Software Heritage

Revolutionary infrastructure for Open Science, Open Source, and AI

Stefano Zacchiroli
Professor @ Télécom Paris
Co-founder & CSO @ Software Heritage

9 January 2025

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

Outline

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

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

A *universal* software archive, as a shared infrastructure

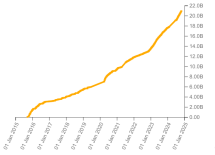
Cultural Heritage | Industry | Research | Public Administration



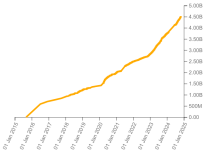

Software Heritage

- One infrastructure, open and shared
- The largest archive ever built

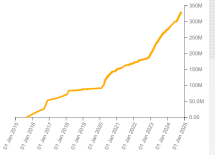
Source files
21,098,377,683



Commits
4,535,490,648



Projects
331,590,016



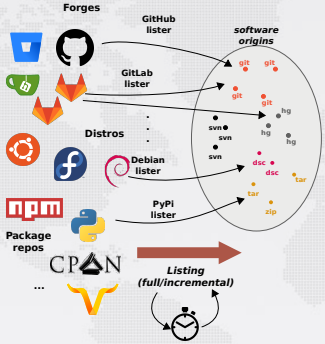
Directories
16,887,472,535

Authors
83,425,808

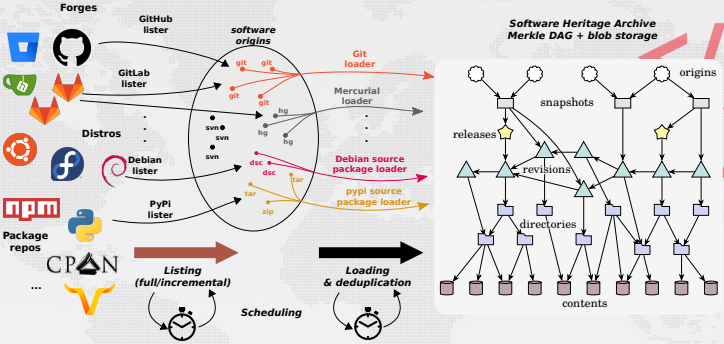
Releases
99,249,418

Bitbucket 2,509,402 origins	56,983 origins	git 24,600 origins
26,599 origins	debian 136,338 origins	53,297 origins
GitHub 197,883,004 origins	gitiles 10,171 origins	GitLab 4,216,298 origins
git 2,926 origins	Gogs 172 origins	971,549 origins
Guix 14,482 origins	GNU 354 origins	heptapod 1,207 origins
launchpad 503,631 origins	Maven 312,461 origins	NixOS 14,482 origins

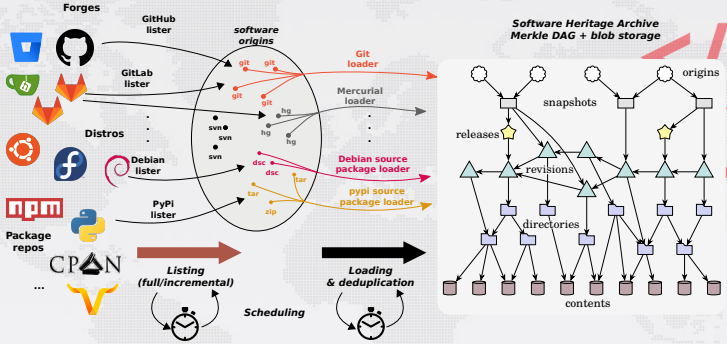
The archive under the hood



The archive under the hood



The archive under the hood



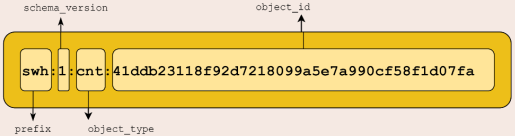
Global development history permanently archived in a uniform data model

- over 20 billion unique source files from over 300 million software projects
- ~2PB (compressed) blobs, ~50 B nodes, ~700 B edges

Software Hash Identifiers (SWHID)

see swhid.org

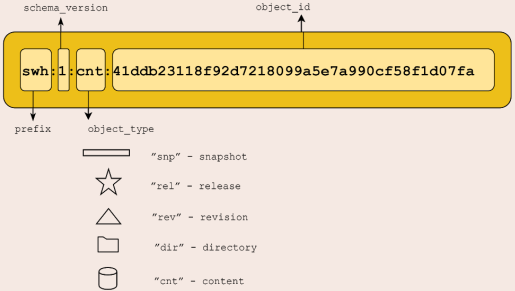
50+B intrinsic, decentralised, cryptographically strong identifiers, SWHIDs



Software Hash Identifiers (SWHID)

see swhid.org

50+B intrinsic, decentralised, cryptographically strong identifiers, SWHIDs



Software Hash Identifiers (SWHID)

see swhid.org

50+B intrinsic, decentralised, cryptographically strong identifiers, SWHIDs

Software Hash Identifiers (SWHID)

see swhid.org

50+B intrinsic, decentralised, cryptographically strong identifiers, SWHIDs

In SPDX 2.2; IANA "swh:"; WikiData P6138; ISO standardization ongoing DIS 18670

The Software Hash persistent identifier (SWHID)

Software Hash Identifiers (SWHID)

see swhid.org

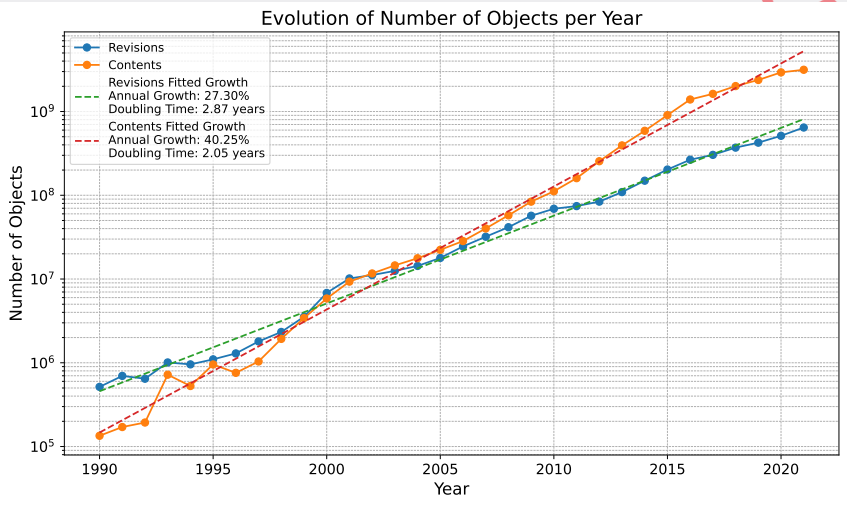
50+B intrinsic, decentralised, cryptographically strong identifiers, SWHIDs

In SPDX 2.2; IANA "swh:"; WikiData P6138; ISO standardization ongoing DIS 18670

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

Examples: Apollo 11 AGC, Quake III rsqrt; Guidelines available: HOWTO and ICMS 2020

Intermezzo: 30 years of growth of public source code

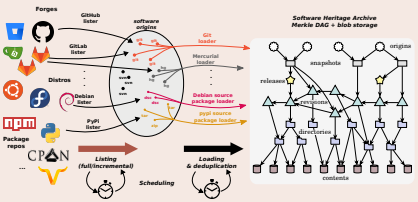


Rousseau, Di Cosmo, Zacchiroli. *Software provenance tracking at the scale of public source code.* Empir. Softw. Eng. 25(4): 2930-2959 (2020)

Outline

Addressing key needs for open science (ARDC)

Archive (20B+ files, 320M+ projects)



Reference (50 billion SWHIDs)

Describe

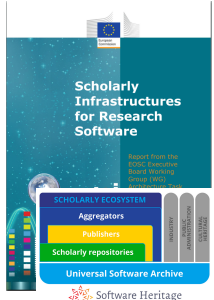
- *Intrinsic metadata* from source code
- Contributed the Codemeta generator

Cite/Credit

- Contributed biblatex-software style
- Software Citation from the archive!

A few adoption indicators

Policy



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

Projects

eosc | FAIRCORE4EOSC
Core Components Supporting a FAIR EOSC

The CodeMeta Project

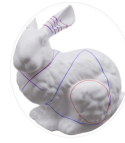
eosc | FAIR-IMPACT
Expanding FAIR solutions across EOSC

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)

doi | Repository

Outline

Tech preview: swh-scanner

Problem

- Detect unknown files in an open source project
- Detect proprietary files leaked on a public forge

Solution

`swh-scanner` : open source and open data source code scanner for compliance workflows, backed by the largest public archive of public source code.

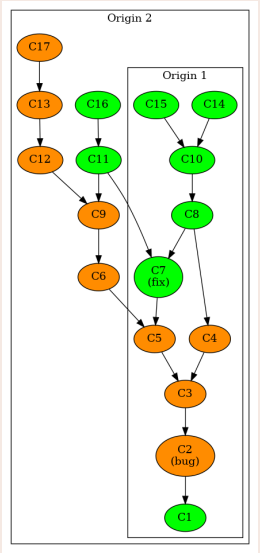
Design

- Software Heritage archive as source of truth about public code
- Merkle DAG model and SWHIDs for maximum scanning efficiency
- Output: source tree partition into known/unknown + provenance information

Package: pypi.org/project/swh.scanner (GPL 3+)
Ref.: Serafini, Zacchiroli. *Efficient Prior Publication Identification for Open Source Code.* OpenSym 2022: 12:1-12:8

Outline

Is my FOSS code (known to be) vulnerable?



- State-of-the-art tech (e.g., OSV.dev by Google) to query whether a given commit is known to be vulnerable do not know about *all* code out there.
- They crawl project repos related to known vulnerabilities, but not their *forks* (across multiple forges!).

Real-world example (one out of many we have identified):

- Linux kernel vulnerability: GSD-2022-1004193
- Example of vulnerable commit: b13baccc3850ca8b8cccbf8ed9912dbaa0fdf7f3
- Fix commit: a92d44b412e75dd66543843165e46637457f22cc
- False negative commit in fork (Raspberry PI Linux): c7c7a1a18af4c3bb7749d33e3df3acdf0a95bbb5

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.

Outline

Looking for founding principles at Software Heritage

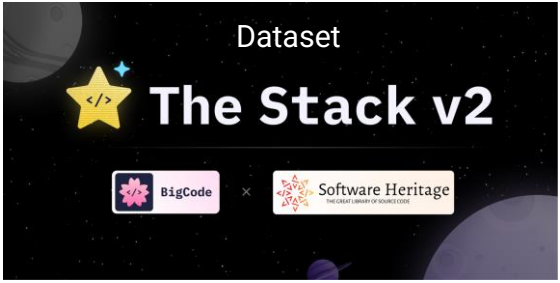
October 19, 2023

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.

Findings from BigCode: The Stack v2 and StarCoder2



Released February 28th 2024

Yes one can build **the best open LLM for code available** while fully adhering to the Software Heritage principles for responsible LLMs, …

and even more: the full training pipeline is made public too!

Outline

Software Heritage is

- The largest public archive of software source code and its history.
- A revolutionary infrastructure for industry, research, culture, society.

Software Heritage enables

- Software archival, reference, integrity, traceability, global knowledge base.
- "Big code" research on the entire software commons.
  (Not covered today; cf. Di Cosmo, Zacchiroli. *The Software Heritage Open Science Ecosystem.* Springer, 2023)

Come in, we're open! https://softwareheritage.org

Software Heritage is

- The largest public archive of software source code and its history.
- A revolutionary infrastructure for industry, research, culture, society.

Software Heritage enables

- Software archival, reference, integrity, traceability, global knowledge base.
- "Big code" research on the entire software commons.

(Not covered today; cf. Di Cosmo, Zacchiroli. *The Software Heritage Open Science Ecosystem.* Springer, 2023)

Join us

Annual report 2023

Outline

7 Research highlights in empirical software engineering

Graph dataset

Use case: large scale analyses of the most comprehensive corpus on the development history of free/open source software.

Antoine Pietri, Diomidis Spinellis, Stefano Zacchiroli
The Software Heritage Graph Dataset: Public software development under one roof
MSR 2019: 16th Intl. Conf. on Mining Software Repositories. IEEE
preprint: `http://deb.li/swhmsr19`

Dataset

- Relational representation of the full graph as a set of tables
- Available as open data: docs.softwareheritage.org/devel/swh-dataset/graph
- Chosen as subject for the MSR 2020 Mining Challenge

Formats

- Local use: set of Apache ORC files (10+ TiB in total)
- Live usage: Amazon Athena (SQL-queriable), Azure Data Lake

Graph dataset — example

```
SELECT COUNT(*) AS c, word FROM (
  SELECT LOWER(REGEXP_EXTRACT(FROM_UTF8(
  message), '^\w+')) AS word FROM revision)
WHERE word != ''
GROUP BY word ORDER BY COUNT(*) DESC LIMIT 5;
```

Graph dataset — example

```
SELECT COUNT(*) AS c, word FROM (
  SELECT LOWER(REGEXP_EXTRACT(FROM_UTF8(
  message), '^\w+')) AS word FROM revision)
WHERE word != ''
GROUP BY word ORDER BY COUNT(*) DESC LIMIT 5;
```

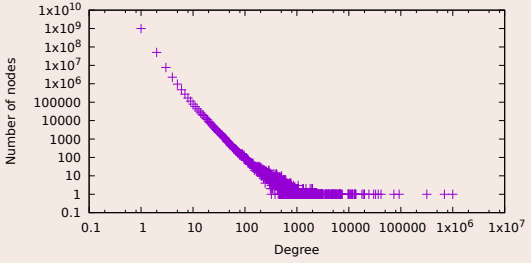
Count	Word
71 338 310	update
64 980 346	merge
56 854 372	add
44 971 954	added
33 222 056	fix

Graph dataset — example

Fork arity

i.e., how often is a commit based upon?

```
SELECT fork_deg, count(*) FROM (
  SELECT id, count(*) AS fork_deg
  FROM revision_history GROUP BY id) t
GROUP BY fork_deg ORDER BY fork_deg;
```

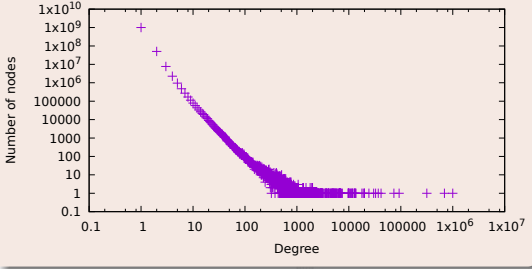


Graph dataset — example

Fork arity

i.e., how often is a commit based upon?

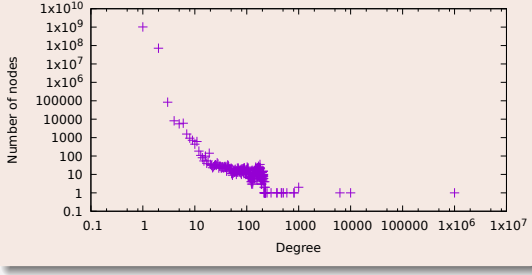
```
SELECT fork_deg, count(*) FROM (
  SELECT id, count(*) AS fork_deg
  FROM revision_history GROUP BY id) t
GROUP BY fork_deg ORDER BY fork_deg;
```



Merge arity

i.e., how large are merges?

```
SELECT merge_deg, COUNT(*) FROM (
  SELECT parent_id, COUNT(*) AS merge_deg
  FROM revision_history GROUP BY parent_id) t
GROUP BY merge_deg ORDER BY merge_deg;
```



License dataset

Stefano Zacchiroli
A Large-scale Dataset of (Open Source) License Text Variants
MSR 2022 (best dataset paper) + Empir. Soft. Eng. 28(6): 147 (2023)
preprint: https://arxiv.org/abs/2308.11258

Dataset

- 6.9 million unique full texts of FOSS license variants
- Detected using filename patterns across the entire SWH archive
  - LICENSE, COPYRIGHT, NOTICE, etc.
- Metadata: file lengths measures, detected MIME type, detected SPDX license (via ScanCode), example origin repository, oldest public commit of origin, ground truth

Use cases

- Empirical studies on FOSS licensing, including phylogenetics
- Training of automated license classifiers
- NLP analyses of legal texts

The Software Heritage Filesystem (SwhFS)

The Software Heritage Filesystem (SwhFS) is a user-space POSIX filesystem that enables browsing parts of the Software Heritage archive as if it were locally available.

- Code: forge.softwareheritage.org/source/swh-fuse
- Documentation: docs.softwareheritage.org/devel/swh-fuse

Thibault Allançon, Antoine Pietri, Stefano Zacchiroli
The Software Heritage Filesystem (SwhFS): Integrating Source Code Archival with Development
ICSE 2021 (Tool track): The 43rd Intl. Conference on Software Engineering
https://arxiv.org/abs/2102.06390

The Software Heritage Filesystem (SwhFS) — example

```
$ mkdir swhfs
$ swh fs mount swhfs/  # mount the archive
$ cd swhfs/

$ cat archive/swh:1:cnt:c839dea9e8e6f0528b468214348fee8669b305b2
#include <stdio.h>

int main(void) {
    printf("Hello, World!\n");
}

$ cd archive/swh:1:dir:1fee702c7e6d14395bbf5ac3598e73bcbf97b030
$ ls | wc -l
127
$ grep -i antenna THE_LUNAR_LANDING.s | cut -f 5
# IS THE LR ANTENNA IN POSITION 1 YET
# BRANCH IF ANTENNA ALREADY IN POSITION 1
```

The Software Heritage Filesystem (SwhFS) — example (cont.)

```
$ cd archive/swh:1:rev:9d76c0b163675505d1a901e5fe5249a2c55609bc

$ ls -F
history/  meta.json@  parent@  parents/  root@

$ jq '.author.name, .date, .message' meta.json
"Michal Golebiowski-Owczarek"
"2020-03-02T23:02:42+01:00"
"Data:Event:Manipulation: Prevent collisions with Object.prototype ..."

$ find root/src/ -type f -name '*.js' | xargs cat | wc -l
10136
```

Graph compression

Paolo Boldi, Antoine Pietri, Sebastiano Vigna, Stefano Zacchiroli

Ultra-Large-Scale Repository Analysis via Graph Compression

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

Research question

Is it possible to efficiently perform software development history analyses at the scale of Software Heritage archive on a single, relatively cheap machine?

Idea

Apply state-of-the-art graph compression techniques from the field of Web graph / social network analysis.

Results

The entire archive graph (35 B nodes, 500 B edges) can be loaded in 300 GiB and then traversed at the cost of tens of ns per edge (= a few hours for a full single-thread visit).

Java and gRPC APIs available: docs.softwareheritage.org/devel/swh-graph/grpc-api.html

Background — (Web) graph compression

Definition (The graph of the Web)

Directed graph that has Web pages as nodes and hyperlinks between them as edges.

Properties (1)

- Locality: pages link to pages whose URLs are lexicographically similar. URLs share long common prefixes.

→ use D-gap compression

Adjacency lists

Node	Outdegree	Successors
…	…	…
15	11	13,15,16,17,18,19,23,24,203,315,1034
16	10	15,16,17,22,23,24,315,316,317,3041
17	0	
18	5	13,15,16,17,50
…	…	…

D-gapped adjacency lists

Node	Outdegree	Successors
…	…	…
15	11	3,1,0,0,0,0,3,0,178,111,718
16	10	1,0,0,4,0,0,290,0,0,2723
17	0	
18	5	9,1,0,0,32
…	…	…

Background — (Web) graph compression (cont.)

Definition (The graph of the Web)

Directed graph that has Web pages as nodes and hyperlinks between them as edges.

Properties (2)

- Similarity: pages that are close together in lexicographic order tend to have many common successors.

→ use reference compression

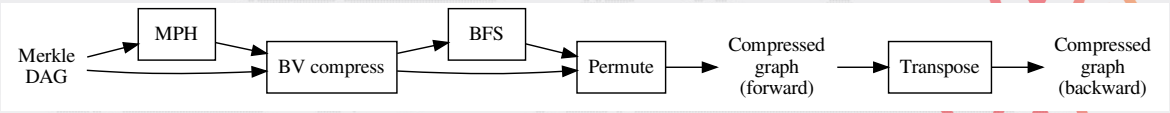
Adjacency lists

Node	Outd.	Successors
…	…	…
15	11	13,15,16,17,18,19,23,24,203,315,1034
16	10	15,16,17,22,23,24,315,316,317,3041
17	0	
18	5	13,15,16,17,50
…	…	…

Copy lists

Node	Ref.	Copy list	Extra nodes
…	…	…	…
15	0		13,15,16,17,18,19,23,24,203,315,1034
16	1	01110011010	22,316,317,3041
17			
18	3	11110000000	50
…	…	…	

Graph compression pipeline

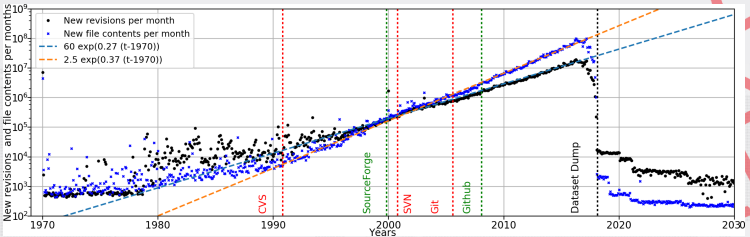


- MPH: minimal perfect hash, mapping Merkle IDs to 0..N-1 integers
- BV compress: Boldi-Vigna compression (based on MPH order)
- BFS: breadth-first visit to renumber
- Permute: update BV compression according to BFS order

(Re)establishing locality

- Key for good compression is a node ordering that ensures locality and similarity
- Which is very much *not* the case with Merkle IDs, … but is the case *again* after BFS reordering

Software provenance and evolution



Key findings

- The amount of original commits in public code doubles every ~30 months and has been doing so for 20+ years; original source code files double every ~22 months
- It is possible to trace the provenance of source code artifacts at this scale in a compact relational model via the notion of isochrone graphs.

Rousseau, Di Cosmo, Zacchiroli

Software Provenance Tracking at the Scale of Public Source Code

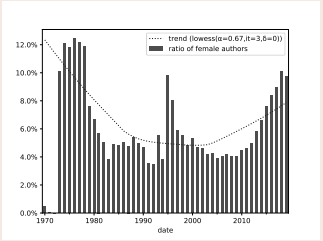
Empir. Softw. Eng. 25(4): 2930-2959 (2020)

Diversity, equity, and inclusion

Archived commit metadata contains public information that can be mined to study DEI traits of the global population of public code authors.

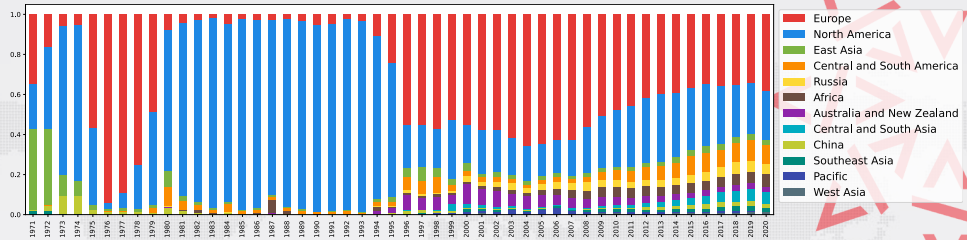
Gender gap — key findings

- Male authors contributed 92% of public code commits up to 2019.
- Female authors (and their commits) have grown stably for 15 years reaching 10% of yearly commits in 2019.
- The COVID-19 pandemic has reversed the trend (and it is not a coincidence!)



- Zacchiroli. *Gender differences in public code contributions: a 50-year perspective.* IEEE Software, 2021
- Rossi and Zacchiroli. *Worldwide gender differences in public code contributions […].* ICSE SEIS, 2022
- Casanueva et al. *The Impact of the COVID-19 Pandemic on Women's Contribution to Public Code.* (Under review.)

Diversity, equity, and inclusion (cont.)



Geographic gap — key findings

- Early decades of public code dominated by contributions from North America, followed by a period of alternating dominance between North America and Europe.
- Since then geographic diversity has increased constantly, with raising importance of contributions from Central and South America.
- Geo *and* Gender gap: the trend of increased female contributions is global, with the exception of some regions in Asia where it is either slower or flat.

Rossi and Zacchiroli. *Geographic diversity in public code contributions.* MSR 2022