

Software Heritage

Towards A New Era for Software Engineering, Cybersecurity, and AI

Roberto Di Cosmo

Director, Software Heritage
Inria and Université Paris Cité

July 4th, 2025

Oregon Programming Language School



Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

- 1 Introduction
- 2 Meet Software Heritage
- 3 Software for research: reference archive
- 4 Research on software: datasets from the archive
- 5 Selected highlight: Impact on software studies
- 6 Selected highlight: Improving Open Source Security with SWH
- 7 Selected highlight: AI and transparent LLMs
- 8 Conclusion

Short Bio: Roberto Di Cosmo

Computer Science professor in Paris, now working at INRIA

- 35+ years of research (Theor. CS, Programming, Software Engineering, Erdos #: 3)
- 25+ years of Free and Open Source Software
- 15+ years building and directing structures for the common good



1999 *DemoLinux* – first live GNU/Linux distro

2007 *Free Software Thematic Group*

150 members 40 projects 200Me

2008 *Mancoosi project* www.mancoosi.org

2010 *IRILL* www.irill.org

2015 *Software Heritage* at INRIA

2018 *National Committee for Open Science*, France

2021 *EOSC Task Force on Infrastructures for Software*,
European Union

Harold Abelson, *Structure and Interpretation of Computer Programs* (1st ed.)

1985

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

Apollo 11 source code (excerpt)

```
P63SPOT3      CA      BIT6      # IS THE LR ANTENNA IN POSITION 1 YET
              EXTEND
              RAND    CHAN33
              EXTEND
              BZF     P63SPOT4      # BRANCH IF ANTENNA ALREADY IN POSITION 1

              CAF     CODE500      # ASTRONAUT: PLEASE CRANK THE
              TC      BANKCALL     # SILLY THING AROUND
              CADR    GOPERF1
              TCF     GOTOP00H     # TERMINATE
              TCF     P63SPOT3     # PROCEED SEE IF HE'S LYING

P63SPOT4      TC      BANKCALL     # ENTER INITIALIZE LANDING RADAR
              CADR    SETPOS1

              TC      POSTJUMP     # OFF TO SEE THE WIZARD ...
              CADR    BURNBABY
```

Covid Sim (excerpt)

```
/**
 * @brief The basic unit of the simulation and is associated to a geographical location.
 *
 * Interventions (e.g., school closures) are tracked at this level. It contains a list of its
 * members (people), places (schools, universities, workplaces etc.), road networks, links to
 * airports etc.
 */
struct Microcell
{
    /* Note use of short int here limits max run time to USHRT_MAX*ModelTimeStep - e.g. 65536*0.25=16384 days=44 yrs.
     * Global search and replace of 'unsigned short int' with 'int' would remove this limit, but use more memory.
     */

    int n; // Number of people in microcell
    int adunit; // admin unit microcell belongs to
    int* members; // array of members/hosts of microcell

    int* places[MAX_NUM_PLACE_TYPES]; // list of places (of various place types) within microcell
    unsigned short int NumPlacesByType[MAX_NUM_PLACE_TYPES]; // number of places (of various place types) within microcell
    unsigned short int keyworkerproph, move_trig, place_trig, socdist_trig, keyworkerproph_trig;
    unsigned short int move_start_time, move_end_time;
    unsigned short int place_end_time, socdist_end_time, keyworkerproph_end_time;
    TreatStat moverest, treat, vacc, socdist, placeclose;
    unsigned short int treat_trig, vacc_trig;
    unsigned short int treat_start_time, treat_end_time;
    unsigned short int vacc_start_time;
    IndexList* AirportList;
};
```

Len Shustek, *Computer History Museum*

2006

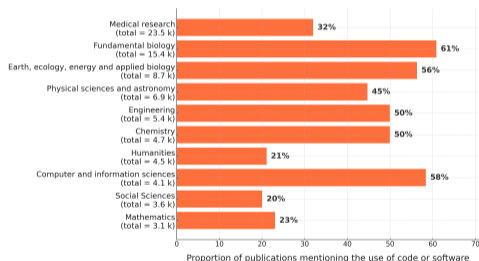
“Source code provides a view into the mind of the designer.”

(Open Source) Software is *precious technical and scientific knowledge*

Yuval Noah Harari (on COVID 19)

"The real antidote [to epidemic] is scientific knowledge and global cooperation."

Software powers modern research



20%+ articles use software, all disciplines
2025 French Open Science Monitor

We can still talk to the early inventors



"Telling historical stories is the best way to teach. It's much easier to understand something if you know the threads it is connected to."

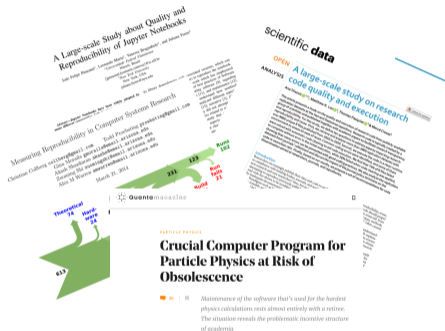
Donald E. Knuth
Len Shustek

CACM, January 2021

How are we managing our (open source) software?

Reproducibility, maintenance in Academia

Security, integrity, traceability in Industry



(articles: [here](#), [here](#), [here](#) and [here](#))



Can they track the software that they

- ship, use, acquire
- has that bug or vulnerability

We need a *dedicated infrastructure* to adress *all* this!

- 1 Introduction
- 2 Meet Software Heritage
- 3 Software for research: reference archive
- 4 Research on software: datasets from the archive
- 5 Selected highlight: Impact on software studies
- 6 Selected highlight: Improving Open Source Security with SWH
- 7 Selected highlight: AI and transparent LLMs
- 8 Conclusion



Software Heritage

THE GREAT LIBRARY OF 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 and share all software source code

Research infrastructure



enable analysis of all software source code

One infrastructure
non profit
open and shared



Inria



The largest archive ever built

Size

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



Diamond sponsors



Platinum sponsors



Gold sponsors



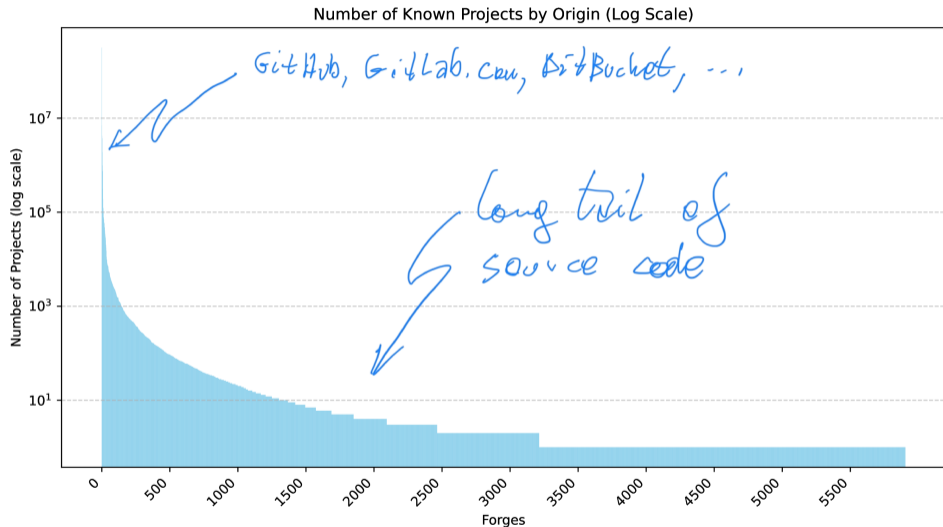
Silver sponsors



Bronze sponsors

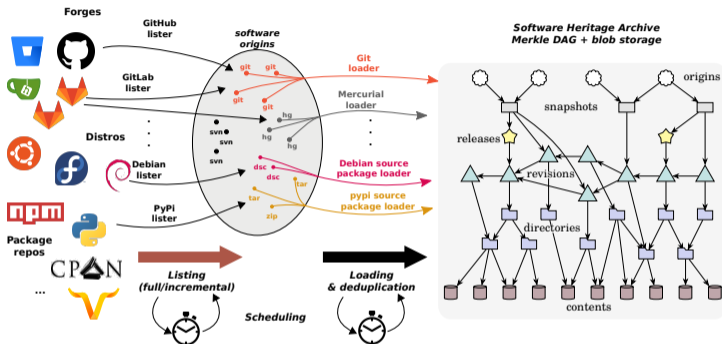


A peek at the code hosting landscape



Over 2 million projects in the long tail!

The archive, under the hood



Global development history permanently archived in a uniform data model

- over 24 billion unique source files from over 375 million software projects
- ~2PB (compressed) blobs, ~50 B nodes, ~800 B edges

The Software Hash identifier (SWHID)

Software Heritage Identifiers (SWHID)

see swhid.org



50+B
intrinsic,
decentralised,
cryptographic

Full fledged *source code references* for traceability, integrity and reproducibility

- Linux Foundation [SPDX 2.2](https://spdx.org/licenses/)
- IANA-registered "swh:"
- WikiData property [P6138](https://www.wikidata.org/wiki/P6138)

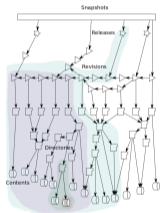
Examples: [Apollo 11 AGC excerpt](#), [Quake III rsqrt](#)
Guidelines available, see [the HOWTO](#)

[ISO/IEC 18670](#), see swhid.org

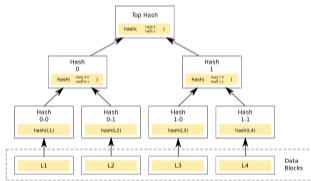
A revolutionary infrastructure

Modern "Library of Alexandria", *international, non profit, long term* initiative addressing the needs of *industry, research, culture and society as a whole*

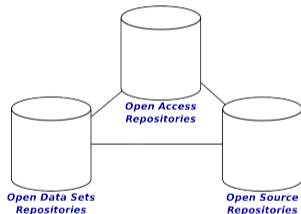
Software Graph



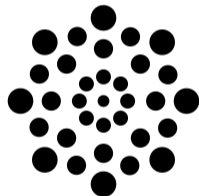
Software Blockchain



Open Science pillar



Big Code



One infrastructure, shared: more efficient, less waste ...

... supporting a broad range of applications

- 1 Introduction
- 2 Meet Software Heritage
- 3 Software for research: reference archive**
- 4 Research on software: datasets from the archive
- 5 Selected highlight: Impact on software studies
- 6 Selected highlight: Improving Open Source Security with SWH
- 7 Selected highlight: AI and transparent LLMs
- 8 Conclusion

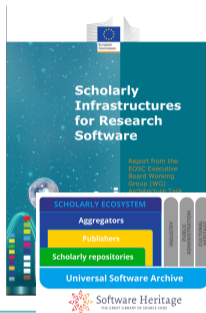
An example is worth a thousand words

- Browse (e.g. [Apollo 11 \[excerpt\]](#), your work [may be already there](#) !)
- Trigger archival, use the [updateswh](#) browser extension, configure the [webhooks](#)
- Get and use SWHIDs ([full specification available online](#))
- Cite software with [biblatex-software](#) package from CTAN
 - [Overleaf ACMART template](#) available
- Example in journals: [article from IPOL](#)
- Example with Parmap: [devel on Github](#), [archive in SWH](#), [curated deposit in HAL](#)
- Extracting all the software products [for Inria](#), [for CNRS](#), [for CNES](#), [for LIRMM](#) or [for Rémi Gribonval](#) using [HalTools](#)
- [Curated deposit in SWH via HAL](#), see for example: [LinBox](#), [SLALOM](#), [Givaro](#), [NS2DDV](#), [SumGra](#), [Coq proof](#), ...
- Example use in research articles:
 - compare Fig. 1 and conclusions in [the 2012 version](#) and [the updated version](#)
 - SWHID in [a replication experiment](#)

A few adoption indicators



Policy



- [Recommendations in ANR 2023 guidelines \(p. 17\)](#)
- HAL+SWH in [the Open Science software booklet](#)

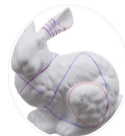
Users and collaborations



What are they “referencing”?

source	n	percentage
Not available	2868	46.22
GitHub	1151	18.55
software heritage	387	6.24
zenodo	142	2.29
r package	70	1.13
cran	56	0.90
r package version	54	0.87
gitlab	35	0.56

Graphics Replicability Stamp Initiative



b/Surf: Interactive Bézier Splines on Surface Meshes

Claudio Mancinelli, Giacomo Nazzaro, Fabio Pellacini, Enrico Puppo
IEEE Transactions on Visualization and Computer Graphics (TVCG)



Repository



Projects



FAIRCORE4EOSC
Core Components Supporting a FAIR EOSC

The CodeMeta Project



FAIR-IMPACT
Expanding FAIR solutions across EOSC

- 1 Introduction
- 2 Meet Software Heritage
- 3 Software for research: reference archive
- 4 Research on software: datasets from the archive**
- 5 Selected highlight: Impact on software studies
- 6 Selected highlight: Improving Open Source Security with SWH
- 7 Selected highlight: AI and transparent LLMs
- 8 Conclusion

<https://registry.opendata.aws/software-heritage/>

Registry of Open Data on AWS



Software Heritage Graph Dataset

[digital preservation](#) [free software](#) [open source software](#) [source code](#)

Description

[Software Heritage](#) is the largest existing public archive of software source code and accompanying development history. The Software Heritage Graph Dataset is a fully deduplicated Merkle DAG representation of the Software Heritage archive. The dataset links together file content identifiers, source code directories, Version Control System (VCS) commits tracking evolution over time, up to the full states of VCS repositories as observed by Software Heritage during periodic crawls. The dataset's contents come from major development forges (including GitHub and GitLab), FOSS distributions (e.g., Debian), and language-specific package managers (e.g., PyPI). Crawling information is also included, providing timestamps about when and where all archived source code artifacts have been observed in the wild.

Update Frequency

Data is updated yearly

License

Creative Commons Attribution 4.0 International. By accessing the dataset, you agree with the Software Heritage [Ethical Charter for using the archive data](#) and the [terms of use for bulk access](#).

Documentation

<https://docs.softwareheritage.org/devel/swh-dataset/graph/athena.html>

Managed By

Software Heritage

See all datasets managed by [Software Heritage](#).

Contact

R. Di Cosmo roberto@dicosmo.org (CC-BY 4.0)

Resources on AWS

Description

Software Heritage Graph Dataset

Resource type

S3 Bucket

Amazon Resource Name (ARN)

```
arn:aws:s3:::softwareheritage
```

AWS Region

```
us-east-1
```

AWS CLI Access (No AWS account required)

```
aws s3 ls --no-sign-request s3://softwareheritage/
```

Description

[S3 Inventory](#) files

Resource type

S3 Bucket

Amazon Resource Name (ARN)

```
arn:aws:s3:::softwareheritage-inventory
```

AWS Region

```
us-east-1
```

AWS CLI Access (No AWS account required)

```
aws s3 ls --no-sign-request s3://softwareheritage-inventory/
```

Example: most popular commit verbs (stemmed)

Query using Amazon Athena

```
SELECT COUNT(*) AS C, word FROM (  
  SELECT word_stem(lower(split_part(  
    trim(from_utf8(message)), ' ', 1)))  
  AS word FROM revision  
  WHERE length(message) < 1000000)  
WHERE word != ''  
GROUP BY word  
ORDER BY C  
DESC LIMIT 20;
```

Results

Completed Time in queue: 272 ms Run time: 33.545 sec Data scanned: 94.51 GB

Results (20) [Copy](#) [Download results](#)

#	c	word
1	271573294	updat
2	163328012	merg
3	140044381	add
4	105800317	fix
5	103646653	ad
6	52891401	bump
7	50067041	initi
8	45609622	creat
9	42633225	remov
10	32230842	chang
11	23110410	delet
12	20734745	new
13	16644508	commit
14	15651821	test

State-of-the-art graph compression from social networks



Paolo Boldi, Antoine Pietri, Sebastiano Vigna, Stefano Zacchioli

Ultra-Large-Scale Repository Analysis via Graph Compression

SANER 2020, 27th Intl. Conf. on Software Analysis, Evolution and Reengineering. IEEE

Results

Full graph structure (50 B nodes, 800 B edges) in 400 GiB RAM

- traversal time is tens of ns per edge
- bidirectional traversals implemented
- **beware:** metadata access is still *off RAM*

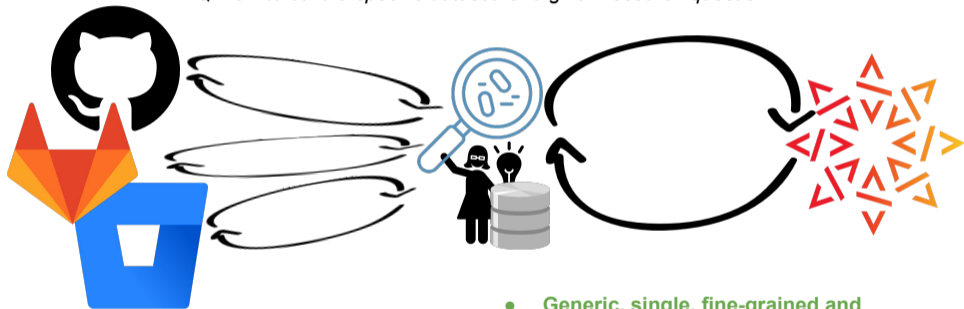
Java Rust and gRPC APIs available ...

docs.softwareheritage.org/devel/swh-graph/grpc-api.html

- 1 Introduction
- 2 Meet Software Heritage
- 3 Software for research: reference archive
- 4 Research on software: datasets from the archive
- 5 Selected highlight: Impact on software studies**
- 6 Selected highlight: Improving Open Source Security with SWH
- 7 Selected highlight: AI and transparent LLMs
- 8 Conclusion

Mining Android Applications on Software Heritage

RQ: how to build a specific dataset for a given research question?



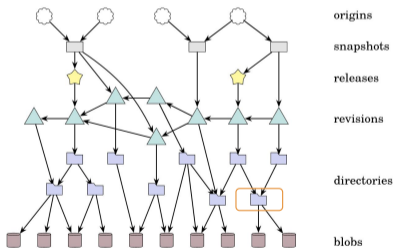
- **Specific and limited API**
- **Hardly reproducible**

- **Generic, single, fine-grained and unlimited API**
- **Growing number of source codes**
- **Easy to update the dataset**

(from the Inria/IRISA DiverSE team)

Using the SWH merkle dag to identify android repositories

Identify android application repositories = Find the AndroidManifest.xml among the sources

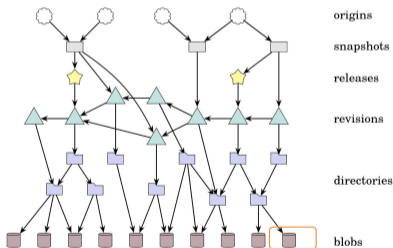


SWH Merkle DAG, Antoine Pietri

1) Iterate over the graph nodes until you find a directory node containing a file named "AndroidManifest.xml".

Using the SWH merkle dag to identify android repositories

Identify android application repositories = Find the AndroidManifest.xml among the sources

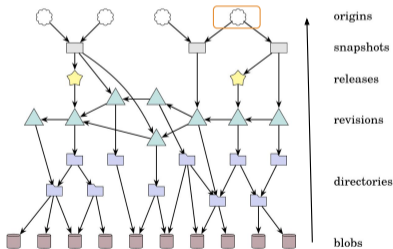


SWH Merkle DAG, Antoine Pietri

2) Extract the SWH identifier of the blob corresponding to the AndroidManifest.xml and download the corresponding file through the SWH Web API

Using the SWH merkle dag to identify android repositories

Identify android application repositories = Find the AndroidManifest.xml among the sources



SWH Merkle DAG, Antoine Pietri

3) Traverse the graph in backward direction to the origin node and get the repository url

Broad variety of sources in *one open dataset*

reduces usual GH bias

Reference simple *standard data format*

VCS and forge details are abstracted away

Simplifies reproducibility packages

no need to create a full copy, *just list the SWHIDs!*

Software Heritage does the heavy lifting for you

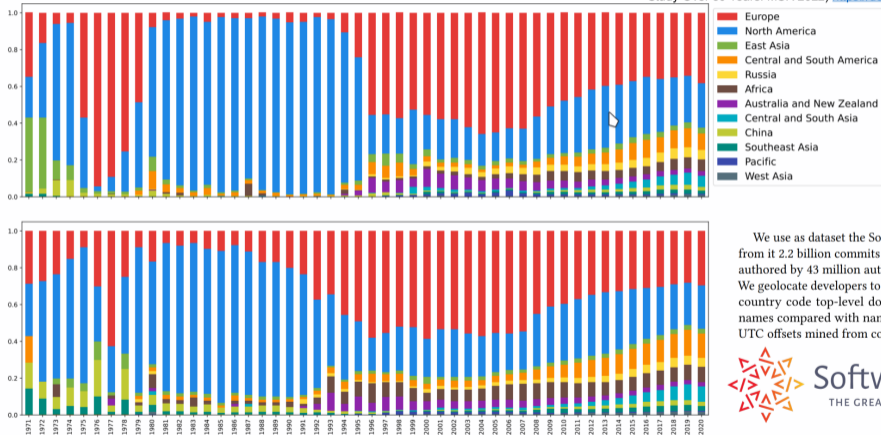
no need to scrape/download repositories all over again

Example: (Open) Source Code comes from all over the world

MSR '22, May 23–24, 2022, Pittsburgh, PA, USA

Davide Rossi and Stefano Zacchiroli

Geographic Diversity in Public Code Contributions: An Exploratory Large-Scale Study Over 50 Years. MSR 2022) <https://doi.org/10.1145/3524842.3528471>



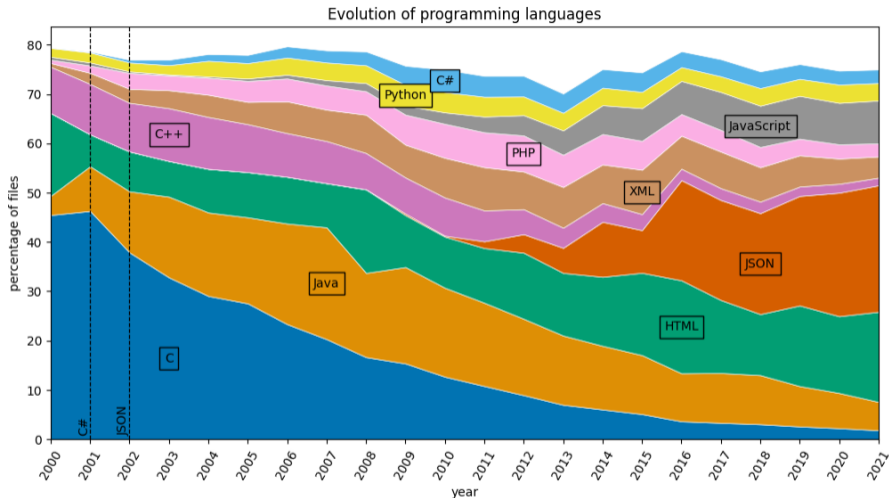
We use as dataset the Software Heritage archive [3] and analyze from it 2.2 billion commits archived from 160 million projects and authored by 43 million authors during the 1971–2021 time period. We geolocate developers to 12 world regions, using as signals email country code top-level domains (ccTLDs) and author (first/last) names compared with name distributions around the world, and UTC offsets mined from commit metadata.



Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

Figure 3: Ratio of commits (above) and active authors (below) by world zone over the 1971–2020 period.

Example: Programming language evolution in (Open) Source



A. Desmazières, R. Di Cosmo, V. Lorentz

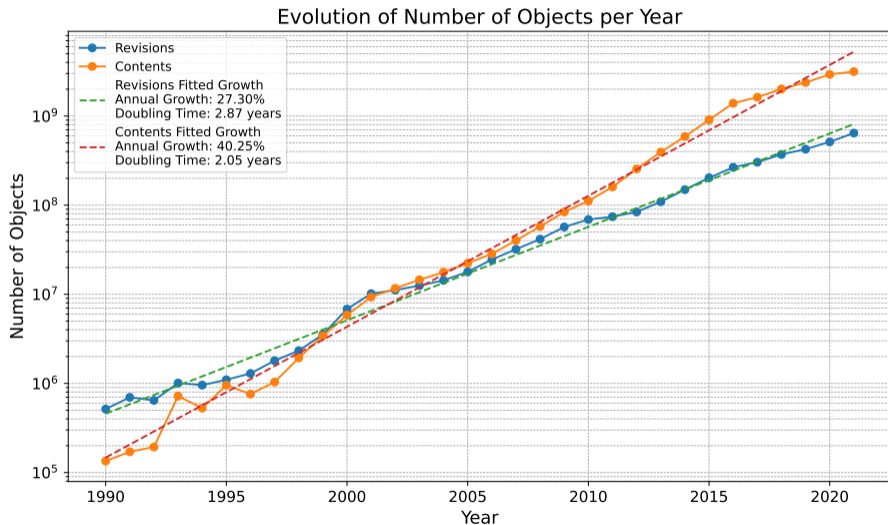
50 years of programming language evolution through the Software Heritage Looking Glass

International Conference on Mining Software Repositories (MSR) 2025.

<https://hal.science/hal-04924849/>



Example: (Open) Source Code grows at an exponential rate



Get the data
and the code



Example: Institutional contribution analysis

Question:

can we leverage the Software Heritage graph to map contributions from a research institution in the galaxy of software development?

Answer: let's try

- **Extract:** projects and contributions using email domain matching in the Software Heritage graph.
- **Deduplicate:** Cluster forks/copies using commit metadata, select a canonical repository.
- **Enrich:** Add GitHub metadata (topics, README, etc.) for context and visibility.
- **Analyze:** Compute key metrics on impact, contributor domains, and research relevance.

First SWH VLT pictures : UC Berkeley contribution to Open Source



2626855

Contributions



34113

Software Projects

Active last 12 months: 997

Still crawled: 31763



27983

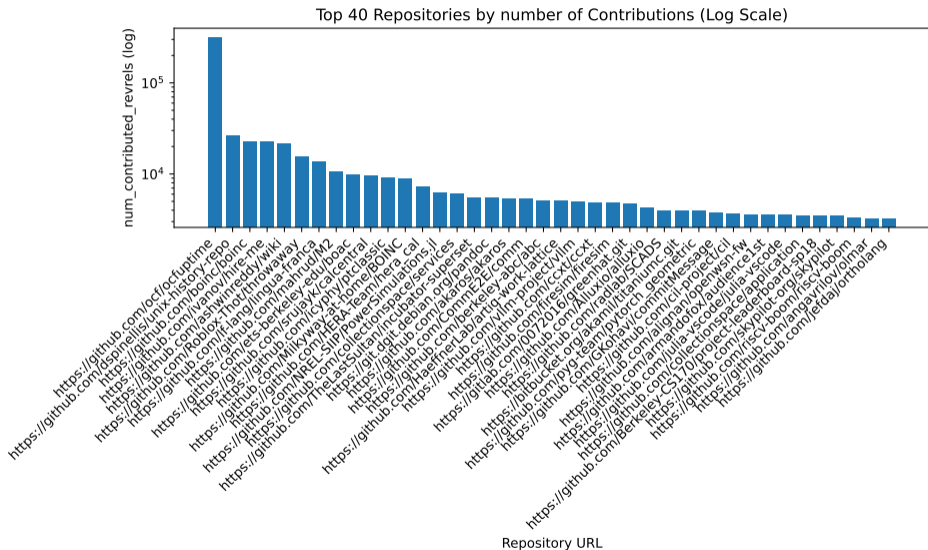
Contributors



1970-01-01
2025-05-17

Time Span

First SWH VLT pictures: UC Berkeley contribution to Open Source

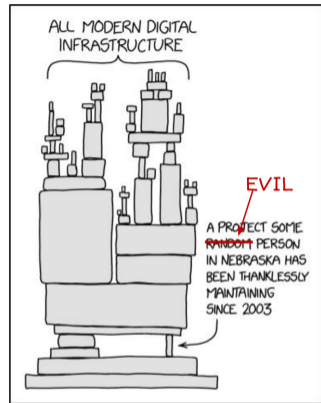
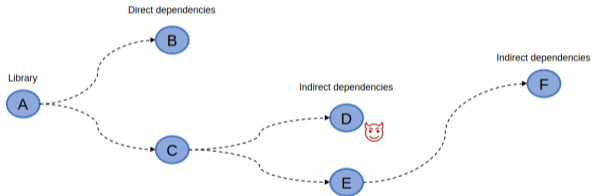


- 1 Introduction
- 2 Meet Software Heritage
- 3 Software for research: reference archive
- 4 Research on software: datasets from the archive
- 5 Selected highlight: Impact on software studies
- 6 Selected highlight: Improving Open Source Security with SWH**
- 7 Selected highlight: AI and transparent LLMs
- 8 Conclusion

Software supply chain attacks

Reusing OSS via dependencies

- **Software dependencies:** a popular way of reusing open source software.
- Software product *A* uses functionalities implemented in OSS product *B* ... and so on.



based on xkcd.com/2347

Attacking the software supply chain

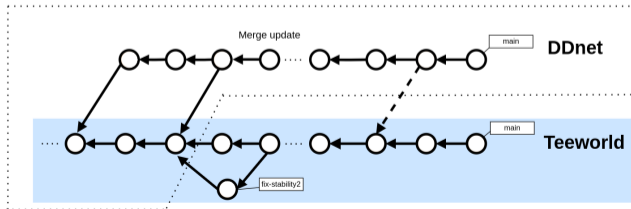
- Attacking **undermaintained "leaf" packages** (e.g., D) → efficient attack strategy
- Many documented attacks: event-stream (2018), node-ipc (2022), XZ utils (2024), ...

One-day vulnerabilities

- Def.: vulnerabilities that are **publicly known, but not fixed yet** in software you use.
- Challenge: **identify them quickly and exhaustively**, then apply countermeasures.
- Many tools available to detect one-day vulnerabilities *via declared dependencies*.

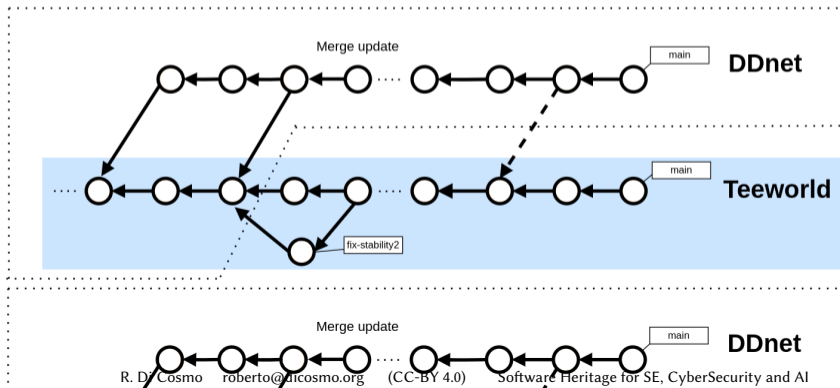
Reusing OSS via forks

Open source is also reused via **forking**: (1) start from existing OSS (e.g., Teeworlds game), (2) create your own (e.g., DDnet), (3) periodically integrate changes.



Vulnerability propagation through forks

- Any change to a piece of software (*commit*) can **introduce a new vulnerability**.
- Or it can **fix an existing vulnerability**.
- What happens if a project is forked **between introduction and fix** of a vulnerability?
- It inherits the vulnerability, ... until the change with the fix is integrated.



Approach

- 1 Start from a **public DB of vuln. introduced/fixed** in public commits (e.g., [OSV.dev](#)).
- 2 **"Color" the entire graph** of public code development history **with vulnerability info.**
 - Software Heritage is the only place where this can be done at the scale of all forks, across all public code, independently of specific development platforms.
- 3 **Inform maintainers** of vulnerable forks. (After validation.)

Early results

- Identified 2.2 M (million) forks of repositories referenced by OSV.dev, containing vulnerable commits; **1.3 M forks vulnerable** in their most recent commit.
- 86.6 M vulnerable commits were specific to forks, **not findable with current tools.**
- Among 66 manually vetted cases, **5 confirmed vulnerabilities (1 critical).**



Romain Lefeuvre, Charly Reux, Stefano Zacchiroli, Olivier Barais, Benoit Combemale
Chasing One-day Vulnerabilities Across Open Source Forks
To appear, 2025.

An universal knowledge base about public code vulnerabilities

Vision

- **Software Heritage** is the perfect (and only) place where to build an universal knowledge base that maps known vulnerabilities to public code artifacts.
- SWH can provide an **open data API mapping SWHIDs to CVEs**, that knows about *all public commits* and can be leveraged to increase open source security.

EU Cyber Resilience Act (CRA)

- Key helper to abide to CRA obligations, coming into effect ~Q3 2026.
- SWH funding member of the **Open Regulatory Compliance Working Group**.



Roadmap

- Context: SWHSec project (PTCC-funded, 2023-2027).
- Current status: working prototype that processes OSV.dev data and use it to "color" the entire SWH commit graph (~5 billion commits) with vulnerability information.

- 1 Introduction
- 2 Meet Software Heritage
- 3 Software for research: reference archive
- 4 Research on software: datasets from the archive
- 5 Selected highlight: Impact on software studies
- 6 Selected highlight: Improving Open Source Security with SWH
- 7 Selected highlight: AI and transparent LLMs**
- 8 Conclusion

Software Heritage and Generative AI, first contacts

October 19, 2023

Software Heritage Statement on Large Language Models for Code



Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting machine learning models must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The initial training data extracted from the Software Heritage archive must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

Software Heritage and Generative AI, first contacts

October 19, 2023

Software Heritage Statement on Large Language Models for Code

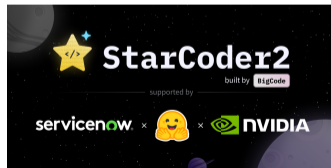
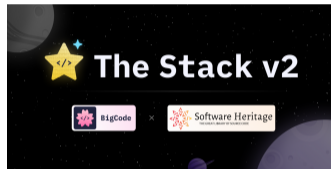


Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting machine learning models must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The initial training data extracted from the Software Heritage archive must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

February 2024

Yes, it's possible!



Software Heritage and Generative AI, first contacts

October 19, 2023

Software Heritage Statement on Large Language Models for Code

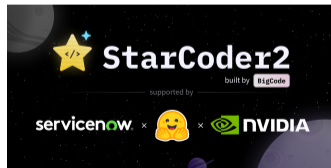
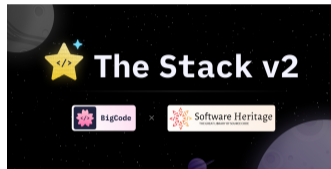


Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting machine learning models must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The initial training data extracted from the Software Heritage archive must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

February 2024

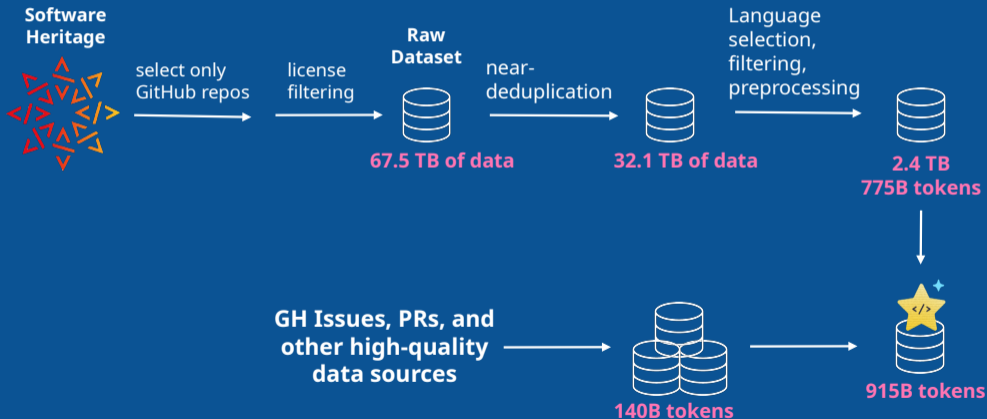
Yes, it's possible!



But it's hard...

The Stack v2

Data collection pipeline fully open and transparent built by BigCode



Lessons learned

Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data extracted from the Software Heritage archive* must be fully and precisely identified by, for example, publishing the corresponding **SWHID identifiers** (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

Lessons learned

Transparency is easy: [SWHID](#) (undergoing ISO standardisation) and Software Heritage
N.B. : may be mandated by regulations!

Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data extracted from the Software Heritage archive* must be fully and precisely identified by, for example, publishing the corresponding **SWHID identifiers** (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

Lessons learned

Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data extracted from the Software Heritage archive* must be fully and precisely identified by, for example, publishing the corresponding **SWHID identifiers** (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

Transparency is easy: [SWHID](#) (undergoing ISO standardisation) and Software Heritage
N.B. : may be mandated by regulations!

Opt out is complex: who is *the real right owner?*
(similar issues to license compliance)

Lessons learned

Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data* extracted from the Software Heritage archive must be fully and precisely identified by, for example, publishing the corresponding **SWHID identifiers** (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

Transparency is easy: [SWHID](#) (undergoing ISO standardisation) and Software Heritage
N.B. : may be mandated by regulations!

Opt out is complex: who is *the real right owner*?
(similar issues to license compliance)



Lessons learned

Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data* extracted from the Software Heritage archive must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

Transparency is easy: [SWHID](#) (undergoing ISO standardisation) and Software Heritage
N.B. : may be mandated by regulations!

Opt out is complex: who is *the real right owner*?
(similar issues to license compliance)

● **Building the training set is complex:** e.g. includes **license compliance** alike work **at massive scale**



Lessons learned

Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data* extracted from the Software Heritage archive must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

Transparency is easy: [SWHID](#) (undergoing ISO standardisation) and Software Heritage
N.B. : may be mandated by regulations!

Opt out is complex: who is *the real right owner?*
(similar issues to license compliance)



- **Building the training set is complex:** e.g. includes **license compliance** alike work **at massive scale**
- **Generating attribution information on model output is more complex** than license compliance

Lessons learned

Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data* extracted from the Software Heritage archive must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

Transparency is easy: [SWHID](#) (undergoing ISO standardisation) and Software Heritage
N.B. : may be mandated by regulations!

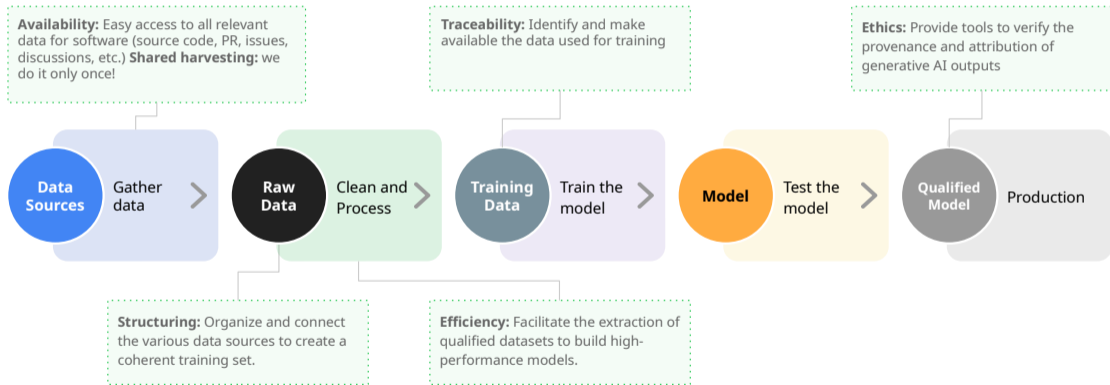
Opt out is complex: who is *the real right owner*?
(similar issues to license compliance)



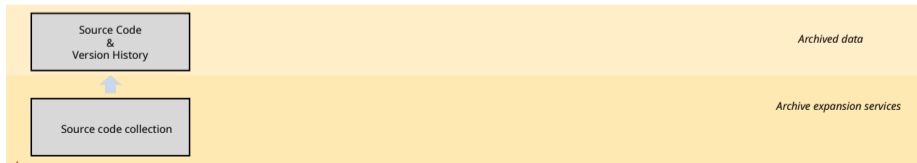
- **Building the training set is complex:** e.g. includes **license compliance** alike work **at massive scale**
- **Generating attribution information on model output is more complex** than license compliance

We need a **coordinated effort** to ensure fully open models will succeed!

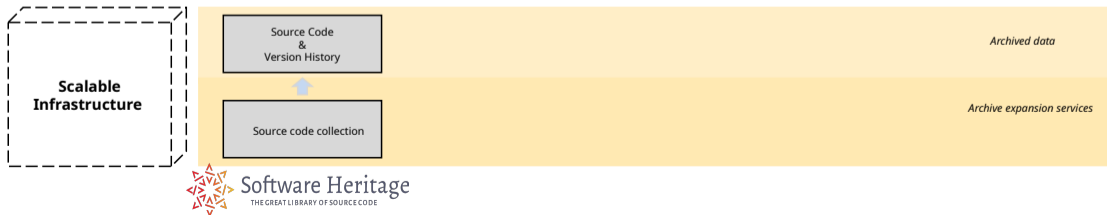
A STEP FORWARD: CodeCommons



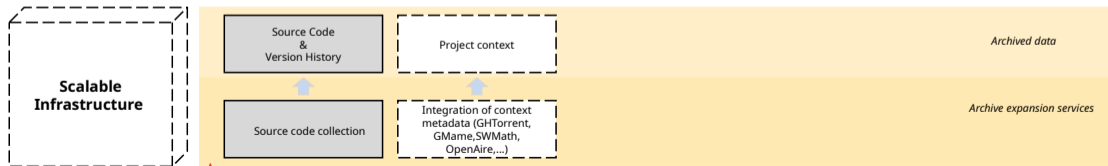
CodeCommons: bird's eye view (technical focus)



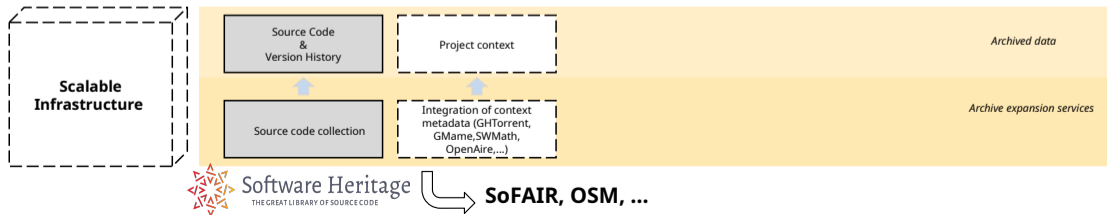
CodeCommons: bird's eye view (technical focus)



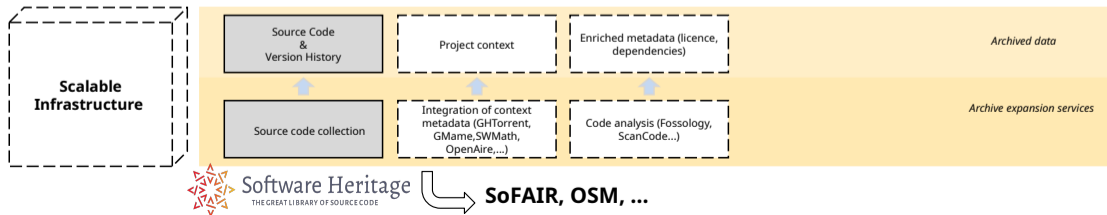
CodeCommons: bird's eye view (technical focus)



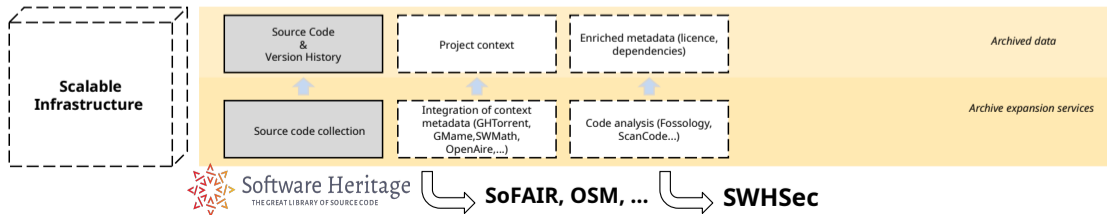
CodeCommons: bird's eye view (technical focus)



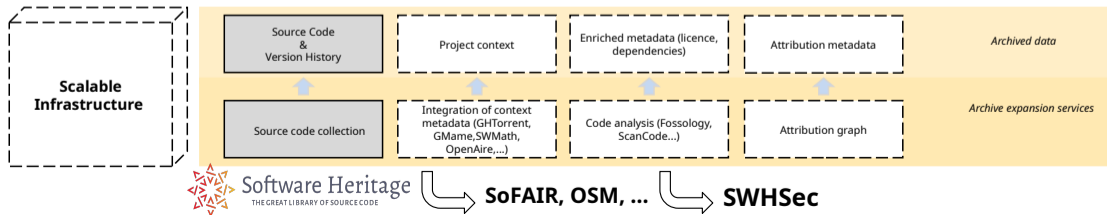
CodeCommons: bird's eye view (technical focus)



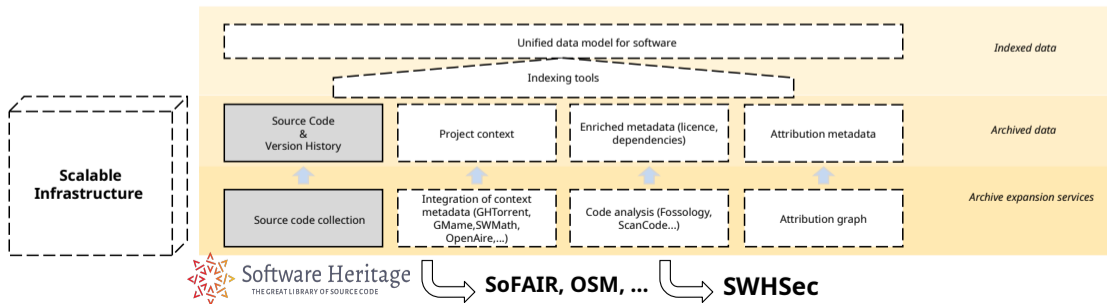
CodeCommons: bird's eye view (technical focus)



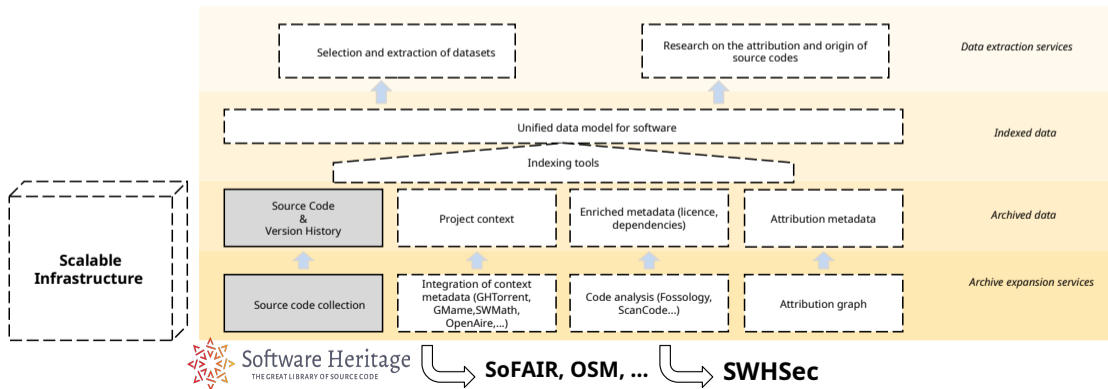
CodeCommons: bird's eye view (technical focus)



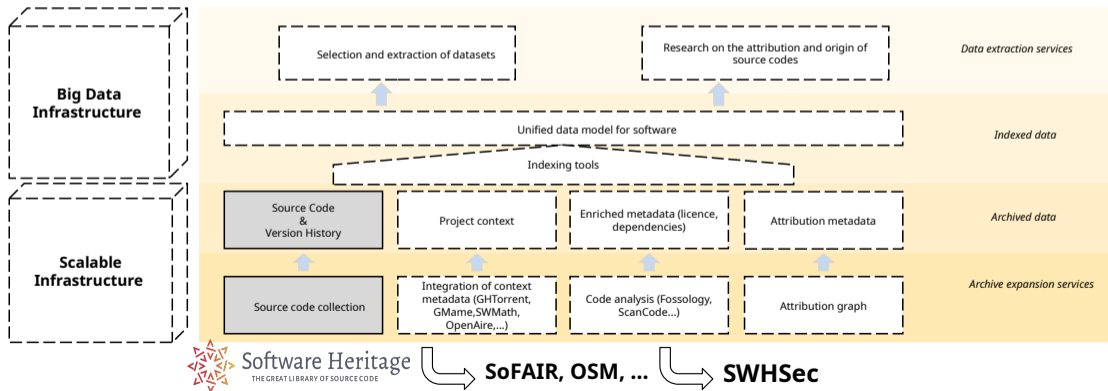
CodeCommons: bird's eye view (technical focus)



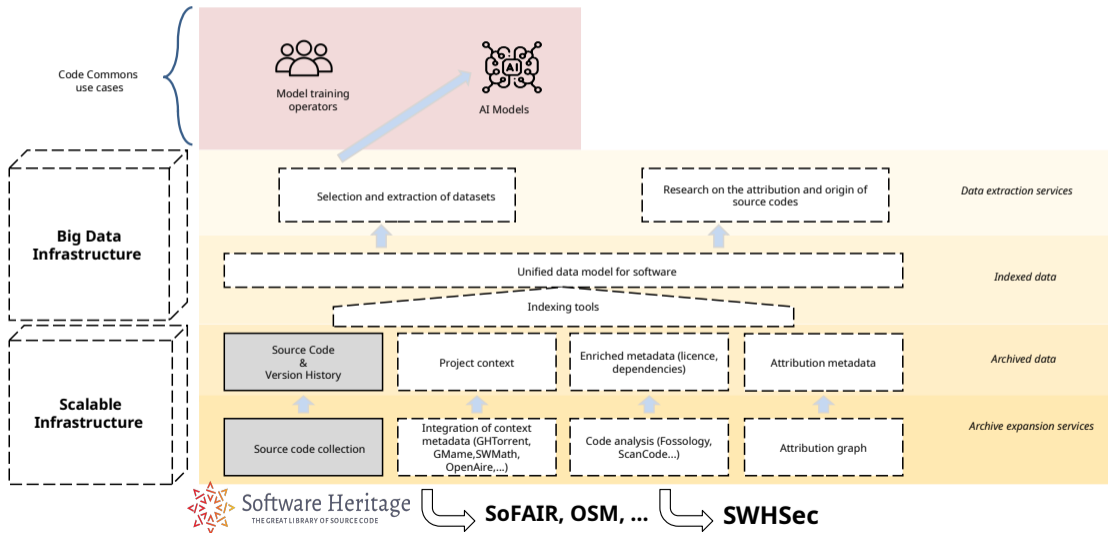
CodeCommons: bird's eye view (technical focus)



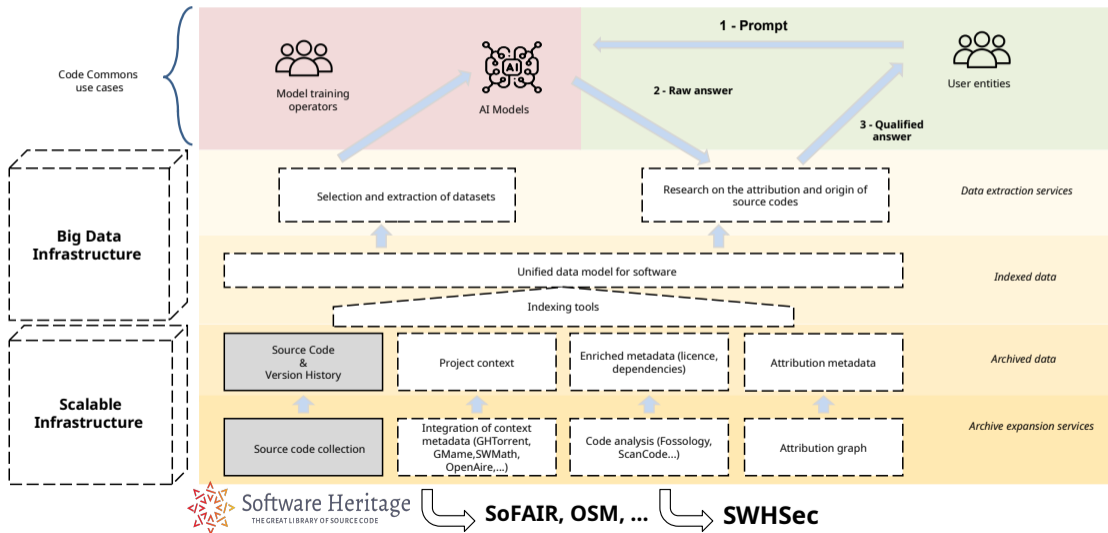
CodeCommons: bird's eye view (technical focus)



CodeCommons: bird's eye view (technical focus)



CodeCommons: bird's eye view (technical focus)



CodeCommons

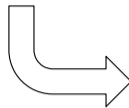
Open, responsible, and transparent AI: Our shared goal

CodeCommons is an ambitious project to create the world's most comprehensive digital commons for code

Building on the existing foundation of Software Heritage, the largest publicly available source code archive, CodeCommons aims to bring into one place all the **critical** and **qualified** information needed to create **smaller, better** datasets for the next generation of AI tools.

At its core, the project prioritizes transparency and traceability, enabling model builders and users to **respect creators' rights** while promoting **sovereign** and **sustainable** AI.

Learn more



Meet the teams



- 1 Introduction
- 2 Meet Software Heritage
- 3 Software for research: reference archive
- 4 Research on software: datasets from the archive
- 5 Selected highlight: Impact on software studies
- 6 Selected highlight: Improving Open Source Security with SWH
- 7 Selected highlight: AI and transparent LLMs
- 8 Conclusion**

Some research and application ideas

Programming language design

Query language for the extended Software Heritage model (Merkle+metadata)

- expressivity
- efficiency

Analysis and classification

- programming language classification
- static analysis at large scale
 - efficient reuse of results
- temporal dependency graphs

Long term reproducibility



Automatic build system fallback on the archive
(generalize the Guix/SWH integration)



A growing and active community

Core Team



All together, 2024 Symposium



R. Di Cosmo roberto@dicosmo.org

Ambassadors



Agustín Benito Bethencourt



Alexis Lebis



Anna-Lena Lamprecht



Baptiste Mèlès



Barry Gungör



Bertrand Néron



Bostjan Spetic



Bruno Khélifi



Camille Françoise



Cécile Arènes



Flavia Marzano



Frédéric Santos



Gavin Henry



Giacomo Lorenzetti



Harsh Pillay



Italo Vignoli



Jaime Arias



Joenio Marques Da Co



Julien Caugant



Linda Angulo-Lopez



Malin Sandström



Max Kalik



Maxence Azzouz-Thuderoz



Mohammad Akhlaghi



Océane Valencia



Pierre Poulain



Sandrine Layrisse



Shiraz Malla Mohamad



Simon Phipps



Violaine Louvet



Wendy Hagenmaier

Becoming an Ambassador

Interested in becoming a Software Heritage ambassador?
Tell us about yourself and your interest in our mission.

ambassadorprogram@softwareheritage.org

(CC-BY 4.0)

Software Heritage for SE, Cybersecurity and AI

July 4th, 2025

39 / 40

A rally flag for a grand vision

Bring together academia, industry, governments, communities

"to build a reference, global infrastructure for open and better software"

Software Heritage is

- vendor neutral, open source
- a worldwide, long term initiative

Call to action

- leverage for AI, Cyber, SwEng
- get involved research, code

Software Heritage enables

- archival, reference, integrity
- qualification, sharing and reuse

- promote for AEC, Journals, etc.
- fund the long term mission

Annual report 2024



5 years in 5 minutes



Get these slides

