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

Roberto Di Cosmo

Director, Software Heritage

Chair of the SIRS TF

November 5th, 2020

EOSC Architecture Working Group

EUROPEAN OPE
SCIENCE CLOU

SIRS Task Force: Key facts

- Approved by EOSC Executive Board in June 2020
- ”Help Software become first-class citizen next to Publications and Data”
- Participants:
  - Nominations by the EOSC Architecture WG
  - 9 existing infrastructures engaged in Research Software management
- 14 Weekly meetings over the summer
- Stewardship
  - Chair: Roberto Di Cosmo, Software Heritage
  - Co-Chair: José Benito Gonzalez Lopez, Zenodo
- Support
  - Lonneke Schrijver, EOSC Secretariat

SIRS Task Force: The report in a nutshell

- 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 execu

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

volves over time: projects may last decades

development history key to its *understanding*

omplex and sophisticated

millions of lines of code

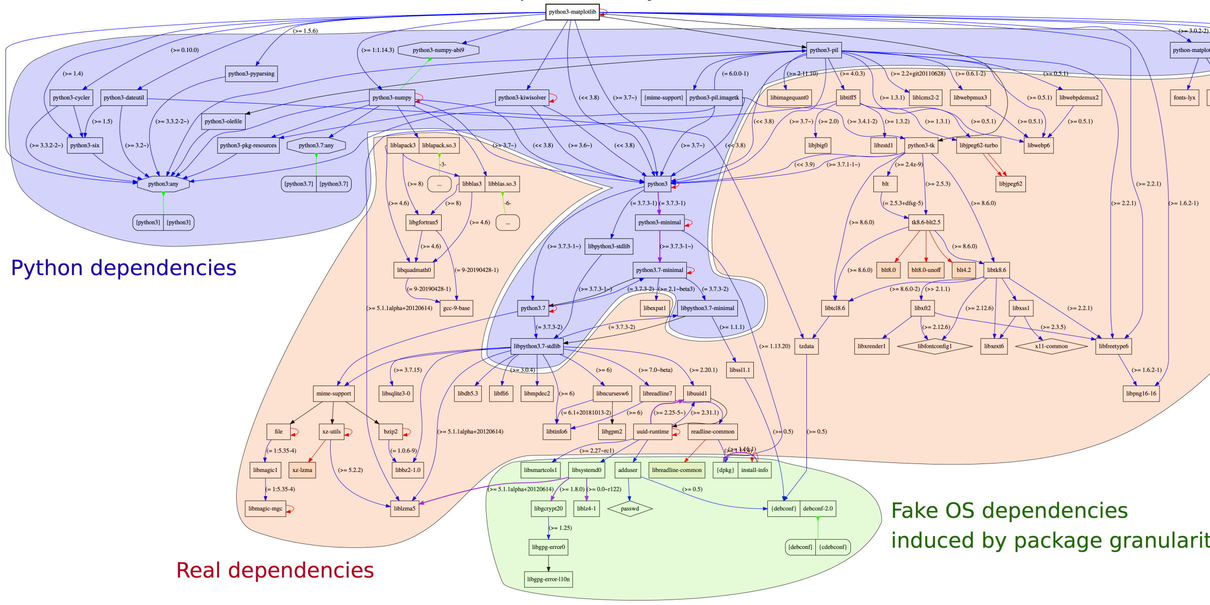
large *web of dependencies*

sophisticated *developer communities*

esearch software is just *a thin layer on top!*

industry+communities drive standards, build support layers

Matplotlib library



Granularity, versioning, author roles...

(there's more to this than meets the eye

ject: *“Inria created **OCaml** and **Scikit-learn**”*

ease: *“2D Voronoi Diagrams were introduced in **CGAL 3.1.0**”*

cise state of a project: *“This result was produced using **commit 0064fbd...**”*

le fragment: *“The core algorithm is in **lines 101 to 143 of the file parmap.ml** containec precise state of the project corresponding to **commit 0064fbd....**”*

hors can have multiple roles:

rchitecture, Management, Development, Documentation, Testing, ...

EUROPEAN OI
SCIENCE CLO

Four pillars: Archive, Reference, Describe, Credit

« the FAIR Guiding Principles for research do not fit [software source code] well, as they were not designed for it … »

*« 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**: »*

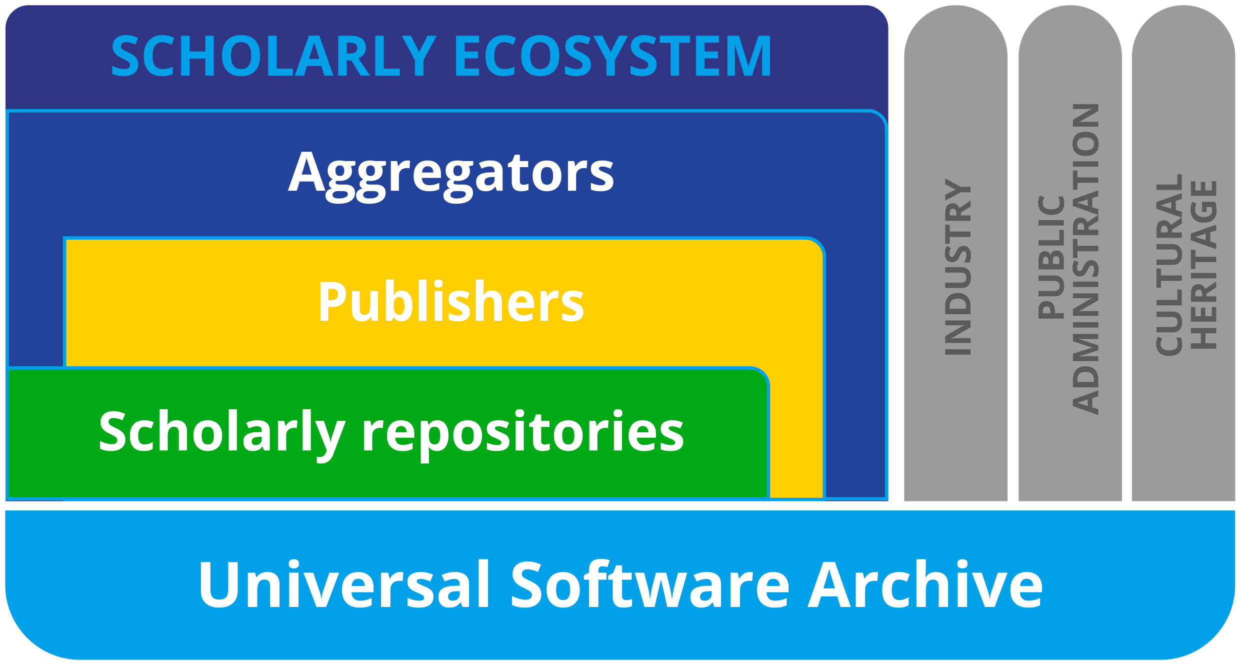
[Archive] ensure software artifacts *are not lost*

[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*

Research Software Infrastructures: Overall Architecture



- **Scholarly ecosystem**
- Aggregators collecting data from:
- Scholarly repositories
- Academic publishers

- Universal Software Archive (Software Heritage) connects with **the global software development ecosystem**

Short term recommendations

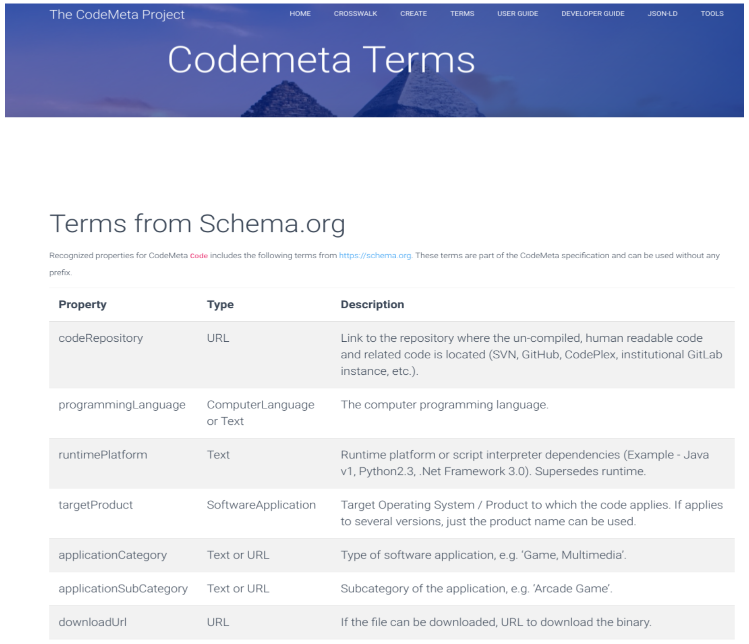
- Strengthening interactions between archives, publishers & aggregators
- Metadata standards & tools
- Generalizing the use of Persistent Identifiers (extrinsic & intrinsic)
- Ensuring *appropriate* credit is given *and measures are not misused*

- EOSC should get actively involved with key infrastructures to ensure their long term sustainability

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



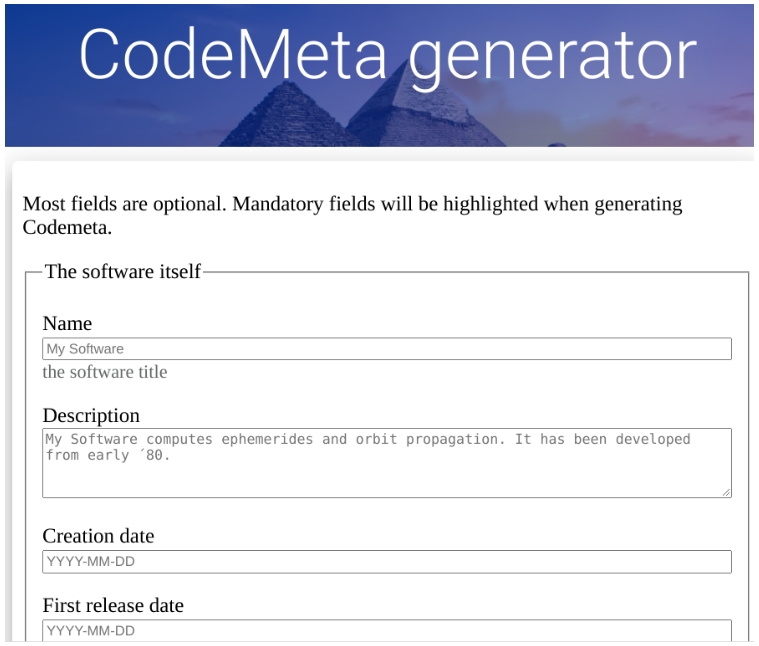
Codemeta Terms

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



CodeMeta generator

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...

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

EUROPEAN O
SCIENCE CLO

Extrinsic systems of identifiers used for software

HAL-ID

Digital Object Identifier

net
ysics Source Code Library

Extrinsic Systems of identifiers used for software (selection)

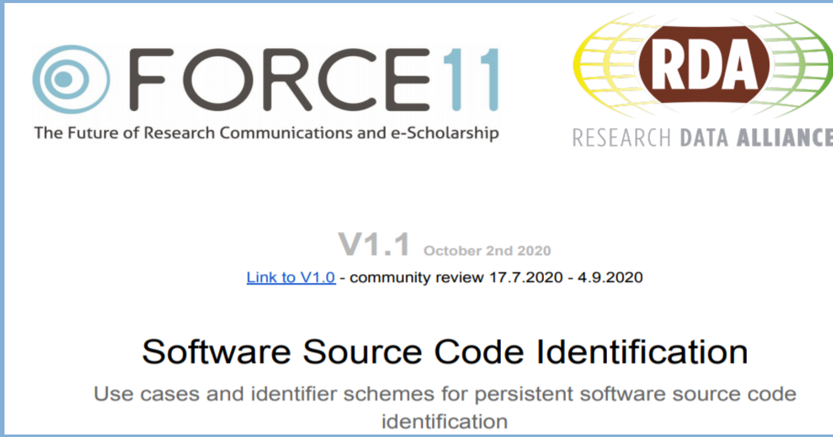
WIKIDATA
Wiki Item identifier (Qxxx)

al Resource Key

Handle
Handle System identifiers

swMATH
an information service for mathematical software

RRID


FORCE11
The Future of Research Communications and e-Scholarship

RDA
RESEARCH DATA ALLIANCE

V1.1 October 2nd 2020
Link to V1.0 - community review 17.7.2020 - 4.9.2020

Software Source Code Identification

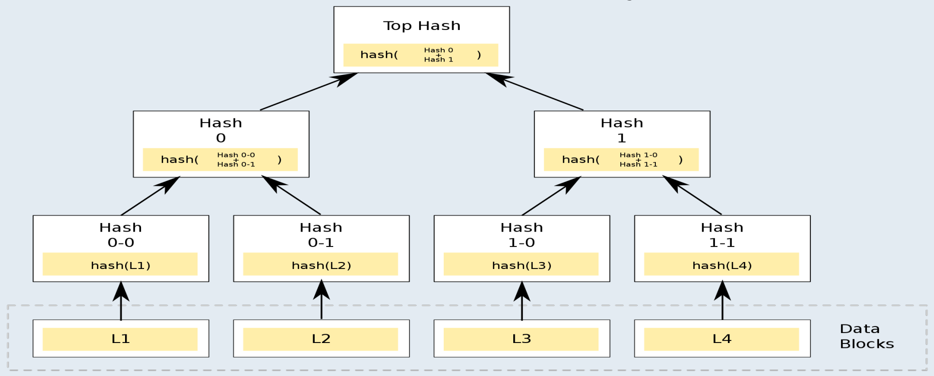
Use cases and identifier schemes for persistent software source code identification

Read more at https://doi.org/10.15497/RDA0

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

Intrinsic systems of identifiers for software

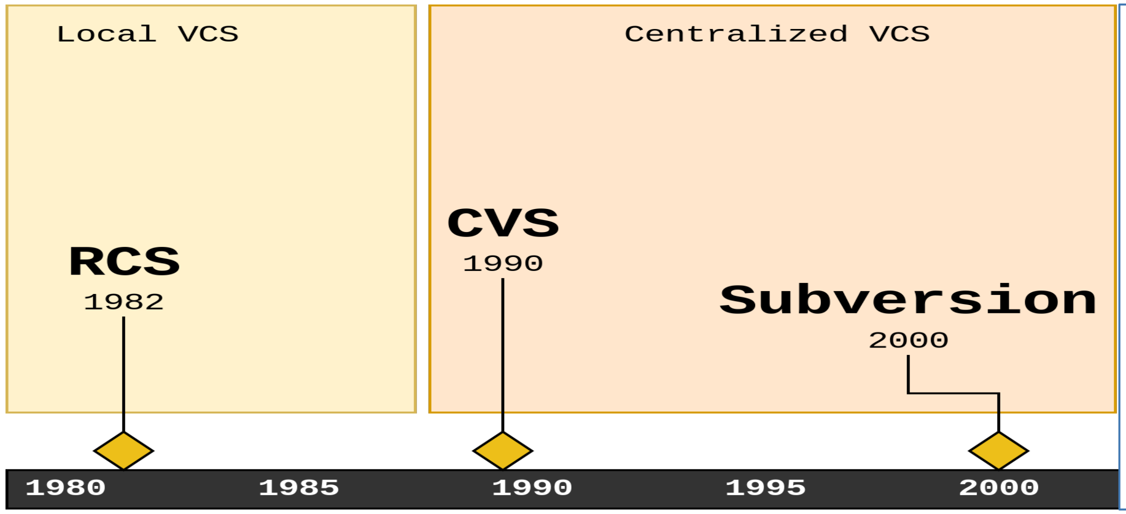
Ralph Merkle, 1987 « *A digital signature based on a conventional encryption function* »



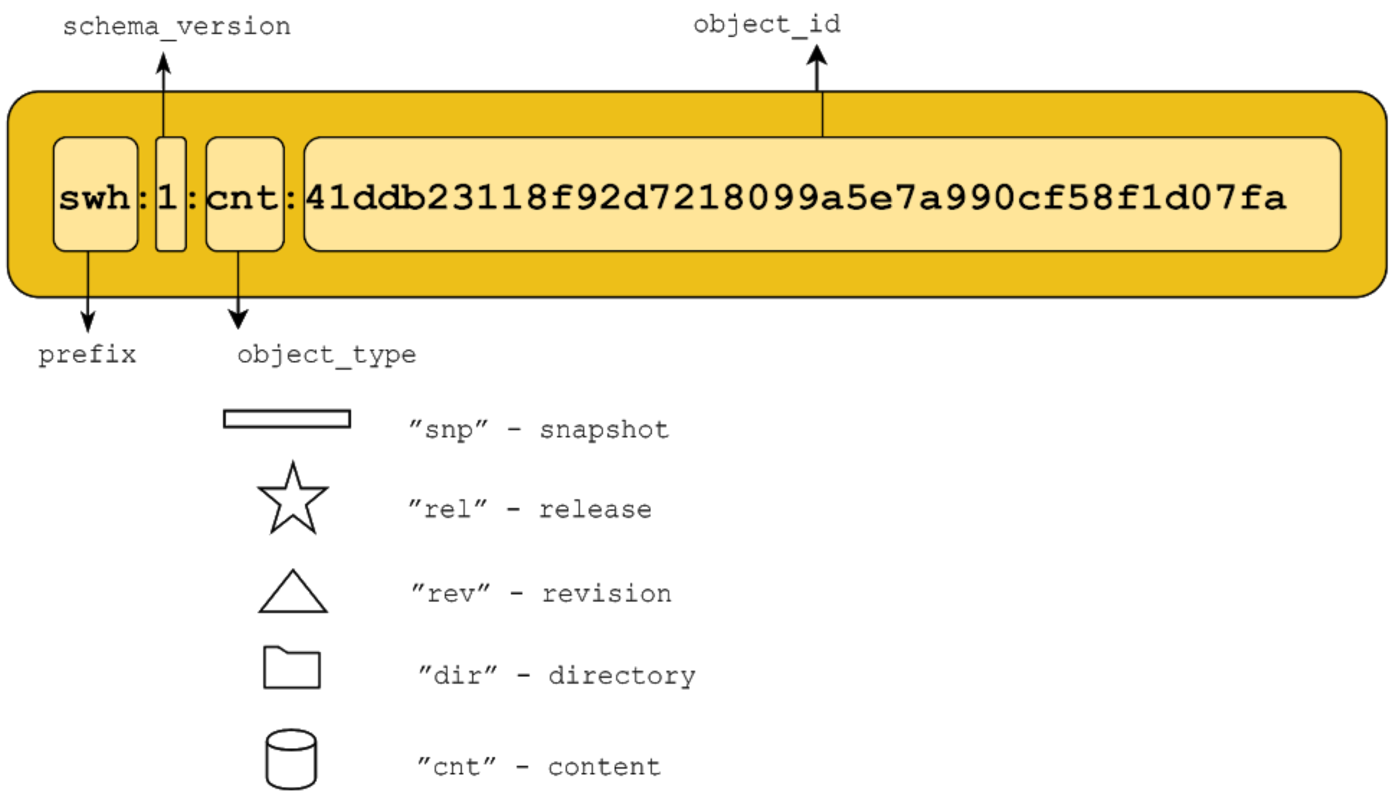
Blockchains

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 P

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

EUROPEAN O
SCIENCE CL

Quality, Curation, Metrics

- ***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”*

- ***Quality and curation***
  - *“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 AECs”*

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*

EUROPEAN OI
SCIENCE CLO

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

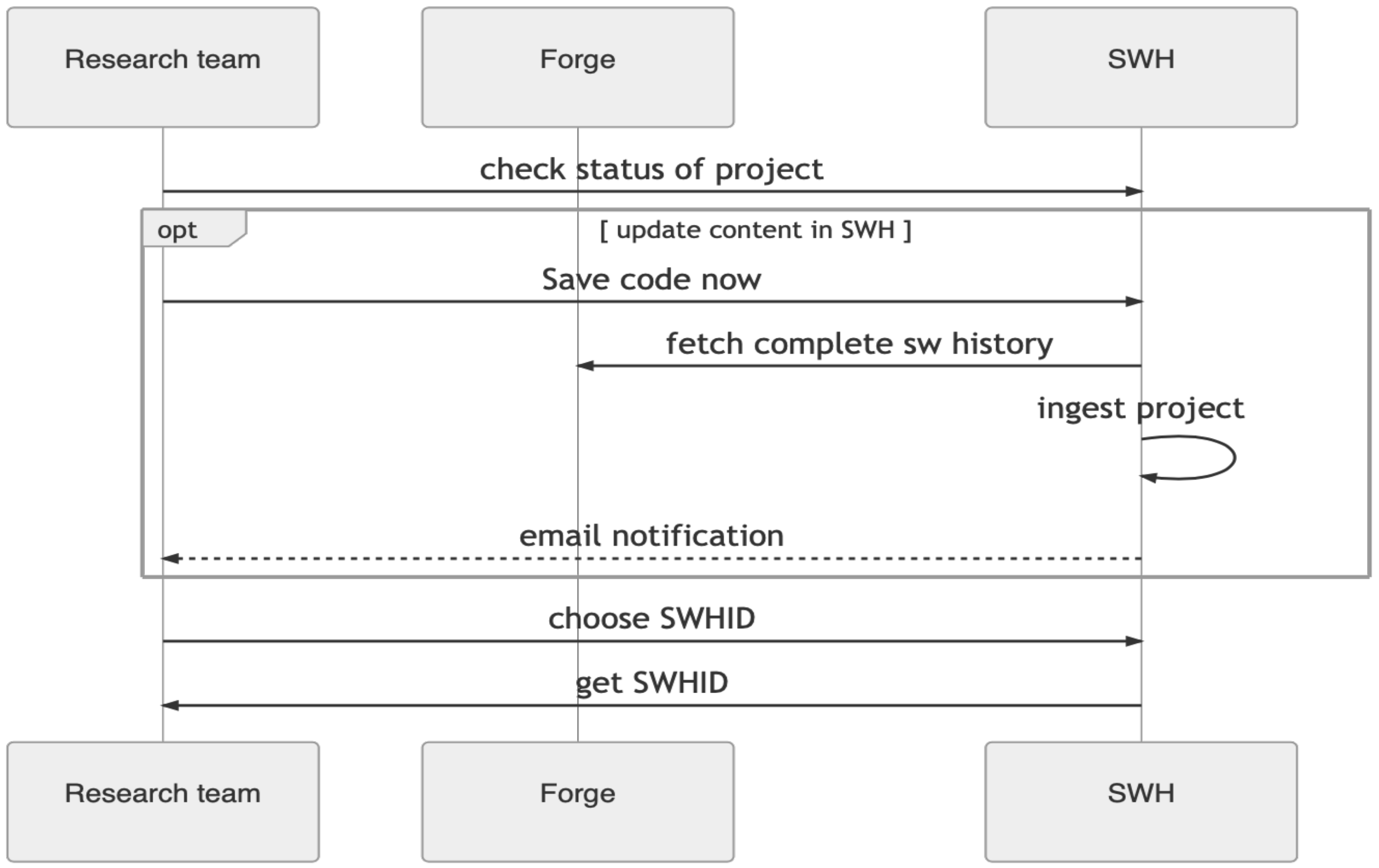
Thank you

roberto@dicosmo.org

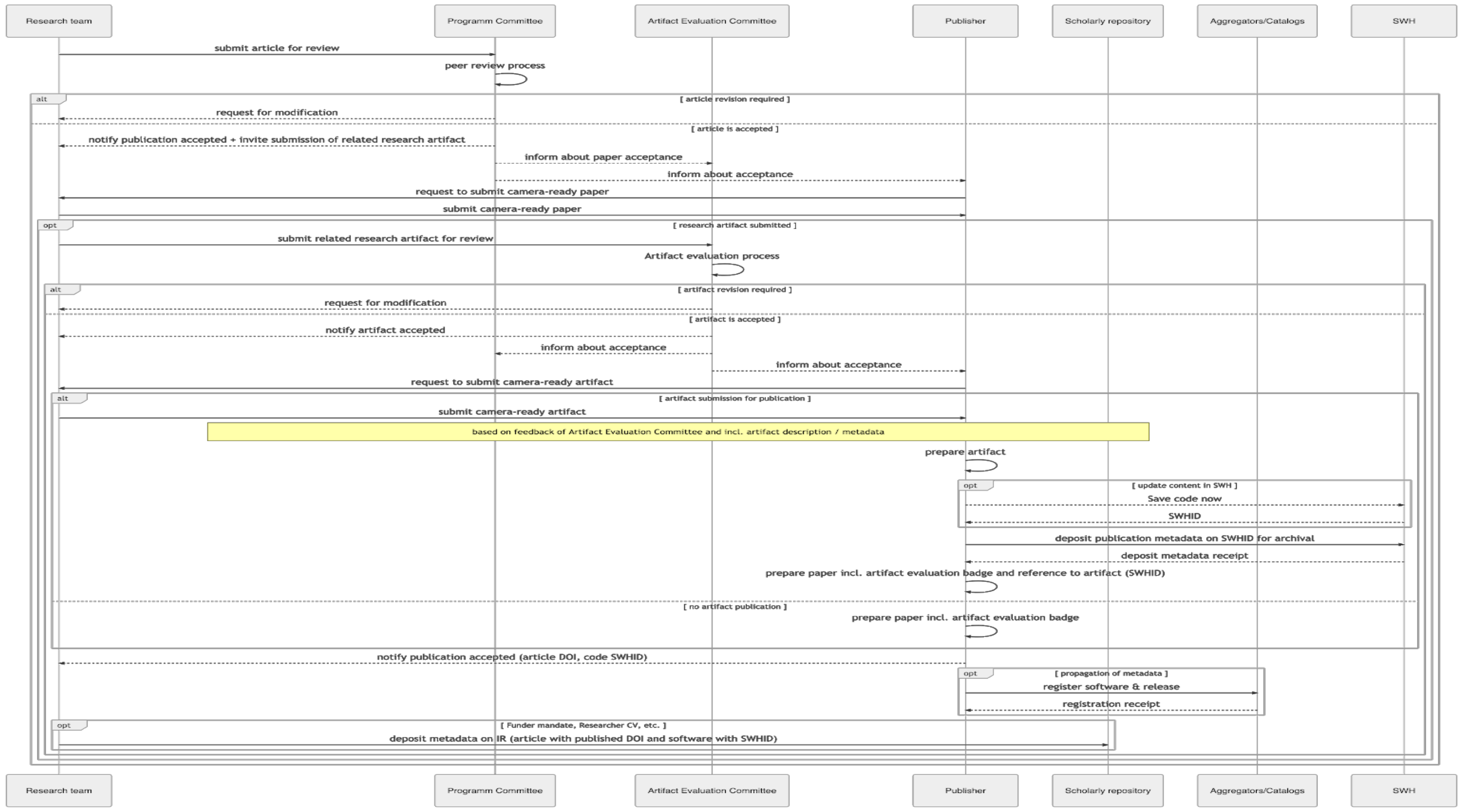
https://www.softwareheritage.org

EUROPEAN OPE
SCIENCE CLOU

Simple Workflow: Archive into Software Heritage



Complex Workflow: Conference with Artefact evaluation


Research team
Programm Committee
Artifact Evaluation Committee
Publisher
Scholarly repository
Aggregators/Catalogs
SWH
submit article for review
peer review process
alt
[article revision required]
request for modification
[article is accepted]
notify publication accepted + invite submission of related research artifact
inform about paper acceptance
inform about acceptance
request to submit camera-ready paper
submit camera-ready paper
opt
[research artifact submitted]
submit related research artifact for review
Artifact evaluation process
alt
[artifact revision required]
request for modification
[artifact is accepted]
notify artifact accepted
inform about acceptance
inform about acceptance
request to submit camera-ready artifact
alt
[artifact submission for publication]
submit camera-ready artifact
based on feedback of Artifact Evaluation Committee and incl. artifact description / metadata
prepare artifact
opt
[update content in SWH]
Save code now
SWHID
deposit publication metadata on SWHID for archival
deposit metadata receipt
prepare paper incl. artifact evaluation badge and reference to artifact (SWHID)
[no artifact publication]
prepare paper incl. artifact evaluation badge
notify publication accepted (article DOI, code SWHID)
opt
[propagation of metadata]
register software & release
registration receipt
opt
[Funder mandate, Researcher CV, etc.]
deposit metadata on IR (article with published DOI and software with SWHID)
Research team
Programm Committee
Artifact Evaluation Committee
Publisher
Scholarly repository
Aggregators/Catalogs
SWH