Towards FAIR Research Software

Daniel Garijo, Ontology Engineering Group, Universidad Politécnica de Madrid, Spain

Assessing Best Research Software Practices through Metadata

✉️ daniel.garijo@upm.es
🐦 @dgarijov
Research Software is one of the pillars of Open Science.
Research Software is one of the pillars of Open Science.
The rise of the FAIR Data principles

Other guidelines:

- Guidelines for Transparency and Openness Promotion (TOP) [2]
- Reproducibility Enhancement Principles (REP) [3]
- ...

FAIR for research software

F: Software, and its associated metadata, is easy for both humans and machines to find.

F1. Software is assigned a globally unique and persistent identifier.
   - F1.1. Components of the software representing levels of granularity are assigned distinct identifiers.
   - F1.2. Different versions of the software are assigned distinct identifiers.
F2. Software is described with rich metadata.
F3. Metadata clearly and explicitly include the identifier of the software they describe.
F4. Metadatas are FAIR, searchable and indexable.

A: Software, and its metadata, is retrievable via standardized protocols.

A1. Software is retrievable by its identifier using a standardized communications protocol.
   - A1.1. The protocol is open, free, and universally implementable.
   - A1.2. The protocol allows for an authentication and authorization procedure, where necessary.
A2. Metadata are accessible, even when the software is no longer available.

I: Software interoperates with other software by exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards.

I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards.
I2. Software includes qualified references to other objects.

R: Software is both usable (can be executed) and reusable (can be understood, modified, built upon, or incorporated into other software).

R1. Software is described with a plurality of accurate and relevant attributes.
   - R1.1. Software is given a clear and accessible license.
   - R1.2. Software is associated with detailed provenance.
R2. Software includes qualified references to other software.

**FAIR is highly related to metadata**

<table>
<thead>
<tr>
<th>F</th>
<th>Create a description of your software</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Register your software in a software registry</td>
</tr>
<tr>
<td>3</td>
<td>Use a Unique and Persistent Identifier for your software</td>
</tr>
<tr>
<td>A</td>
<td>Make sure that people can download your software</td>
</tr>
<tr>
<td>5</td>
<td>Explain the functionality of your software</td>
</tr>
<tr>
<td>6</td>
<td>Use standard (community agreed) formats for inputs and outputs</td>
</tr>
<tr>
<td>R</td>
<td>Document your software</td>
</tr>
<tr>
<td>8</td>
<td>Give your software a licence</td>
</tr>
<tr>
<td>9</td>
<td>State how to cite your software</td>
</tr>
<tr>
<td>10</td>
<td>Follow best practices for software development</td>
</tr>
</tbody>
</table>

**FAIR4RS (RDA) [1]**

<table>
<thead>
<tr>
<th>F: Software, and its associated metadata, is easy for both humans and machines to find.</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1. Software is assigned a globally unique and persistent identifier.</td>
</tr>
<tr>
<td>- F1.1. Components of the software representing levels of granularity are assigned distinct identifiers.</td>
</tr>
<tr>
<td>- F1.2. Different versions of the software are assigned distinct identifiers.</td>
</tr>
<tr>
<td>F2. Software is described with rich metadata.</td>
</tr>
<tr>
<td>F3. Metadata clearly and explicitly include the identifier of the software they describe.</td>
</tr>
<tr>
<td>F4. Metadata are FAIR, searchable and indexable.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A: Software, and its metadata, is retrievable via standardized protocols.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1. Software is retrievable by its identifier using a standardized communications protocol.</td>
</tr>
<tr>
<td>- A1.1. The protocol is open, free, and universally implementable.</td>
</tr>
<tr>
<td>- A1.2. The protocol allows for an authentication and authorization procedure, where necessary.</td>
</tr>
<tr>
<td>A2. Metadata are accessible, even when the software is no longer available.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I: Software interoperates with other software by exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards.</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards.</td>
</tr>
<tr>
<td>I2. Software includes qualified references to other objects.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R: Software is both usable (can be executed) and reusable (can be understood, modified, built upon, or incorporated into other software).</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1. Software is described with a plurality of accurate and relevant attributes.</td>
</tr>
<tr>
<td>- R1.1. Software is given a clear and accessible license.</td>
</tr>
<tr>
<td>- R1.2. Software is associated with detailed provenance.</td>
</tr>
<tr>
<td>R2. Software includes qualified references to other software.</td>
</tr>
</tbody>
</table>

---

“The goal of CodeMeta is to create a concept vocabulary that can be used to standardize the exchange of software metadata across repositories and organizations” - [https://github.com/codemeta/codemeta](https://github.com/codemeta/codemeta)

Website: [https://codemeta.github.io/](https://codemeta.github.io/)

The CodeMeta Project

JSON-LD representation

Needs to be filled by hand
Challenge: Harvesting Research Software metadata

Research Software metadata is not abundant and machine readable.

I already did! Did you read the project readme?

Did you see the online documentation?

Perhaps the you saw the paper?

Can you please describe your software component with metadata?

Many domain-specific registries are curated by hand by experts.
Towards FAIR Research Software FAIR ZB_MED Colloquium. June, 2023

Extracting knowledge from software projects

- **Documentation**
  - Text classification
  - Named entity recognition and relation extraction

- **Code**
  - Static code analysis
Text classification: Software Metadata Extraction Framework

https://github.com/KnowledgeCaptureAndDiscovery/somef/

- **Readme Analysis**
  - Supervised classification
  - Regular expressions
  - Header analysis

- **File exploration**
  - Notebooks
  - Dockerfiles
  - Documentation

- **GitHub API**

SOMEF: File Exploration and Regular Expressions

- Extraction based on frequent header analysis
  - Fuzzy matching based on synsets

**Installation**

Installation through Docker

```
docker pull uscisi12/kgtk
```

To run KGTK in the command line:

```
docker run -it --rm --user root -e NB_GID=100 -e GEN_CERT=yes -e GRANT_SUDO=yes uscisi12/kgtk:latest
```

**Wordnet**

**KGTK: Knowledge Graph Toolkit**

Regular expressions, based on common practices (e.g., DOI, .bib, etc.)

The Knowledge Graph Toolkit (KGTK) is a comprehensive framework for the creation and exploitation of large hyper-relational knowledge graphs (KGs), designed for ease of use, scalability, and speed. KGTK represents KGs in tab-separated (TSV) files with four columns: edge-identifier, head, edge-label, and tail. All KGTK commands consume and produce KGs represented in this simple format, so they can be composed into pipelines to perform complex transformations on KGs. KGTK provides:
- Paragraph-based text classification
- Four main categories:
  - Installation, citation, description, invocation.
- Binary classification problem

<table>
<thead>
<tr>
<th>Truth Value</th>
<th>Category</th>
<th>Approx. Ratio</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Description</td>
<td>0.5</td>
<td>275</td>
</tr>
<tr>
<td>False</td>
<td>Installation</td>
<td>0.125</td>
<td>68</td>
</tr>
<tr>
<td>False</td>
<td>Invocation</td>
<td>0.125</td>
<td>68</td>
</tr>
<tr>
<td>False</td>
<td>Citation</td>
<td>0.125</td>
<td>68</td>
</tr>
<tr>
<td>False</td>
<td>Treebank</td>
<td>0.125</td>
<td>68</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>1.0</td>
<td>547</td>
</tr>
</tbody>
</table>
SOMEF: Recognizing Metadata Categories

- Name (GA)
- Full title (RE)
- Description (SC, HA)
- Citation (SC, RE, HA)
- Installation instructions (SC, HA)
- Invocation (SC)
- Usage examples (HA)
- Documentation (HA, FE)
- Requirements (HA)
- Contributors (HA)
- FAQ (HA)
- Support (HA)
- License (GA, HA, FE)
- Stars (GA)

Method used (provenance):
- Supervised Classification (SC)
- Header Analysis and Synset comparison (HA)
- File Exploration (FE)
- Regular Expressions (RE)
- GitHub API (GA)

- Contact (HA)
- Download URL (HA, GA)
- DOI (RE)
- DockerFile (FE)
- Notebooks (FE)
- Executable notebooks (Binder, Collab) (RE)
- Owner: (GA)
- Keywords (GA)
- Source code (GA)
- Releases (GA)
- Changelog (GA)
- Acknowledgements (HA)
- Logos (RE)
- Images (RE)
- Shell scripts (FE)
- Code of conduct (FE)
- Repository status (RE)
- Arxiv links (RE)
- Support channels (RE)
- Software category (SC) (Work in progress)
- …
Static code analysis in Python
- Extraction of available classes and functions
  - Documentation
- Requirements (reusing existing libraries)
- Call list
- File hierarchy
- Control flow (reusing existing libraries)
- Software invocation
  - Service? Package? Library? & invocation command
- Metadata export in JSON

Benefits
- Understanding, reuse, ML featurization, similarity, best practices

https://github.com/SoftwareUnderstanding/inspect4py

So what?

What can we do now?
- Assist
  - Ease descriptions
  - Ease reuse
  - Augment impact
- Assess (measure practices)
So what?

What can we do now?
- **Assist**
  - Ease descriptions
  - Ease reuse
  - Augment impact
- Assess (measure practices)

Towards FAIR Research Software FAIR ZB_MED Colloquium. June, 2023
Easing descriptions: SOMEF-Vider

https://github.com/SoftwareUnderstanding/SOMEF-Vider


Get Codemeta files automatically!

- JSON
- CodeMeta
- Turtle
Easing reuse and explore: Automated software catalogs

Useful for
- Comparison
- Exploring
- Reuse

Alpha available at: https://software.oeg.fi.upm.es/ Github: https://github.com/oeg-upm/soca

Work by Daniel Rodriguez
Augmenting impact: Software-Article Linker Toolkit

 Wikidata **bot** to link code repositories with Wikidata articles

Does the repository have a link to a paper?
Does the paper exist in Wikidata?
**Connect them!** Create a software entry if it does not exist

Jorge Bolinches (work in progress)
Many articles are not in WD. Can we include them?
Towards FAIR Research Software FAIR ZB_MED Colloquium. June, 2023

So what?

What can we do now?
- Assist
  - Ease descriptions
  - Ease reuse
  - Augment impact
- Assess (measure practices)
Which best practices are followed in an organization?

Tracking number of best practices across time

Work by Miguel Arroyo
Assessing best practices

Number of Repositories vs Number of Good Practices

Number of Repositories per best practice

Total Number
- 281

Repositories with Readme
- 217

Number with Citation
- 21

Number with Recent release
- 17

Number With Licenses
- 155

Zoom in Number of Repositories

Towards FAIR Research Software FAIR ZB_MED Colloquium. June, 2023
Assessing best practices (2)

Number of Repositories per Readme Best Practice

- Number with Readme: 217
- Number with Installation: 58
- Number with Requirement: 19
- Number with Documentation: 36

requirement section?
Readthepdocs?
External doc? Wiki?

Additional insight on licenses used

- Apache
- MIT
- Other
- GPL
Zenodo DOI? Paper DOI? Software heritage?

This is work in progress! We are including Containerization, package managers, individual repository assessment, etc.
Current life cycle requires researchers to:
- Create separate metadata files
- Curate them and maintain them
- Re-introduce metadata manually in different registries

Current aim:
- Improve and maintain high quality readme files
- Let the extraction tools do the work
- Maximize benefits from metadata extraction
  - Move towards assisting researchers produce FAIR software
Research software is a **critical asset for Open Science**

Software metadata **is key**:  
- Reusability, Comparison, Search  
- FAIR research software  
- Propagate impact of research

From **ASSESSing**, we are working towards **ASSISTing** using metadata  
- Highlight the **added benefit** to researchers and developers
Thanks to Yolanda Gil, Varun Ratnakar, Maximiliano Osorio, Hernán Vargas, Deborah Khider, Allen Mao, Aidan Kelley, Haripriya Dharmala, Jiajing Wang, Rosa Filgueira, Pablo Calleja, Oscar Corcho, Laura Camacho, Jhon Toledo, Miguel Angel García, Esteban Gonzalez & all the students at UPM and USC who participated in the initiatives mentioned in this presentation

This work has been supported by the Madrid Government (Comunidad de Madrid-Spain) under the Multiannual Agreement with Universidad Politécnica de Madrid in the line Support for R&D projects for Beatriz Galindo researchers, in the context of the V PRICIT (Regional Programme of Research and Technological Innovation)
Let's create **machine-actionable** software metadata