



# FAIRSFair

Fostering Fair Data Practices in Europe



# Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

## Software as a first class output in a FAIR ecosystem

Morane Gruenpeter - Inria, Software Heritage

WoSSS21

6/10/2021



FAIRSFair "Fostering FAIR Data Practices in Europe" has received funding from the European Union's Horizon 2020 project call H2020-INFRAEOSC-2018-2020 Grant agreement 831558



# Outline

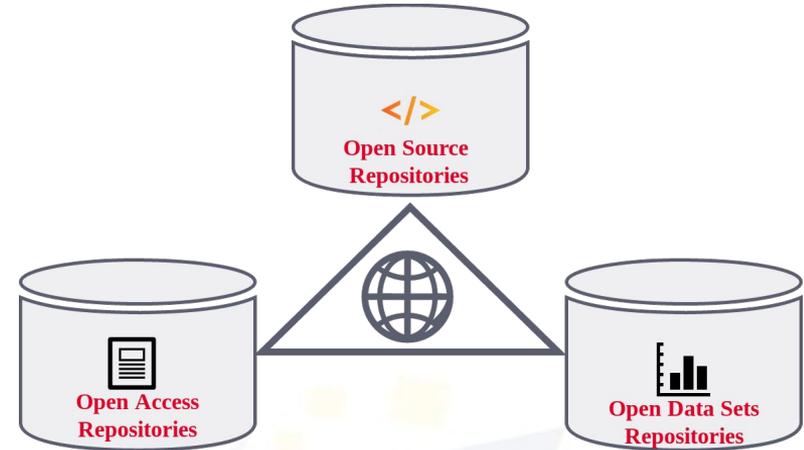
- Introduction: Research Software a first class output
- Why (research software) archiving is important?
- Software Heritage: the universal source code archive
- Software in a FAIR ecosystem
- FAIR4RS Working Group



# Software in Research: A pillar of Open Science

Multiple facets, it can be seen as:

- a **tool**
- a research **outcome** or result
- **the object** of research



*Three pillars of Open Science  
Gruenpeter, Software Heritage CC-BY 4.0 2019*



**Tim Berners-Lee** i the **World Wide Web, 1989**,at CERN



**Margaret Hamilton** **Apollo 11 Guidance Computer (~60.000 lines), 1969**

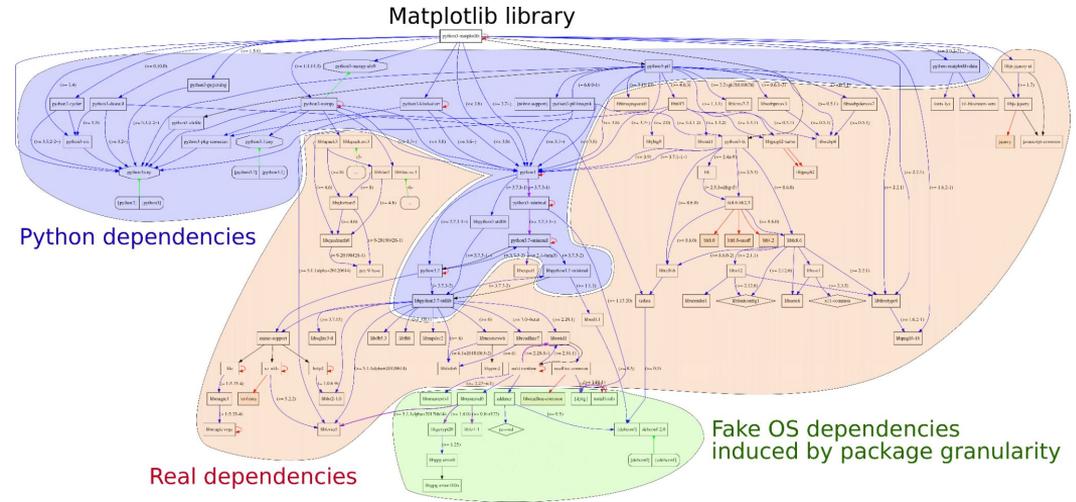
# Software Source Code is special (not just data)

## 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



(Alliez, et al. 2019). <http://doi.org/10.1109/MCSE.2019.2949413>. hal-02135891v2

# Software Source Code human readable and executable knowledge

Full width
Home Development Documentation Donate login

Software Heritage  
Archive

---

Features

- Search
- Downloads
- Save code now
- Help

```

52
53 # THE MASTER IGNITION ROUTINE IS DESIGNED FOR USE BY THE FOLLOWING LEM PROGRAMS: P12, P40, P42, P61, P63.
54 # IT PERFORMS ALL FUNCTIONS IMMEDIATELY ASSOCIATED WITH APS OR DPS IGNITION: IN PARTICULAR, EVERYTHING LYING
55 # BETWEEN THE PRE-IGNITION TIME CHECK -- ARE WE WITHIN 45 SECONDS OF TIG? -- AND TIG + 26 SECONDS, WHEN DPS
56 # PROGRAMS THROTTLE UP.
57 #
58 # VARIATIONS AMONG PROGRAMS ARE ACCOMMODATED BY MEANS OF TABLES CONTAINING CONSTANTS (FOR AVEGEXIT, FOR
59 # WAITLIST, FOR PINBALL) AND TCF INSTRUCTIONS. USERS PLACE THE ADRES OF THE HEAD OF THE APPROPRIATE TABLE
60 # (OF P61TABLE FOR P61LM, FOR EXAMPLE) IN ERASABLE REGISTER 'WHICH' (E4). THE IGNITION ROUTINE THEN INDEXES BY
61 # WHICH TO OBTAIN OR EXECUTE THE PROPER TABLE ENTRY. THE IGNITION ROUTINE IS INITIATED BY A TCF BURNBABY,
62 # THROUGH BANKJUMP IF NECESSARY. THERE IS NO RETURN.
63 #
64 # THE MASTER IGNITION ROUTINE WAS CONCEIVED AND EXECUTED, AND (NOTA BENE) IS MAINTAINED BY ADLER AND EYLES.
65 #
66 #           HONI SOIT QUI MAL Y PENSE
67 #
68 # *****
69 #           TABLES FOR THE IGNITION ROUTINE
70 # *****
71 #
72 #           NOLI SE TANGERE
73 #
74 P12TABLE      VN      0674      # (0)
75              TCF      ULLGN0T    # (1)
76              TCF      COMFAIL3    # (2)
77              TCF      GOCUTOFF    # (3)
78              TCF      TASKOVER    # (4)
79              TCF      P12SP0T    # (5)
80              DEC      0           # (6)      NO ULLAGE
81              EBANK=  WHICH
82              2CADR  SERVEXIT      # (7)
83
84              TCF      DISPCHNG    # (11)
85              TCF      WAITABIT    # (12)
86              TCF      P12IGN      # (13)
87
88 P40TABLE      VN      0640      # (0)
89              TCF      ULLGN0T    # (1)
90              TCF      COMFAIL4    # (2)
91              TCF      G0POST      # (3)
92              TCF      TASKOVER    # (4)
93              TCF      P40SP0T    # (5)
          
```

↻ Permalinks

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

Harold Abelson, 1985  
Structure and Interpretation of Computer Programs (1st ed.),

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

Len Shustek, 2006  
Computer History Museum

[Go to the code!](#)

# Why are we here?

## A plurality of needs in the scholarly ecosystem

### Researchers

- **archive and reference** software used and created in articles
- **find** useful software
- **get credit** for developed software
- **verify/reproduce/improve** results

### Laboratories/teams

- **track** software contributions
- **produce** reports
- **maintain** web page

### Research Organization

- know its **software assets** for:
  - technology transfer,
  - impact metrics,
  - strategy



Why (research software) archiving is important?



Source code is fragile

# Source code can be destroyed

## Google Kills Off Google Code

Natasha Lomas @riptari / 10:58 AM GMT+1 • March 13, 2015

1.4 million projects

Posted: Thursday, March 12, 2015

 377

 Tweet 1,210

 Like 404

When we started the Google Code project hosting service in 2006, the world of project hosting was limited. We were worried about reliability and stagnation, so we took action by giving the open source community another option to choose from. Since then, we've seen a wide variety of better project hosting services such as GitHub and Bitbucket bloom. Many projects moved away from Google Code to those other systems. To meet developers where they are, we ourselves migrated nearly a thousand of our own open source projects from Google Code to [GitHub](#).

As developers migrated away from Google Code, a growing share of the remaining projects were spam or abuse. Lately, the administrative load has consisted almost exclusively of abuse management. After profiling non-abusive activity on Google Code, it has become clear to us that the service simply isn't needed anymore.

Beginning today, we have disabled new project creation on Google Code. We will be shutting down the service about 10 months from now on January 25th, 2016. Below, we provide links to migration tools designed to help you move your projects off of Google Code. We will also make ourselves available over the next three months to those projects that need help migrating from Google Code to other hosts.

- March 12, 2015 - New project creation disabled.
- August 24, 2015 - The site goes read-only. You can still checkout/view project source, issues, and wikis.
- January 25, 2016 - The project hosting service is closed. You will be able to download a tarball of project source, issues, and wikis. These tarballs will be available throughout the rest of 2016.

Google will continue to provide Git and Gerrit hosting for certain projects like Android and Chrome. We will also continue maintaining our mirrors of projects like Eclipse, kernel.org and others.

# In science, reproducibility requires long-term access



**Gabriel Altay**  
@gabrielaltay

Just realized [@Bitbucket](#) disabled all mercurial repositories when the [@asclnet](#) informed me that a link associated with an old paper of mine was down. Thought all was lost, but someone archived all the repos! very classy move by [@octopus\\_net](#) and [@SWHeritage](#).

[Traduire le Tweet](#)

1:48 AM · 31 août 2020 · Twitter Web App

## Sunsetting Mercurial support in Bitbucket

April 21, 2020 | 3 min read



Denise Chan

[Update Aug 26, 2020] All hg repos have now been disabled and cannot be accessed.

[Update July 1, 2020] Today, mercurial repositories, snippets, and wikis will turn to read-only mode. After July 8th, 2020 they will no longer be accessible.

Source: [BitBucket blog](#)

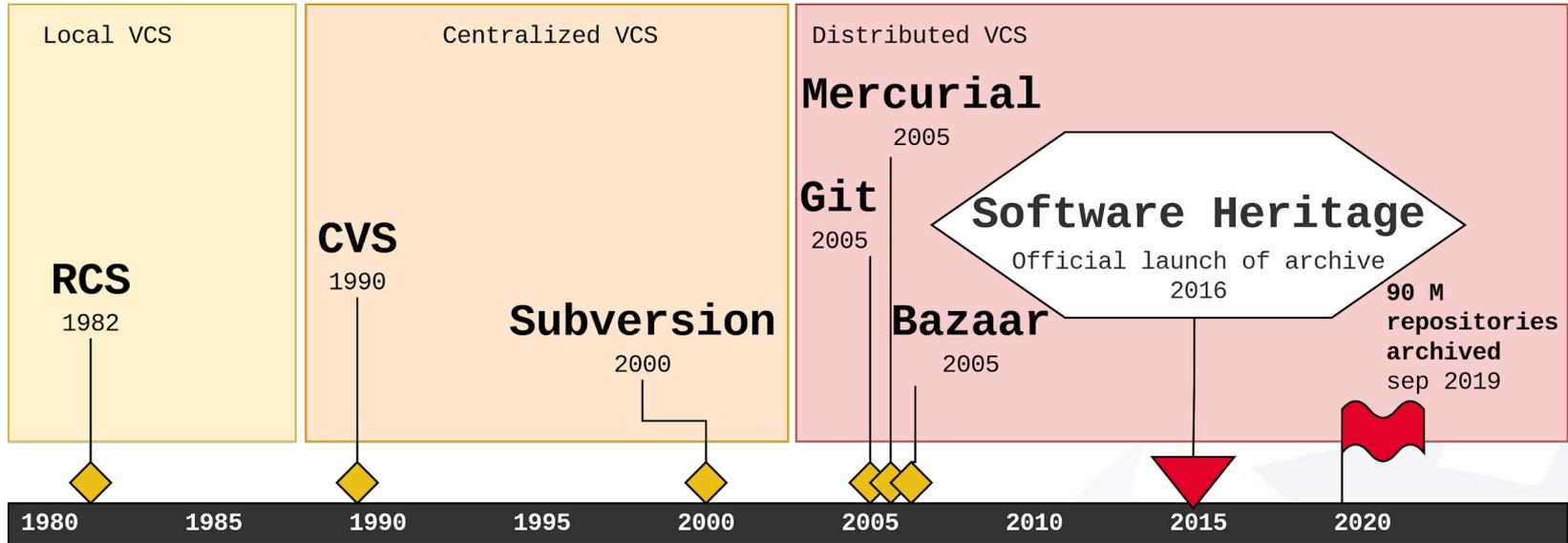
250,000 repos

## Hosting your open-source project

- On a free, publicly available platform is fine.
- But you have to prepare for the platform shutdown (you need a plan B).



# Version control system (VCS) history



- records changes made to a (set of) source code file (s)
- allows to operate on versions: diff/merge/fork/recover etc.
- essential tool for software development



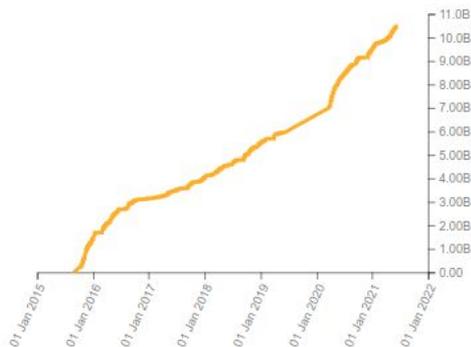
# Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

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

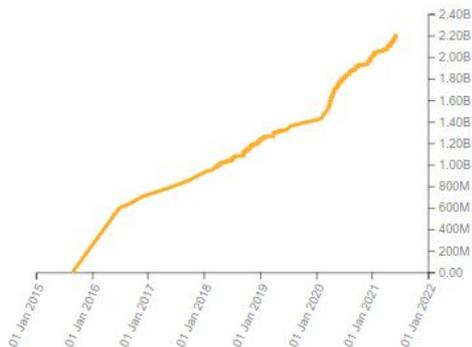
## Source files

10545239307



## Commits

2215028000



## Projects

161168065



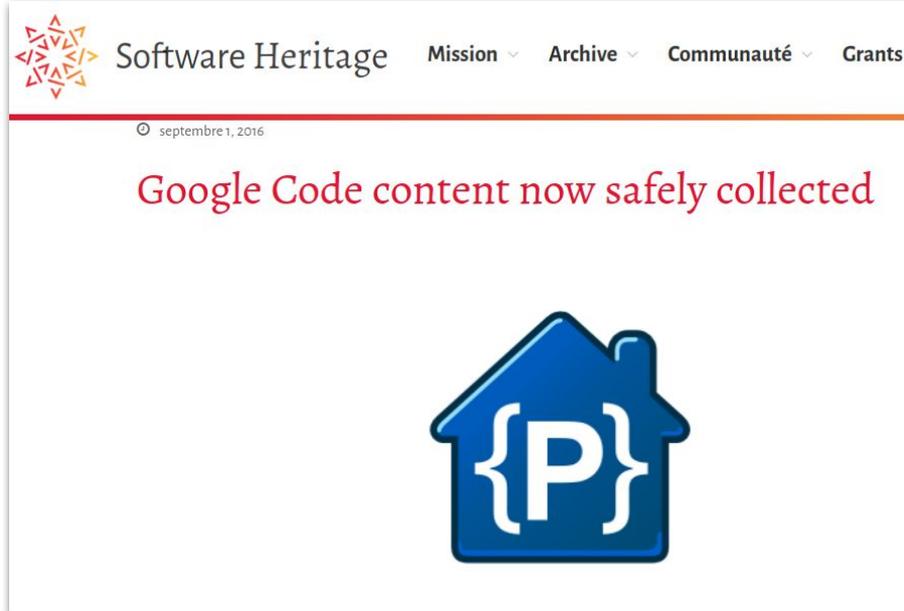
Source: Software Heritage (June 2021)

[Visit the archive](#)

# Archiving software



# Rescuing software



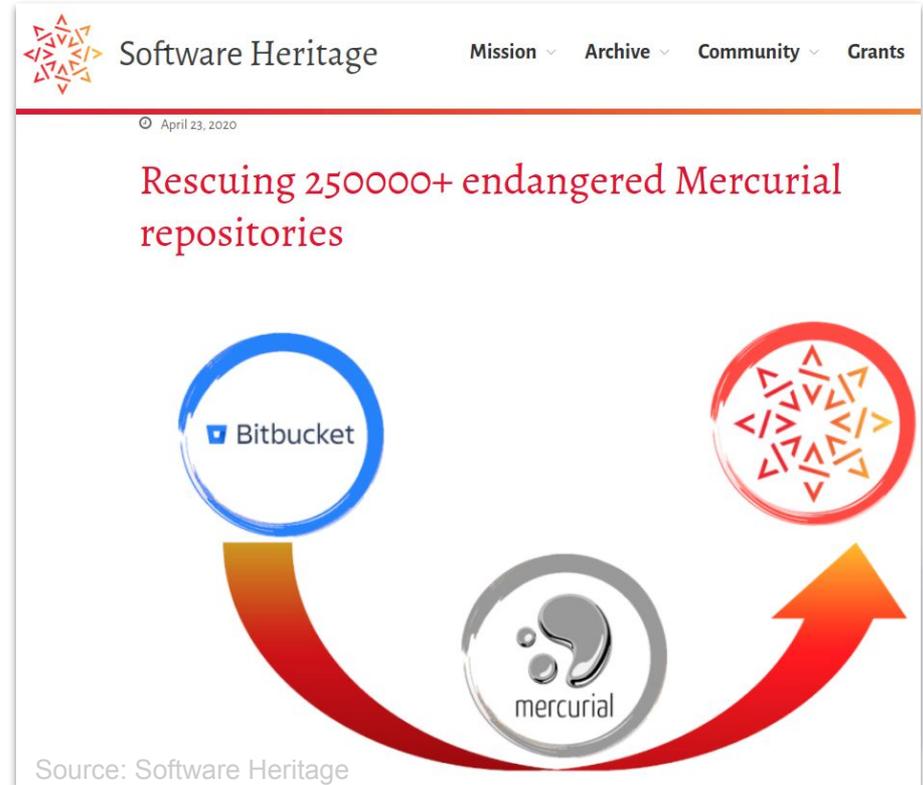
Software Heritage Mission Archive Communauté Grants

septembre 1, 2016

## Google Code content now safely collected



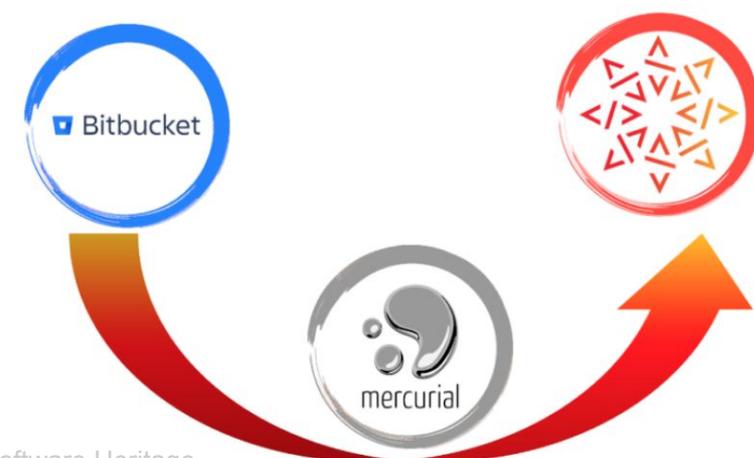
Source: Software Heritage



Software Heritage Mission Archive Community Grants

April 23, 2020

## Rescuing 250000+ endangered Mercurial repositories

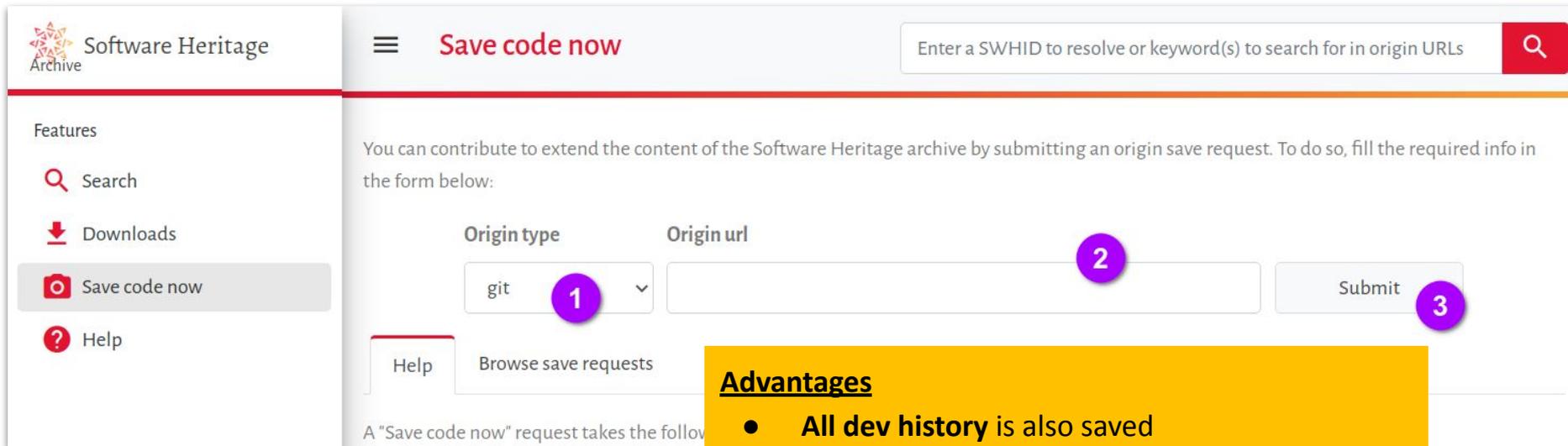


The diagram illustrates the rescue process. A blue circle containing the Bitbucket logo is connected by a red arrow to a grey circle containing the Mercurial logo. A second red arrow points from the Mercurial logo to a red circle containing the Software Heritage logo, indicating the final destination of the rescued repositories.

Source: Software Heritage

# Saving code for everyone!

<https://archive.softwareheritage.org/save/>



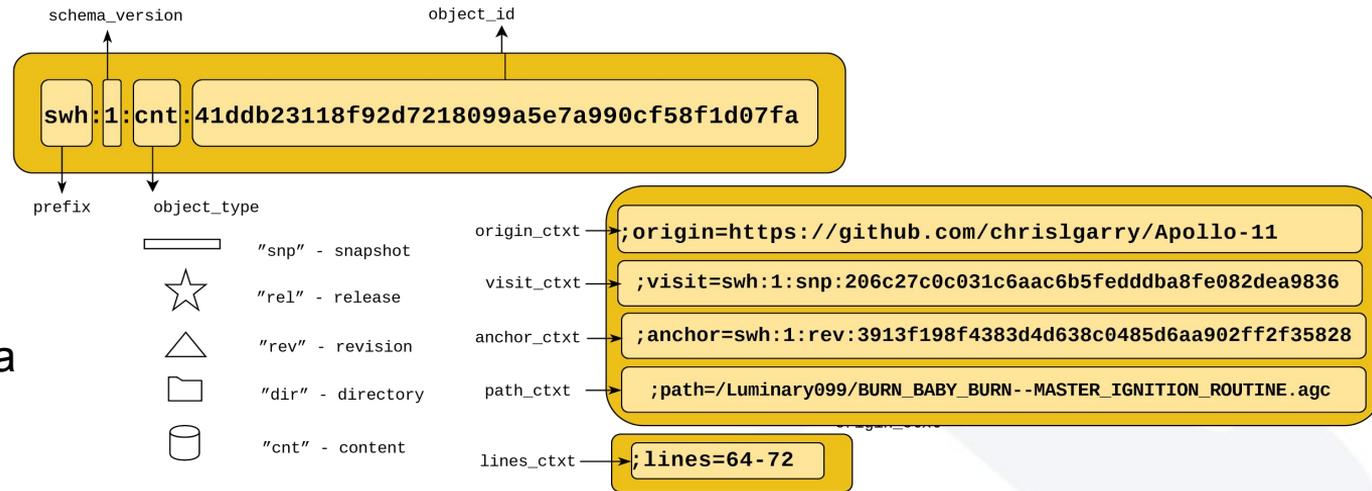
The screenshot shows the 'Save code now' page on the Software Heritage website. The page has a navigation sidebar on the left with options: Search, Downloads, Save code now (highlighted), and Help. The main content area features a search bar at the top right with the placeholder text 'Enter a SWHID to resolve or keyword(s) to search for in origin URLs'. Below the search bar, there is a heading 'Save code now' and a paragraph explaining that users can contribute to the archive by submitting an origin save request. The form consists of two main fields: 'Origin type' and 'Origin url'. The 'Origin type' field is a dropdown menu currently set to 'git', with a purple circle '1' next to it. The 'Origin url' field is a text input box with a purple circle '2' next to it. To the right of the 'Origin url' field is a 'Submit' button with a purple circle '3' next to it. Below the form, there is a 'Help' button and a link to 'Browse save requests'. At the bottom of the page, there is a yellow box containing the heading 'Advantages' and a list of three bullet points.

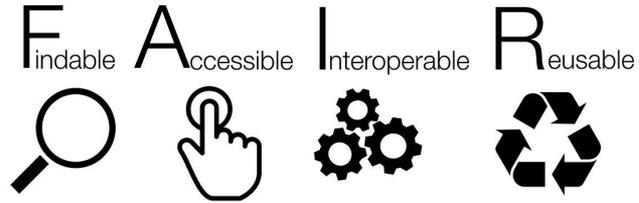
## Advantages

- **All dev history** is also saved
- Urls from **different platforms** are accepted
- PID to **reference** specific pieces of code (even algorithms)

# The SWHID: An intrinsic identifier to reference source code

- **Intrinsic:** compute a unique **digital fingerprint**
- **decentralised:** do not need a registry, only agreement on a standard
- **cryptographically strong** identifiers





# Building bridge between communities

Software development communities  
&  
Research Software communities



Created by Jolij Fish  
from Noun Project



Created by Vectors Point  
from Noun Project

# Software is not just another type of data

## *Recommendation n°5 :*

*Recognise that FAIR guidelines will require **translation for other digital objects** and support such efforts.*

**2019: ‘Six Recommendations for Implementation of FAIR Practice’**

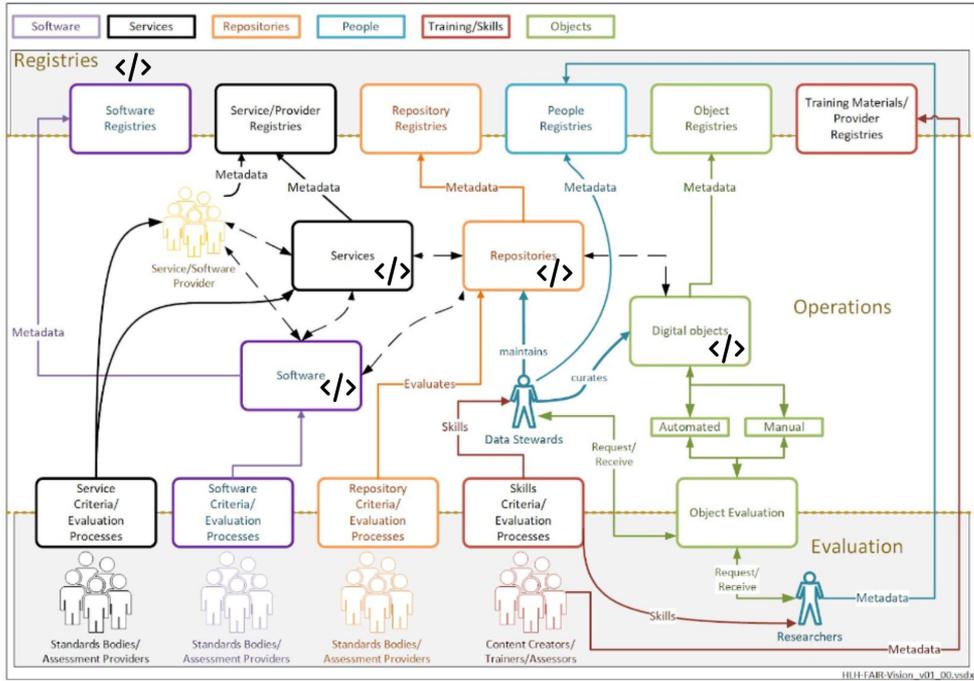
([FAIR Practice TF, 2020](#))

## *Recommendation n° 2 :*

*Make sure **the specific nature of software** is recognized and not considered as “just data” particularly in the context of discussion about the notion of FAIR data.*

**2019: the Opportunity Note** by the French national Committee for Open Science's Free Software and Open Source Project Group  
([Clément-Fontaine, 2019](#))

# Software in the FAIR ecosystem



Ecosystem components, to highlight the software roles in the Ecosystem, the symbol `</>` was added (Orig

[10.5281/zenodo.4095092](https://10.5281/zenodo.4095092)

## Webinar FAIR + Software: decoding the principles (Nov 2020)

October 16, 2020

Project milestone Open Access

### M2.15 Assessment report on 'FAIRness of software'

Gruenpeter, Morane; Di Cosmo, Roberto; Koers, Hylke; Herterich, Patricia; Hooft, Rob; Parland-von Essen, Jessica; Tana, Jonas; Aalto, Tero; Jona, Sarah

Software has an important place in academia and as such it has an important place in the FAIR ecosystem. Software can be used throughout the research process; however it can also be an outcome of the research process. Distinguishing between these different roles is essential for any assessment of the 'FAIRness of software'.

This is the first milestone of the FAIRSFair project focused specifically on software as a digital object. In this report we discuss the state-of-the-art of software in the scholarly ecosystem in general and in the FAIR literature in particular. We identify the challenges of different stakeholders when it comes to finding and reusing software. Furthermore, we present an analysis of nine resources that call for the recognition of software in academia and that present guidelines or recommendations to improve its status - either by becoming more FAIR or by improving the curation of software in general. With this analysis we demonstrate to what extent each of the FAIR principles is seen as relevant, achievable and measurable, and in what sense it benefits software artifacts. Finally, we present 10 high-level recommendations for organizations that seek to define FAIR principles or other requirements for research software in the scholarly domain.

Feedback and suggestions will be most welcome as comments on the public Google Doc version of this report <https://docs.google.com/document/d/1yvDLSP6oH3XozV4cJtThzGNHkseCBdvmxfuRDLB60/edit?usp=sharing>

Project Title	Fostering FAIR Data Practices in Europe
Project Acronym	FAIRSFair
Grant Agreement No.	881138
Instrument	H2020-INFRAEOSC-2018-4
Topic	INFRAEOSC-01-2018-2019 Support to the EOSC Governance
Start Date of Project	31st March 2019
Duration of Project	36 months
Project Website	www.fair4eu.eu

### M2.15 ASSESSMENT REPORT ON 'FAIRNESS OF software'

Work Package	WP2 - FAIR practices: semantics, interoperability and services
Lead Author (Org)	Marcus Koers (DFKI)
Contributing Author(s) (Org)	Roberto Di Cosmo (DFKI), Hylke Koers (DFKI), Patricia Herterich (DFKI), Rob Hooft (DFKI), Jessica Parland-von Essen (DFKI), Jonas Tana (DFKI), Tero Aalto (DFKI), Sarah Jona (DFKI)
Doc Date	30/10/2020
Date	16/10/2020

# FAIR for Research Software (FAIR4RS) Working Group

## Main objective

Defining **FAIR principles for research software**

## Steering committee and WG chairs:

Morane Gruenpeter, Paula A. Martinez, Carlos Martinez, Michelle Barker, Daniel S. Katz, Leyla Garcia, Neil Chue Hong, Fotis Psomopoulos and Jennifer Harrow

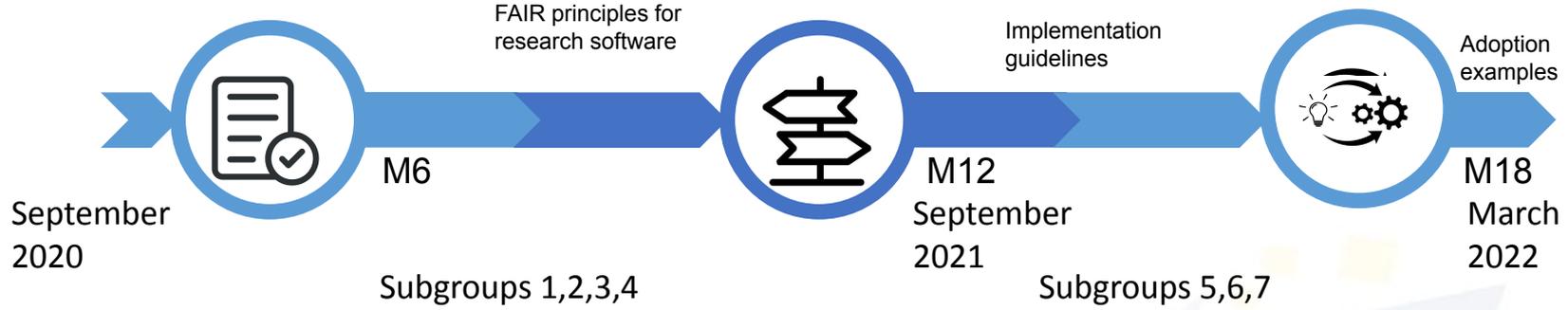
- Acknowledging the ~228 members and contributors of the FAIR for Research Software working group #FAIR4RS



[Join the WG](#)

Endorsed by RDA  
Sep 2020

# #FAIR4RS timeline



# Development of the FAIR4RS Principles

- **Intent and methods of the FAIR Guiding Principles taken as starting point:**
  - “maximize the added-value gained by contemporary, formal scholarly digital publishing”
  - “to ensure transparency, reproducibility, and reusability.”
- **The FAIR Principles are aspirational, and FAIR is not binary**
  - The aim of FAIR (and FAIR) metrics is to show progress to increasing FAIRness
- **Software encompasses many forms, which may benefit different users**
  - Source code is often the most useful form to understand the software, and the easiest form to apply the FAIR4RS Principles.
- **Many software engineering practices are relevant to the FAIR4RS Principles**
  - For instance: localization can improve findability, design patterns can improve interoperability, and documentation and encapsulation can improve reusability.
  - Nevertheless, while important more generally for producing high quality software, they are best addressed separately from (but as a complement to) the FAIR4RS Principles.

# FAIR Principles for Research Software

**Findable: Software, and its associated metadata, is easy to find for both humans and machines.**

F1. Software is assigned a globally unique and persistent identifier

- F1.1. Different components of the software are assigned distinct identifiers representing different levels of granularity
- F1.2. Different versions of the same software are assigned distinct identifiers

F2. Software is described with rich metadata

F3. Metadata clearly and explicitly include the identifier of the software they describe

F4. Metadata are FAIR and are searchable and indexable

**Accessible: Software, and its metadata, is retrievable via standardized protocols.**

A1. Software is retrievable by its identifier using a standardized communications protocol

- A1.1. The protocol is open, free, and universally implementable
- A1.2. The protocol allows for an authentication and authorization procedure, where necessary

A2. Metadata are accessible, even when the software is no longer available

**Interoperable: Software interoperates with other software through exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards.**

I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards

I2. Software includes qualified references to other objects

**Reusable: Software is both usable (it can be executed) and reusable (it can be understood, modified, built upon, or incorporated into other software).**

R1. Software is described with a plurality of accurate and relevant attributes

- R1.1. Software is given a clear and accessible license
- R1.2. Software is associated with detailed provenance

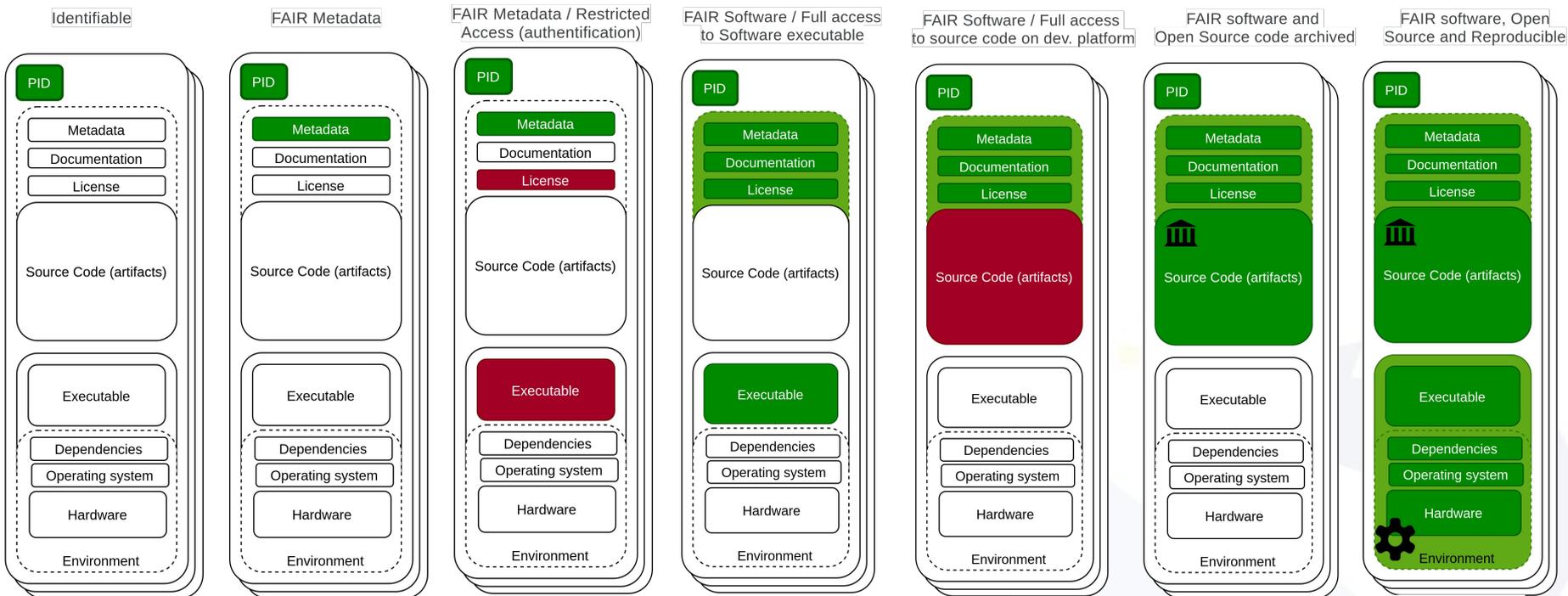
R2. Software includes qualified references to other software

R3. Software meets domain-relevant community standards

*FAIR4RS WG. (2021, June). FAIR Principles for Research Software*

# Beyond FAIR: FAIR is not the end goal

[10.15497/RDA00065](https://doi.org/10.15497/RDA00065)



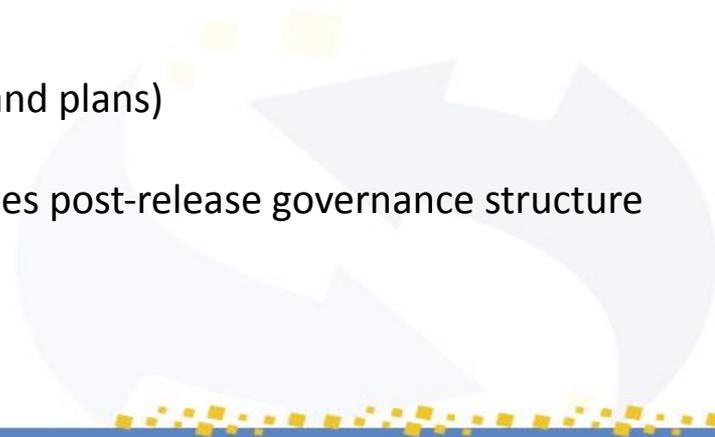
(Katz et al. 2020) arXiv:2101.10883

# New subgroups

Three new subgroups now starting

- **Subgroup 5: Adoption guidelines**
  - Identify, create, review existing resources that facilitate the adoption of FAIR4RS principles
- **Subgroup 6: Adoption support**
  - Identify & start to work with organisations following FAIR4RS guidelines (or willing to do so)
  - Stimulate adoption of FAIR4RS guidelines
  - Document & share examples of FAIR4RS adoption (and plans)
- **Subgroup 7: Governance**
  - Create communications plan and content that clarifies post-release governance structure

Join one or multiple subgroups via [the form](#)



## Wrap up

1. **Archive** source code in [Software Heritage](#)
2. **Join** the [FAIR4RS](#) Working Group
  - a. receive updates
  - b. contribute to the subgroups work
  - c. discuss the FAIR definition for research software
3. **Adopt** good practices to develop FAIR software
4. **Spread the word** and let's start **recognizing software in academia**



# Thank you!

Keep in touch: [morane@softwareheritage.org](mailto:morane@softwareheritage.org)

 @moraneottilia, @SWHeritage

 [s://www.fairsfair.eu/fairsfair-newletters/](https://www.fairsfair.eu/fairsfair-newletters/)

<https://www.softwareheritage.org/newsletter/>

