user/partner
- comes with metadata
Listers and Loaders

<table>
<thead>
<tr>
<th>Lister</th>
<th>Origins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitbucket</td>
<td>2,012,743</td>
</tr>
<tr>
<td>git</td>
<td>5,353</td>
</tr>
<tr>
<td>R</td>
<td>18,796</td>
</tr>
<tr>
<td>debian</td>
<td>35,970</td>
</tr>
<tr>
<td>GitHub</td>
<td>124,964,867</td>
</tr>
<tr>
<td>GitLab</td>
<td>894,322</td>
</tr>
<tr>
<td>Guix</td>
<td>10,937</td>
</tr>
<tr>
<td>GNU</td>
<td>354</td>
</tr>
<tr>
<td>heaptapod</td>
<td>982</td>
</tr>
<tr>
<td>launchpad</td>
<td>20,397</td>
</tr>
<tr>
<td>NixOS</td>
<td>30688</td>
</tr>
<tr>
<td>npm</td>
<td>1,725,269</td>
</tr>
<tr>
<td>3,516</td>
<td></td>
</tr>
<tr>
<td>SOURCEFORGE</td>
<td>341,563</td>
</tr>
</tbody>
</table>

Regular crawling
These software origins get continuously discovered and archived using the listers implemented by Software Heritage.

David Douard
Save Code Now

Save code now on https://archive.softwareheritage.org/save/

- git, svn or mercurial
- intrinsic metadata files
- complete history
- A deposited artifact is provided from one of the SWH partners which is regarded as a trusted authority.
- A deposited artifact requires metadata properties describing the source code artifact.
- A deposited artifact has a codemeta metadata entry attached to it.
- A deposited artifact has the same visibility on the SWH Archive than a collected repository.
- A deposited artifacts can be searched with its provided url property on the SWH Archive.
- The deposit API uses the SWORD v2 API, thus requires some tooling to send deposits to SWH. These tools are provided with this repository.
The deposit

Deposit software (in HAL) poster

Generic mechanism:
- SWORD based
- review process
- versioning

How to do it:
- (guide)
- deposit .zip or .tar.gz file with metadata

David Douard
www.softwareheritage.org Sepember 30th, 2021 23 / 41
The deposit

Deposit software (in HAL) poster

Generic mechanism:
- SWORD based
- review process
- versioning
The deposit

Deposit software (in HAL) poster

Generic mechanism:
- SWORD based
- review process
- versioning

How to do it: (guide)
- deposit .zip or .tar.gz file with metadata
Example: The deposit view in HAL

CGAL 3D Periodic Mesh Generation

Michael Botsanov 1, Aymeric Petit 1, Mazi Rouxel-Labbé 2, Monique Teillaud 2

1 GEOMETRICA - Geometric computing
2 CRISAM - Inria Sophia Antipolis - Méditerranée, Inria Saclay - Île de France
3 GAMBLE - Geometric Algorithms and Models Beyond the Linear and Euclidean realm

Inria Nancy Grand Est \textit{LORIA} - \textit{ALGO} - Department of Algorithms, Computation, Image and Geometry

Abstract: This package is dedicated to the generation of geometric simplicial meshes discretizing periodic 3D domains. The domain to be meshed is a region of the three-dimensional unit torus with cubical fundamental domain.

Document type: Software

BROWSE

https://hal.inria.fr/hal-02983420
Contributor: Monique Teillaud <monique.teillaud@inria.fr>
Submitted on: Thursday, October 29, 2020 - 3:57:43 PM
Last modification on: Sunday, November 1, 2020 - 2:46:1 AM

David Douard
“Ontologies are agreements, made in a social context, to accomplish some objectives. It’s important to understand those objectives, and be guided by them.”

“Ontologies are agreements, made in a social context, to accomplish some objectives. It’s important to understand those objectives, and be guided by them.”


What do we want to describe?

- a software project?
- a software artifact? a collection of artifacts?
- With what terms or vocabulary?
Software Citation Principles (Smith et al. 2016)

- **Importance**: first class citizen in the scholarly ecosystem
- **Credit and attribution**: authors, maintainer
- **Unique identification**: points to a unique, specific software version (DOI, Git SHA1 hash, etc.)
- **Persistence**: identification beyond the lifespan of the software (swh-id)
- **Accessibility**: url, publisher
- **Specificity**: version, environment
The metadata landscape

Software schemes

General schemes

- Dublin Core
- PRONOM
- Digital Preservation
- PREMIS
The metadata landscape

Software schemes

- DOAP
- Dublin Core
- PRONOM
- Digital Preservation
- PREMIS
- ADMS.SW
- catalogs / registries
- FSF directory
- swMATH
- libraries.io
- Pypi
- NPM
- Maven
- Package Management

General schemes

EOSC-Pillar
The metadata landscape

Software schemes

- catalogs / registries
- DOAP
- Dublin Core
- PRONOM
- PREMIS
- Q7397- software
  Q341- free software
- swMATH
- ADMS.SW
- Wikidata
- Digital Preservation
- Schema.org
- Linked Data
- DBPedia

General schemes

Package Management

libraries.io

NPM

Maven

Pypi

FSF directory

EOSC-Pillar

David Douard

www.softwareheritage.org

Sepember 30th, 2021
The metadata landscape

Software schemes

DOAP

Dublin Core

PRONOM

Digital Preservation

PREMIS

WikiData

SoftwareSourceCode

SoftwareApplication

Q7397- software
Q341- free software

resourceType = Software

Linked Data

Scholarly Ecosystem

Package Management

NPM

Maven

Pypi

libraries.io

swMATH

ADMS.SW

catalogs / registries

FSF directory

EOSC-Pillar

Coordination and harmonization of national & transnational initiatives to support EOSC

David Douard

www.softwareheritage.org

September 30th, 2021
1. Introduction
2. Software Heritage: the universal source code archive
3. SWHID: Software Heritage Persistent Identifiers
4. Archiving source code in the Software Heritage Archive
5. Retrieving Source Code from the Archive
6. Use Case Status
7. Conclusion
Browsing the web application

Origin: https://github.com/chrislgarry/Apollo-11

<table>
<thead>
<tr>
<th>File</th>
<th>Mode</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGC_BLOCK_TWO_SELF_CHECK.agc</td>
<td>-rw-r--r--</td>
<td>13.4 KB</td>
</tr>
<tr>
<td>ACS_INITIALIZATION.agc</td>
<td>-rw-r--r--</td>
<td>6.0 KB</td>
</tr>
<tr>
<td>ALARM_AND_ABORT.agc</td>
<td>-rw-r--r--</td>
<td>5.0 KB</td>
</tr>
<tr>
<td>AOSTASK_AND_AOSIOB.agc</td>
<td>-rw-r--r--</td>
<td>28.0 KB</td>
</tr>
<tr>
<td>AOTMARK.agc</td>
<td>-rw-r--r--</td>
<td>16.6 KB</td>
</tr>
<tr>
<td>ASCENT_GUIDANCE.agc</td>
<td>-rw-r--r--</td>
<td>10.9 KB</td>
</tr>
<tr>
<td>ASSEMBLY_AND_OPERATION_INFORMATION.agc</td>
<td>-rw-r--r--</td>
<td>31.1 KB</td>
</tr>
<tr>
<td>ATTITUDE_MANEUVER_ROUTINE.agc</td>
<td>-rw-r--r--</td>
<td>26.5 KB</td>
</tr>
<tr>
<td>BURN_BABY_BURN--MASTER_INITIALIZATION_ROUTINE.agc</td>
<td>-rw-r--r--</td>
<td>21.8 KB</td>
</tr>
<tr>
<td>CONIC_SUBROUTINES.agc</td>
<td>-rw-r--r--</td>
<td>46.5 KB</td>
</tr>
<tr>
<td>CONTROLLED_CONSTANTS.agc</td>
<td>-rw-r--r--</td>
<td>14.5 KB</td>
</tr>
<tr>
<td>DAPIDLER_PROGRAM.agc</td>
<td>-rw-r--r--</td>
<td>12.7 KB</td>
</tr>
</tbody>
</table>
Using the Vault

David Douard
source code is thoroughly deduplicated within the Software Heritage archive

bulk download of large artefacts (e.g., a Linux kernel release) requires collecting millions of objects

the **Software Heritage Vault** cooks and caches source code bundles for bulk download needs
Bulk download

- source code is thoroughly deduplicated within the Software Heritage archive.
- bulk download of large artefacts (e.g., a Linux kernel release) requires collecting millions of objects.
- the Software Heritage Vault cooks and caches source code bundles for bulk download needs.

```
$ curl -X POST /api/1/vault/revision/a86747d2.../gitfast
{
    'fetch_url': '/api/1/vault/revision/a86747d2.../gitfast/raw/',
    'progress_message': None,
    'status': 'new',
    'id': 4,
    'obj_id': 'a86747d201ab8f8657d145df4376676d5e47cf9f',
    'obj_type': 'revision_gitfast'
}
```

$ curl -O dump.gz /api/1/vault/revision/a86747d2.../gitfast/raw/
$ git init
$ zcat dump.gz | git fast-import
$ git checkout HEAD
Bulk download

- source code is thoroughly deduplicated within the Software Heritage archive
- bulk download of large artefacts (e.g., a Linux kernel release) requires collecting millions of objects
- the Software Heritage Vault cooks and caches source code bundles for bulk download needs

```bash
$ curl -X POST /api/1/vault/revision/a86747d2.../gitfast
{
    'fetch_url': '/api/1/vault/revision/a86747d2.../gitfast/raw/',
    'progress_message': None,
    'status': 'new',
    'id': 4,
    'obj_id': 'a86747d201ab8f8657d145df4376676d5e47cf9f',
    'obj_type': 'revision_gitfast'
}

$ curl -O dump.gz /api/1/vault/revision/a86747d2.../gitfast/raw/
$ git init
$ zcat dump.gz | git fast-import
$ git checkout HEAD
```
Using public APIs

freely accessible but with rate limiting

rate-limit-free access is possible with authentication
Outline

1. Introduction
2. Software Heritage: the universal source code archive
3. SWHID: Software Heritage Persistent Identifiers
4. Archiving source code in the Software Heritage Archive
5. Retrieving Source Code from the Archive
6. Use Case Status
7. Conclusion
Persistent identifiers for source code artefacts have been specified and their usage are being promoted for general adoption in scientific research for software citation.

The deposit API for the Software Heritage has been developed and is available for registered partners.

For non-registered partners, the Save Code Now feature of the Software Heritage Archive allows everyone to ask for a public repository (git, mercurial, etc.) to be ingested in the Software Heritage Archive.

Access APIs allowing to retrieve source code content from the Software Heritage Archive are available and documented.

The integration of the Software Heritage Archive in the Vitam archiving solution provided by the CINES as part of the EOC eTDR service has been started recently and is in progress.
Next Steps

- Document the Software Heritage Archive features and present them to all EOSC-Pillar partners to make sure every one is aware of the kind of service it provides and has the opportunity to integrate their tools and services with the Software Heritage Archive.
- [WIP] Finish the specification of the packages to be deposited in Vitam.
- [WIP] Implement a prototype of an automatic archiving service that consumes the (kafka) journal of the Software Heritage Archive to assemble packages to be deposited in Vitam.
Integration with EOSC and EOSC-Pillar partners

- The Software Heritage has been registered in the EOSC catalog.
- AAI integration has been tested and validated connecting Software Heritage’s authentication system with Indigo-IAM.
- Some EOSC-Pillar partners (especially UC6.2 and UC6.3) plan to have their EOSC-Pillar related source code repositories (eg. D4Science EarthScience VRE or GitHub repositories) harvested and archived by Software Heritage.
- Some work is in progress to integrate Software Heritage, via a SWHID, as BinderHub / JupyterHub source of easy to run and reproducible Jupyter notebooks.
Outline

1. Introduction
2. Software Heritage: the universal source code archive
3. SWHID: Software Heritage Persistent Identifiers
4. Archiving source code in the Software Heritage Archive
5. Retrieving Source Code from the Archive
6. Use Case Status
7. Conclusion
Software Heritage

- universal source code archive
- intrinsic identifiers (SWHIDs)
- open, non-profit, long term
- infrastructure for Open Science
Come in, we’re open!

Software Heritage
- universal source code archive
- intrinsic identifiers (SWHIDs)
- open, non-profit, long term
- infrastructure for Open Science

You can help
- use SWH and save *relevant* source code
- build on SWH (see swmath.org and ipol.im)
- contribute to SWH - it is *open source*
- spread the word: become a Software Heritage Ambassador
Thank you

Software Heritage

Thank you!

Contact: david.douard@softwareheritage.org

- P. Alliez, R. Di Cosmo, B. Guedj, A. Girault, M. Hacid, A. Legrand, N. Rougier
  Attributing and Referencing (Research) Software: Best Practices and Outlook From Inria

- Roberto Di Cosmo, Morane Gruenpeter, Stefano Zacchioli
  Referencing Source Code Artifacts: a Separate Concern in Software Citation
  Computing in Science & Engineering, 2020, ISSN: 1521-9615