Software Source Code Interest Group

Metadata, identifiers and reproducibility

Roberto Di Cosmo (Software Heritage, INRIA)
Morane Gruenpeter (Software Heritage, CrossMiner)

roberto@dicosmo.org

Mars 22nd, 2018

Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

Software Heritage

Our mission

Collect, preserve and share the *source code* of *all the software* that is publicly available

Past, present and future

Preserving the past, *enhancing* the present, *preparing* the future

10th RDA plenary Software Source Code IG results

Identified

- interest in *Software Source Code*
- use cases
- ontology/vocabularies used
- properties needed for Software Source Code
- advantages for structured data

Interest in IG

Author point of view

- software accompany data
- software citation- get credit
- register and describe software
- promote software as a first class research product
- PID for software
- managing code : incorporate better practices for software

User point of view

- discover and recover software
- software citation- how to cite
- software discovery and research
- improve publication
- reuse
- preserving software source code

Identified use cases

Author point of view

- publish / deposit source code with metadata
- archive software
- expose metadata to indexes
- credit attribution and authorship
- conditions/restrictions for use
- link to people, data, funding

User point of view

- discovery (semantic search)
- lookup software source code
- reproducibility
- what compiler is required
- what test data are available
- build software
- integrate to workflow

Metadata terms

identify

- identifier
- title
- authors
- version
- type
- origin source

execute

- link to compiled version
- repository
- compiler
- environment
- examples

classify

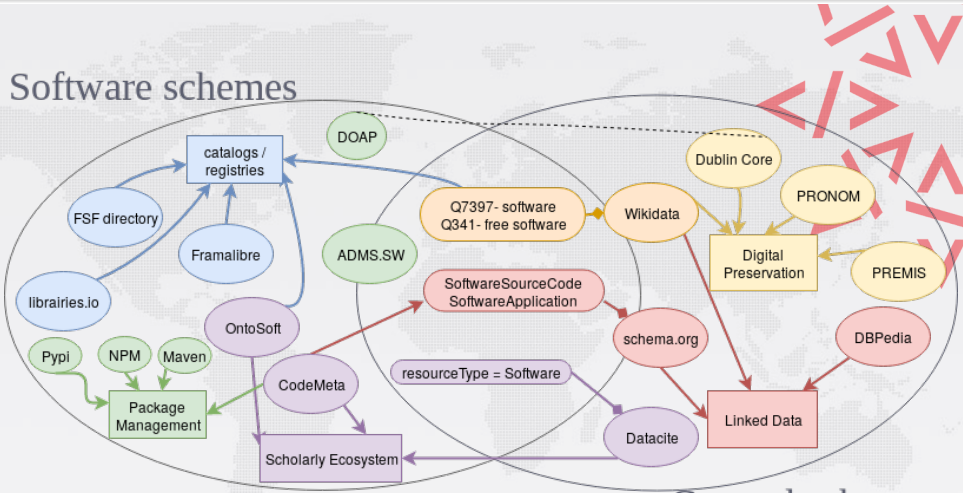
- description
- keywords
- in/out data
- references
- algorithms
- docs url
- status

administrative

- contact
- authorship
- funders
- license
- publisher
- dates

With what terms should we describe a *software artifact*?

Explore the metadata landscape



Back to basics: DIOs vs. IDOs

DIO (digital identifier of an object)

- digital identifiers for traditional (non digital) objects
  - epistemic complications and significant governance issues, …

Back to basics: DIOs vs. IDOs

DIO (digital identifier of an object)

- digital identifiers for traditional (non digital) objects
  - epistemic complications and significant governance issues, …

The software concept/project needs a DIO

Back to basics: DIOs vs. IDOs

DIO (digital identifier of an object)

- digital identifiers for traditional (non digital) objects
  - epistemic complications and significant governance issues, …

The software concept/project needs a DIO

IDO (identifier of a digital object)

- (digital) identifier for digital objects
  - simpler to build/handle and can be intrinsic

Back to basics: DIOs vs. IDOs

DIO (digital identifier of an object)

- digital identifiers for traditional (non digital) objects
  - epistemic complications and significant governance issues, ...

The software concept/project needs a DIO

IDO (identifier of a digital object)

- (digital) identifier for digital objects
  - simpler to build/handle and can be intrinsic

The software source code needs an IDO for each version or state

Back to basics: DIOs vs. IDOs

DIO (digital identifier of an object)

- digital identifiers for traditional (non digital) objects
  - epistemic complications and significant governance issues, …

The software concept/project needs a DIO

IDO (identifier of a digital object)

- (digital) identifier for digital objects
  - simpler to build/handle and can be intrinsic

The software source code needs an IDO for each version or state

Separation of concerns

- yes, we need both DIOs and IDOs
- no, we must not mistake DIOs for IDOs (and viceversa)

Our challenge in the PID arena

Long term

Identifiers must be there for the long term

No middle man

Identifiers must be meaningful even if resolvers go away

Integrity, not just naming

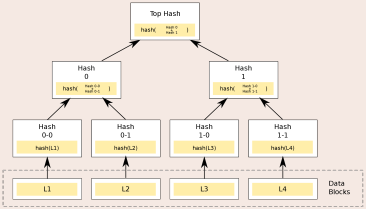
Identifier must ensure that the retrieved object is the intended one

Uniqueness by design

only one name for each object, each object has only one name

Intrinsic identifiers in Software Heritage

Merkle tree (R. C. Merkle, Crypto 1979)



Combination of

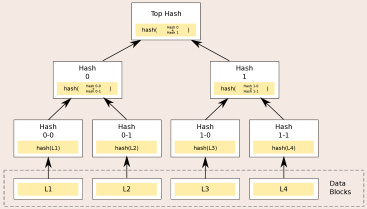
- tree
- hash function

Classical cryptographic construction

fast, parallel signature of large data structures, built-in deduplication

Intrinsic identifiers in Software Heritage

Merkle tree (R. C. Merkle, Crypto 1979)



Combination of

- tree
- hash function

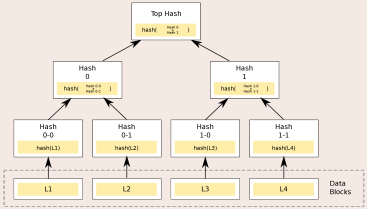
Classical cryptographic construction

fast, parallel signature of large data structures, built-in deduplication

- satisfies all three criteria
- widely used in industry (e.g., Git, nix, blockchains, IPFS, …)

Intrinsic identifiers in Software Heritage

Merkle tree (R. C. Merkle, Crypto 1979)



Combination of

- tree
- hash function

Classical cryptographic construction

fast, parallel signature of large data structures, built-in deduplication

- satisfies all three criteria
- widely used in industry (e.g., Git, nix, blockchains, IPFS, …)

Working together

Example: links to *software source code* in an article

Leveraging the Software Heritage universal archive:

set of files `swh:1:tree:06741c8c37c5a384083082b99f4c5ad94cd0cd1f`
id of tree object listing all the files in a project (at a given time)

revision `swh:1:rev:7598fb94d59178d65bd8d2892c19356290f5d4e3`
id of commit object which a tree and (a pointer to) the history

Working together

Example: links to *software source code* in an article

Leveraging the Software Heritage universal archive:

set of files `swh:1:tree:06741c8c37c5a384083082b99f4c5ad94cd0cd1f`
id of tree object listing all the files in a project (at a given time)

revision `swh:1:rev:7598fb94d59178d65bd8d2892c19356290f5d4e3`
id of commit object which a tree and (a pointer to) the history

metadata this *will* involve some form of DIO

- and we get all the complications back