Report from the Task Force on Scholarly Infrastructures for Research Software (SIRS)

Roberto Di Cosmo

Director, Software Heritage

Chair of the SIRS TF

February 11th, 2021

RDA VP 17 – Software Source Code IG

Introduction

Roberto Di Cosmo (roberto@dicosmo.org https://www.dicosmo.org)

Computer Science professor in Paris, now working at INRIA

- *30 years* research (Theor. CS, Programming, Software Engineering, Erdos #: 3)
- *20 years* Free and Open Source Software
- *10 years* building and directing structures for the common good

- 1999: DemoLinux (first live GNU/Linux distribution)
- 2007: Free Software Thematic Group in Systematic
  - 150 members, 40+ projects, 200Me
- 2010: Irill, research on free software
- 2015: Software Heritage
- 2018: National Committee on Open Science, France

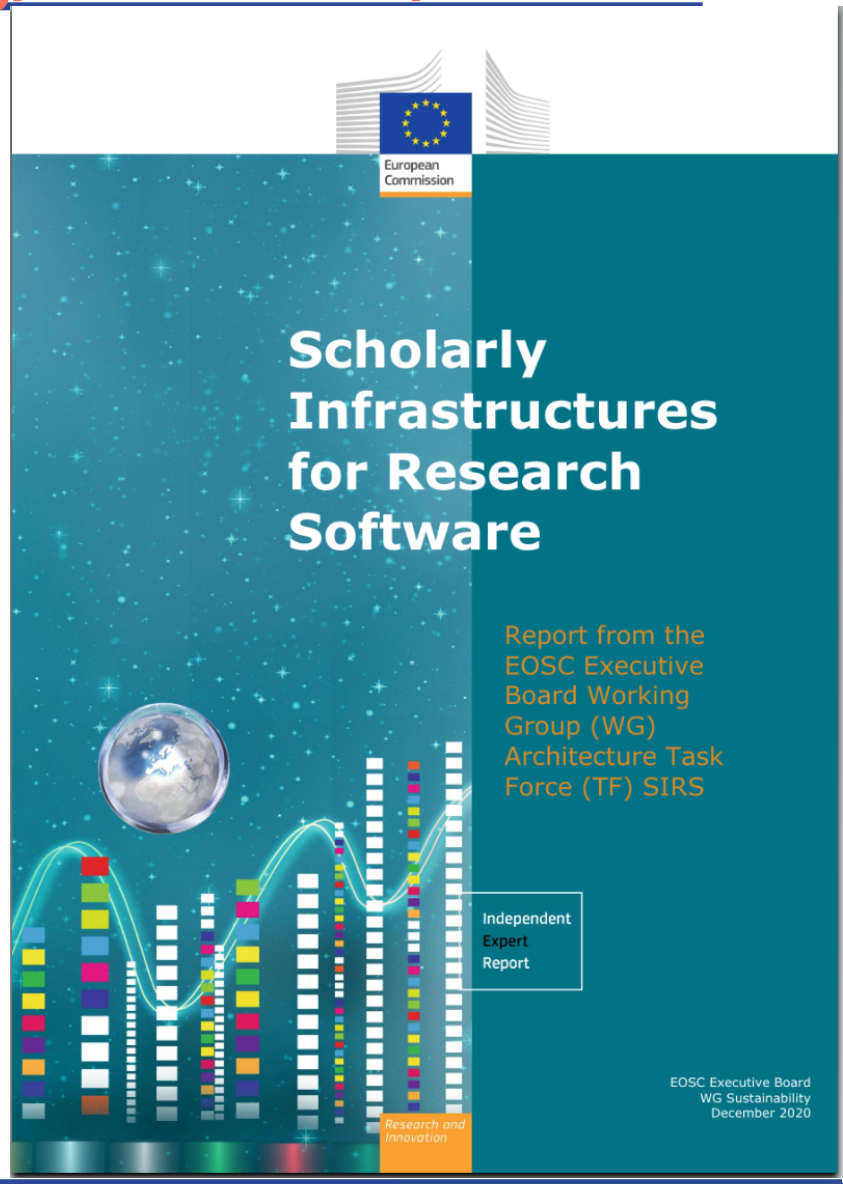
EOSC SIRS report (12/2020) doi.org/10.2777/28598

Chair: Roberto Di Cosmo, Software Heritage

Co-Chair: José Benito Gonzalez Lopez, Zenodo

- Focus on **Software Source Code**
- Four Pillars **Archive, Reference, Describe, Credit**
- State of the Art
  - Best Practices & Open Problems
  - Cross Cutting Concerns
- The Road ahead
  - Requirements & Criteria
  - 13 Workflows / Use Cases examples
- Recommendations
  - Standards & Tools
  - Policy recommendations
  - Long term perspectives

- **Archives**
  - HAL
  - Software Heritage
  - Zenodo
- **Publishers**
  - Dagstuhl
  - eLife
  - IPOL
- **Aggregators**
  - OpenAIRE
  - scanR
  - swMATH



SIRS Focus: software *source code*

"*Source code provides a view into the mind of the designer*" Len Shustek, 2006

"*[…] aware of the many difficult challenges that need to be tackled when one tries to ensure that a given **executable** or a full software system can be reliably run again, enabling **full reproducibility** of research results, as well as of the complex organizational, economic, and strategy issues that need to be addressed for its **sustainability**"*

"***The focus of the work of this TF is different**, as we have on purpose addressed **only software source code** in the world of research, for two main reasons:"*

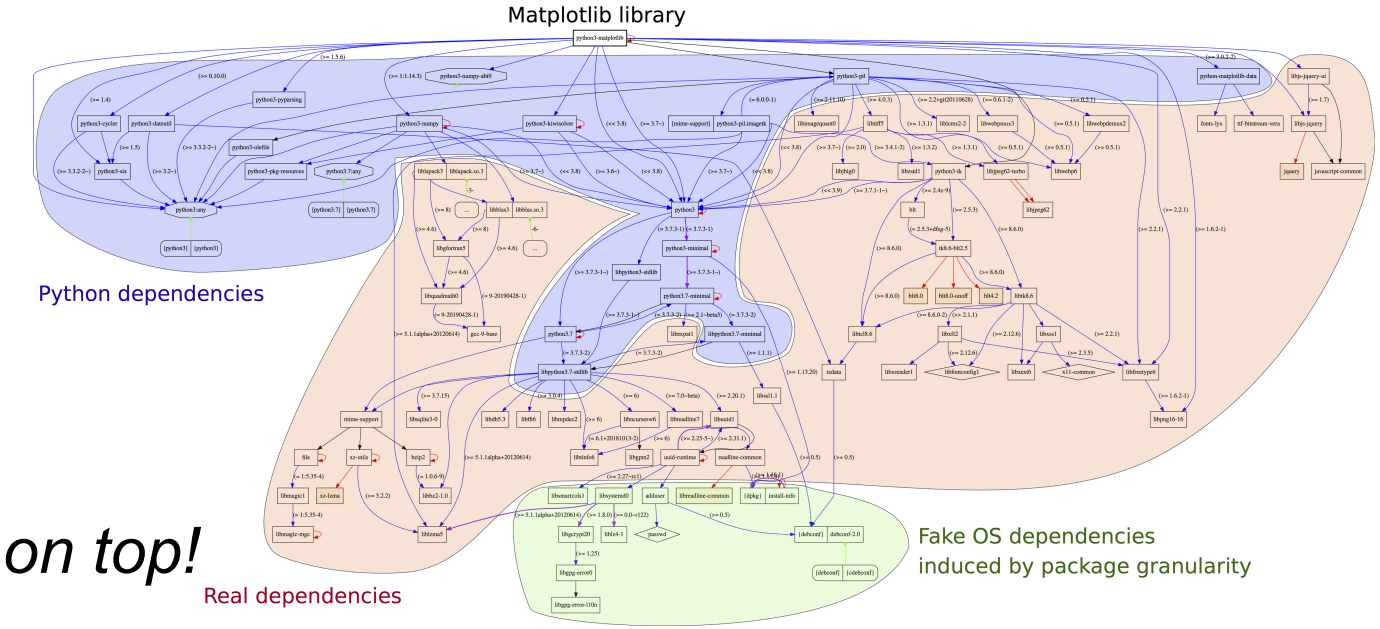
- Source code is "*human readable knowledge, and **embodies precious technical and scientific information** that cannot be extracted from the executables, and **that can be understood even when the corresponding executable can no longer be run**"*
- *"[…] handling software source code raises for scholarly infrastructures is a **significant challenge** by itself, […] it is easier to provide actionable recommendations by focusing on this first"*

Software Source Code is *special*

(it is not "just data")

> ***Programs must be written for people to read, and only incidentally for machines to execute.***
>
> *Harold Abelson, Structure and Interpretation of Computer Programs (1st ed.) 1985*

- Evolves over time: projects may last decades
  - *development history* key to its *understanding*
- Complex and sophisticated
  - millions of lines of code
  - large *web of dependencies*
  - sophisticated *developer communities*
- Research software is just *a thin layer on top!*
  - ***industry+communities drive standards***



Granularity, versioning, author roles...

(there's more to this than meets the eye)

Project: *"Inria created **OCaml** and **Scikit-learn**"*

Release: *"2D Voronoi Diagrams were introduced in **CGAL 3.1.0**"*

Precise state of a project: *"This result was produced using **commit 0064fbd...**"*

Code fragment: *"The core algorithm is in **lines 101 to 143 of the file parmap.ml** contained in the precise state of the project corresponding to **commit 0064fbd....**"*

Authors can have multiple roles:

Architecture, Management, Development, Documentation, Testing, ...

Four pillars: Archive, Reference, Describe, Credit

*« Software is a hybrid object in the world research as it is equally a driving force (as a tool), a result (as proof of the existence of a solution) and an object of study (as an artefact). » (**Opportunity note, GPLO, 2019**)*

*« the **FAIR Guiding Principles** for research do not fit [software source code] well, as they **were not designed for it** … » (FAIR does not fit publications either...)*

*« We focus here on **four key concrete issues** that need to be tackled to make software a first-class citizen in the scholarly world, and **where scholarly infrastructures play a prominent role**: »*

[Archive] ensure software artifacts *are not lost*

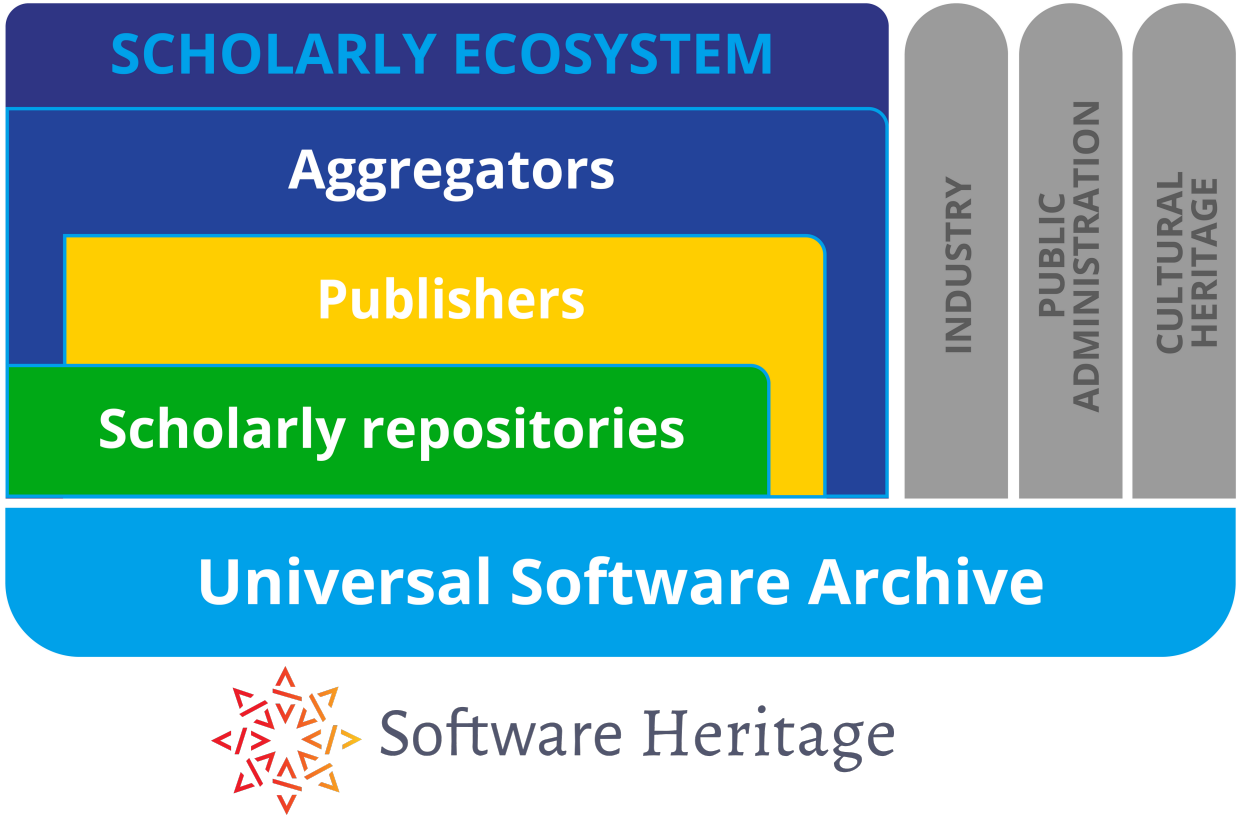
[Reference] ensure software artifacts *can be precisely identified*

[Describe] make it easy to *discover / find* software artifacts

[Credit] ensure *proper credit* is given *to authors*

*The EOSC SIRS Report collects the **key requirements** to address these issues, through and **open architecture of interconnected infrastructures**.*

Research Software Infrastructures: Overall Architecture



- **Scholarly ecosystem**
  - Aggregators collecting data from…
  - Academic publishers
  - Scholarly repositories
- **Software Heritage** connects with the global software development ecosystem

Short term recommendations

- Metadata standards & tools
- Generalizing the use of Persistent Identifiers (extrinsic & intrinsic)
- Ensuring *appropriate* credit is given *and measures are not misused*
- Strengthening interactions between archives, publishers & aggregators

- Strengthen key infrastructures sustainability and governance
- Identify resilient funding models

Metadata standard(s) for interoperability

Codemeta « *extension of the schema.org standard, extensive vocabulary designed to allow mapping other metadata vocabularies, embrionary community process* »

Vocabulary

Terms from Schema.org

Recognized properties for CodeMeta Code includes the following terms from https://schema.org. These terms are part of the CodeMeta specification and can be used without any prefix.

Property	Type	Description
codeRepository	URL	Link to the repository where the un-compiled, human readable code and related code is located (SVN, GitHub, CodePlex, institutional GitLab instance, etc.).
programmingLanguage	ComputerLanguage or Text	The computer programming language.
runtimePlatform	Text	Runtime platform or script interpreter dependencies (Example - Java v1, Python2.3, .Net Framework 3.0). Supersedes runtime.
targetProduct	SoftwareApplication	Target Operating System / Product to which the code applies. If applies to several versions, just the product name can be used.
applicationCategory	Text or URL	Type of software application, e.g. 'Game, Multimedia'.
applicationSubCategory	Text or URL	Subcategory of the application, e.g. 'Arcade Game'.
downloadUrl	URL	If the file can be downloaded, URL to download the binary.

Tools

Most fields are optional. Mandatory fields will be highlighted when generating Codemeta.

The software itself

Name

My Software

the software title

Description

My Software computes ephemerides and orbit propagation. It has been developed from early ´80.

Creation date

YYYY-MM-DD

First release date

YYYY-MM-DD

Software Package Data eXchange (SPDX) standard maintained by the **Linux Foundation**

Recognized reference for ***the list of software licences***.

Systems of Identifiers: extrinsic and intrinsic

- ***Extrinsic:*** *use a* ***register*** *to keep the correspondence between the identifier and the designated object*
  - Examples *before the digital era*: passport number, social security number, …
  - Examples *in the digital era*: DNS, Handle, ARK, DOI, …

- ***Intrinsic:*** *intimately bound to the designated object, no need for a register, only agreement on a* ***standard***
  - Examples *before the digital era*: chemical notation, musical notation, …
  - Examples *in the digital era*: cryptographic signatures, commit hashes, SWHID…

Read more at https://www.softwareheritage.org/2020/07/09/intrinsic-vs-extrinsic-identifiers/

Extrinsic systems of identifiers used for software

HAL-ID

Digital Object Identifier

WIKIDATA

Wiki Item identifier (Qxxx)

ASCL.net
Astrophysics Source Code Library

Extrinsic Systems of identifiers used for software (selection)

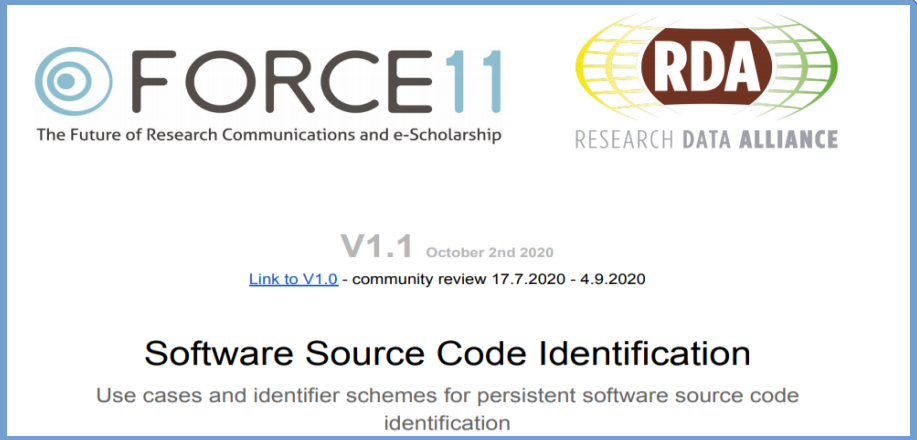
ARK
Archival Resource Key

Handle
Handle System identifiers

swMATH
an information service for mathematical software

RRID

~

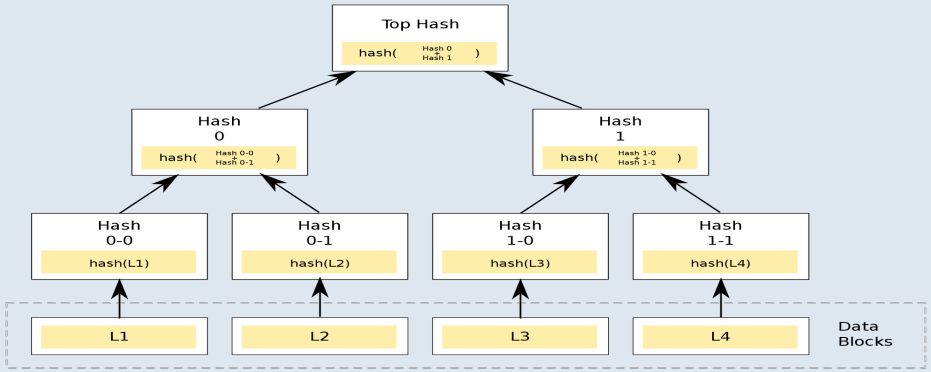


Read more at https://doi.org/10.15497/RDA00053

«We recommend that an inclusive approach is explored to guarantee that existing well-established extrinsic identifiers are taken into account.»

Intrinsic systems of identifiers for software

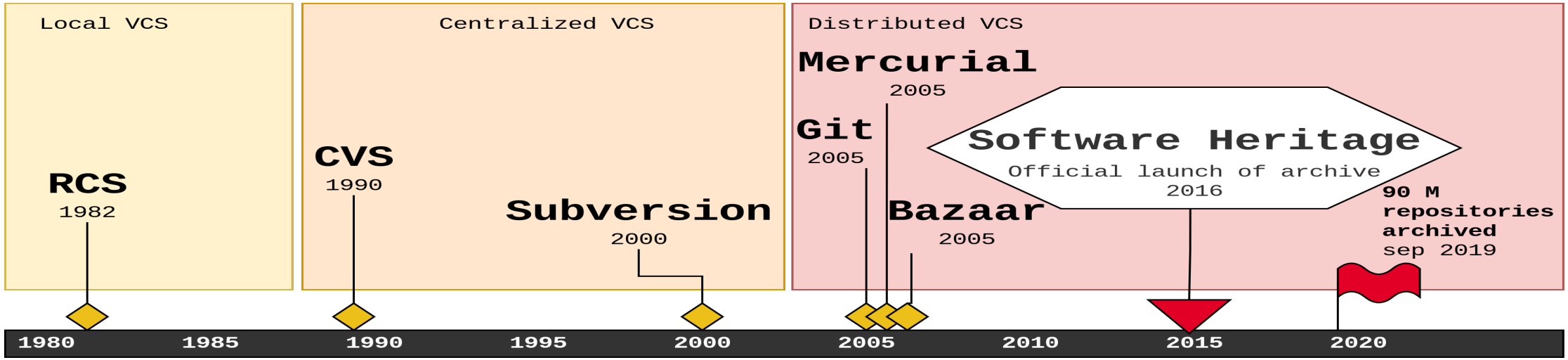
Ralph Merkle, 1987 « *A digital signature based on a conventional encryption function* »



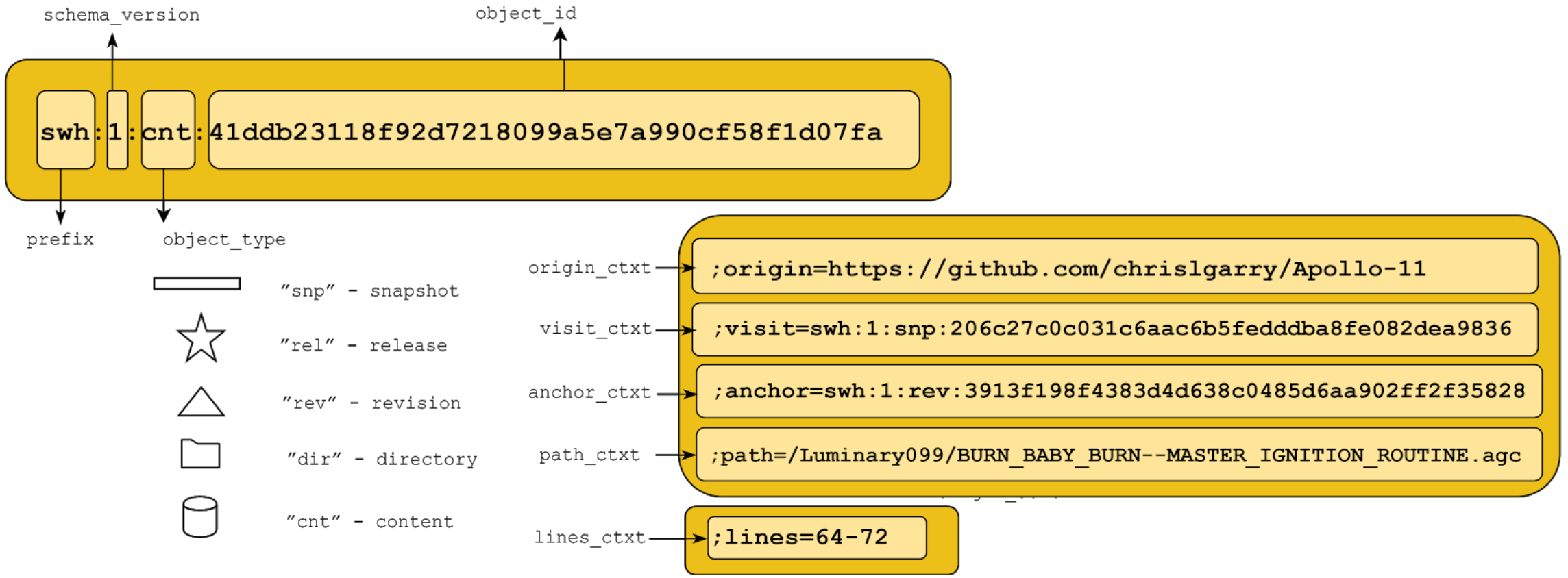
Blockchains

Distributed file systems

As of 2020
40+ million developers
140+ million repositories



SWHID: a standard for intrinsic software identifiers



Let's try it!

Included in SPDX 2.2 – Prefix « swh » registered with IANA – Wikipedia Property P6138

Use « SWHID intrinsic identifiers for all publicly available software source code »

SWHID: growing adoption in scholarly publishing

HAL software curated deposit workflow

Curated Archiving of Research Software Artifacts
International Journal of Digical Curation, 2020

Reference archive for swmath.org

swMATH
an information service for mathematical software

See *code* links, e.g. SemiPar package

IPOL (image processing)

- archive (deposit)
- reference
- BibLaTeX

eLife (life sciences)

- archive (save code now)
- reference

JTCAM (Mechanics)

- instructions for authors
- biblatex-software in journal LaTeX class

https://elifesciences.org/inside-elife/c5428dc9/elife-latest-our-commitment-to-software-preservation-and-reuse

Credit: Quality, Curation & Metrics

- ***Quality and curation (software quality is HARD, curation of metadata is easier)***
  - *"ensure that the **peer review process** also covers software source code, with the level of evaluation most appropriate for their field"*
  - *"develop a set of common guidelines for **moderation and curation** protocols"*
  - *"development of a set of standard **tools and workflows** […] to support and ease adoption of more sophisticated levels of review, like the ones implemented by Artifact Evaluation Committees"*
- ***Metrics***
  - *"should be open, verifiable, and shareable"*
  - *"**not reduced to simple numeric indicators**"*
  - *"include in the conversation […] the research community that will be directly impacted by the creation of these metrics"*

Development of tools and connectors (selection)

- ***Connectors*: scholarly repositories ↔ universal software archive**
  - *standards exist: development, deployment and maintenance (2 years horizon)*
- ***Tools and standards*: adapt publisher pipelines**
  - *standards exist: get involved to evolve them*
- ***Converters and adaptors*: ensure Codemeta can be exported and imported**
  - *standards exist: development, deployment and maintenance (2 years horizon)*
- ***Tools*: automation of source code archival and reference for publishers**
  - *standards do not exist: two pronged approach with a 4 years timeframe*

Long term recommendations

- Advanced technologies
  - Open plagiarism detection
  - Advanced search engines
- Integration with publications and data

- Common Infrastructures hosted by not-for-profit organizations
- **Open Source license by default**

Now it’s time to implement these recommendations!

Thank you

roberto@dicosmo.org @rdicosmo https://www.softwareheritage.org @swheritage