Software Ontologies and Metadata Schemes

Morane Gruenpeter

Inria, Software Heritage

September 19, 2017
"Ontologies are agreements, made in a social context, to accomplish some objectives. It’s important to understand those objectives, and be guided by them."

“Ontologies are agreements, made in a social context, to accomplish some objectives. It’s important to understand those objectives, and be guided by them.”


Software Ontology

- What is software?
- With what terms should we describe a software artifact?
- What about software source code?
Metadata about Software Source Code

Software metadata objectives
manage, share, discover, archive software source code

Use cases
- semantic search: find software by author, version, keywords
- browse source code with context information
- cite and be cited

LOV- Linked open vocabularies
“Vocabularies provide the semantic glue enabling data to become meaningful data.”
Where is the metadata available?

<table>
<thead>
<tr>
<th>catalogs and registries</th>
<th>Publisher’s repositories</th>
</tr>
</thead>
<tbody>
<tr>
<td>libraries.io</td>
<td>GitHub</td>
</tr>
<tr>
<td>OpenHub</td>
<td>Bitbucket</td>
</tr>
<tr>
<td>OntoSoft</td>
<td>SourceForge</td>
</tr>
</tbody>
</table>

- **registries**
  - not created by author
  - added by authors/maintainers
  - very detailed
  - not a priority

- **repositories**
  - depends on registry
  - depends on publisher
Where is the metadata available?

**catalogs and registries**
- libraries.io
- OpenHub
- OntoSoft

**Publisher’s repositories**
- GitHub
- Bitbucket
- SourceForge

**advantages and drawbacks**

<table>
<thead>
<tr>
<th></th>
<th>registries</th>
<th>repositories</th>
</tr>
</thead>
<tbody>
<tr>
<td>accuracy</td>
<td>- not created by author</td>
<td>+ added by authors/maintainers</td>
</tr>
<tr>
<td>completeness</td>
<td>+ very detailed</td>
<td>- not a priority</td>
</tr>
<tr>
<td>longevity</td>
<td>- depends on registry</td>
<td>- depends on publisher</td>
</tr>
</tbody>
</table>
Where is the metadata available?

**in the *software source code* itself**

- package management file
- CITATION file
- .About file
- codemeta.json file

Advantages and drawbacks:

**metadata file**

- accuracy: + created by author and evolves with code
- completeness: + freedom of vocabulary and terms used
- longevity: + not dependent on platform (repository or registry)

Botatomline: to insure the archival of metadata, keep it in the data
Where is the metadata available?

In the *software source code* itself:

- package management file
- CITATION file
- .About file
- codemeta.json file

Advantages and drawbacks:

<table>
<thead>
<tr>
<th>Metadata File</th>
<th>Advantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>accuracy</td>
<td>+ created by author and evolves with code</td>
</tr>
<tr>
<td>completeness</td>
<td>+ freedom of vocabulary and terms used</td>
</tr>
<tr>
<td>longevity</td>
<td>+ not dependent on platform (repository or registry)</td>
</tr>
</tbody>
</table>
Where is the metadata available?

**in the *software source code* itself**
- package management file
- CITATION file
- .About file
- codemeta.json file

**advantages and drawbacks**

<table>
<thead>
<tr>
<th>metadata file</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>accuracy</td>
<td>+ created by author and evolves with code</td>
</tr>
<tr>
<td>completeness</td>
<td>+ freedom of vocabulary and terms used</td>
</tr>
<tr>
<td>longevity</td>
<td>+ not dependent on platform (repository or registry)</td>
</tr>
</tbody>
</table>

**Bottomline:** to insure the archival of metadata, keep it **in** the data.
This is not *software source code*
Software Citation Principles (FORCE11’s 2015 conference and WG)

- **Importance**: first class citizen in the scholarly ecosystem
- **Credit and attribution**: authors, maintainer
- **Unique identification**: points to a unique, specific software version (DOI, Git SHA1 hash, etc.)
- **Persistence**: identification beyond the lifespan of the software (swh-id)
- **Accessibility**: url, publisher
- **Specificity**: version, environment
Landscape of Software Ontologies

- Dedicated for Software
- Package Management
- Scholarly Ecosystem
- Linked Data
- Digital Preservation
- General schemes
Landscape of Software Ontologies

- DOAP
- ADMS.SW
- SEON
- NPM
- Pypi
- Maven
- Package Management
- Dedicated for Software
- Ontosoft
- Scholarly Ecosystem
- Datacite
- SoftwareSourceCode
- SoftwareApplication
- Linked Data
- schema.org
- Q7397
- Digital Preservation
- Wikidata
- General schemes
- Dublin Core
- MARC
- PRONOM
Landscape of Software Ontologies

- DOAP
- ADMS.SW
- SEON
- NPM
- Pypi
- Maven
- Package Management
- Scholarly Ecosystem
- Dedicated for Software
- OntoSoft
- CodeMeta
- SoftwareSourceCode SoftwareApplication
- Linked Data
- schema.org
- Digital Preservation
- Wikidata
- Q7397
- General schemes
- Dublin Core
- MARC
- PRONOM
The CodeMeta Initiative

A Rosetta Stone for Metadata in Scientific Software

CodeMeta aims to create a framework \{schema, crosswalk, guidelines\} that can be used to \textit{standardize the exchange of software metadata}

Advantages

- the crosswalk table
- built on schema.org \textit{SoftwareSourceCode}
- an active community
Discussion

**CodeMeta - where are the gaps?**
- missing properties
- missing ontologies
- semantic misconceptions

**Software Source Code** metadata recommendations
- use cases
- best practices / guidelines
Reminder

RDA page
https://www.rd-alliance.org/ig-software-source-code-rda-10th-plenary-meeting

Working document used during the session
http://bit.ly/2wggInQ