Metadata, use cases and identifiers

Roberto Di Cosmo (SWH, INRIA)
Morane Gruenpeter (SWH, Crossminer)

roberto@dicosmo.org

Mars 22nd, 2018
Outline

1. Explore the metadata landscape
2. 10th RDA plenary Software Source Code IG results
3. Updates from, and links to Software Heritage
4. Conclusion
What is software?
The metadata challenge

What is software?

Software as a concept

- software project / entity
What is software?

Software as a concept

- software project / entity
- the creators and the community around it
What is software?

**Software as a concept**
- software project / entity
- the creators and the community around it

**Software artifact**
- the binaries for different environments
The metadata challenge

What is software?

Software as a concept
- software project / entity
- the creators and the community around it

Software artifact
- the binaries for different environments
- the software source code for each version
The metadata landscape

Software schemes

General schemes

- Dublin Core
- PRONOM
- Digital Preservation
- PREMIS
The metadata landscape

Software schemes

FSF directory
librairies.io
Pypi
NPM
Maven
Package Management

catalogs / registries
swMATH
ADMS.SW
DOAP

General schemes

Dublin Core
PRONOM
Digital Preservation
PREMIS
The metadata landscape

Software schemes

- catalogs / registries
  - FSF directory
  - swMATH
  - ADMS.SW
  - OntoSoft
  - CodeMeta
  - Scholarly Ecosystem

- DOAP

- Q7397 - software
  - Q341 - free software

- Wikidata
  - SoftwareSourceCode
  - SoftwareApplication
  - Digital Preservation

- Dublin Core
- PRONOM
- PREMIS
- DBPedia
- Linked Data
- schema.org

General schemes

Package Management
Outline

1. Explore the metadata landscape
2. 10th RDA plenary Software Source Code IG results
3. Updates from, and links to Software Heritage
4. Conclusion
Subjects discussed with questionnaire

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

Research topics

- software accompany data
- promote software as a first class research product
- software citation
- research software
- improve publication
Interest in IG

Research topics

- software accompany data
- promote software as a first class research product
- software citation
- research software
- improve publication

General topics

- PID for software
- managing code: incorporate better practices for software
- discover and recover software
- reuse
- preserving software source code
Identified use cases

Research use cases

- publish / deposit source code with metadata
- credit attribution and authorship
- reproducibility
- what test data are available
- research software source code
Identified use cases

Research use cases
- publish / deposit source code with metadata
- credit attribution and authorship
- reproducibility
- what test data are available
- research software source code

General use cases
- archive software
- expose metadata to indexes
  - link to people, data, funding
- discovery (semantic search)
- conditions/restrictions for use
- build software (what compiler is required)
- integrate into workflow
Metadata terms

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

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

**Execute**
- link to a compiled version
- repository
- compiler
- environment
- examples
Metadata terms

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

**Execute**
- link to a compiled version
- repository
- compiler
- environment
- examples

**Classify**
- description
- keywords
- in/out data
- references
- algorithms
- docs url
<table>
<thead>
<tr>
<th>Metadata terms</th>
<th>identify</th>
<th>execute</th>
<th>classify</th>
<th>administrate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>identifier</td>
<td>link to a compiled version</td>
<td>description</td>
<td>contact</td>
</tr>
<tr>
<td></td>
<td>title</td>
<td>repository</td>
<td>keywords</td>
<td>authorship</td>
</tr>
<tr>
<td></td>
<td>authors</td>
<td>compiler</td>
<td>in/out data</td>
<td>funders</td>
</tr>
<tr>
<td></td>
<td>version</td>
<td>environment</td>
<td>references</td>
<td>license</td>
</tr>
<tr>
<td></td>
<td>type</td>
<td>examples</td>
<td>algorithms</td>
<td>editor</td>
</tr>
<tr>
<td></td>
<td>origin</td>
<td></td>
<td>docs url</td>
<td>(publisher)</td>
</tr>
<tr>
<td></td>
<td>source</td>
<td></td>
<td></td>
<td>dates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>status</td>
</tr>
</tbody>
</table>
**Metadata terms**

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

- **execute**
  - link to a compiled version
  - repository
  - compiler
  - environment
  - examples

- **classify**
  - description
  - keywords
  - in/out data
  - references
  - algorithms
  - docs url

- **administrate**
  - contact
  - authorship
  - funders
  - license
  - editor
    - (publisher)
  - dates
  - status

**Group activity: review, comments and questions**
Group activity: review, comments and questions

Discussion questions

- Are the categories (identify, execute, classify and administrate) enough? should there be more?
- Are the terms correctly organized?
- What terms are missing?
- If we compare to the metadata terms identified in the SCIWG, which terms are needed for software citation?
Our mission

Collect, preserve and share the source code of all the software

Preserving the past, enhancing the present, preparing the future
Archive and observatory, serving the needs of society as a whole

Software Heritage

Source files
4,250,616,071

Commits
973,163,303

Projects
83,796,733

largest collection of software source code in the world
Our challenge in the PID arena

Our requirements

- **Long term** identifiers must be there for the long term
- **Free** one cannot *buy* billions of identifiers
- **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
### Our requirements

**Long term** identifiers must be there for the long term

**Free** one cannot *buy* billions of identifiers

**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

### We can find no satisfaction...

- Ark, PURLs, DOIs, Handle, … all miss a part of it
- we use cryptographic hashes instead (Merkle trees, circa 1979)
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)
Request for comment

Feedback on our PID schemas

see https://docs.softwareheritage.org/devel/swh-model/persistent-identifiers.html

Test the archive navigation in Software Heritage


Thoughts on the DIO / IDO conceptualization

contact Roberto and Morane
Questions?