Accelerating Research Software Understandability Through Knowledge Capture

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

✉️ 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
Is this situation familiar?

Don't worry, you don't have to start your code from scratch.

You can re-use the software that the previous person on the project wrote several years ago.

Are there instructions for how to use it?

I doubt it.

Is the code commented?

Not likely.

Where are the files?

Who knows.

This is going to be painful, isn't it?

Just a scratch.

www.phdcomics.com
DON'T WORRY, YOU DON'T HAVE TO START YOUR CODE FROM SCRATCH.

YOU CAN RE-USE THE SOFTWARE THAT THE PREVIOUS PERSON ON THE PROJECT WROTE SEVERAL YEARS AGO.

ARE THERE INSTRUCTIONS FOR HOW TO USE IT?

DOUBT IT.

IS THE CODE COMMENTED?

NOT LIKELY.

WHERE ARE THE FILES?

WHO KNOWS.

THIS IS GOING TO BE PAINFUL, ISN'T IT?

JUST A SCRATCH.

Readme file, paper(s)

Months

GitHub, Gitlab, Docker image? etc.
In [1] we tried to reproduce an effort from one year before.

- All data were available online
- All tools were available online (except one, but authors had a replacement)

In [1] we tried to reproduce an effort from one year before.
- All data were available online
- All tools were available online (except one, but authors had a replacement)
- > 250 hrs to full reproducibility
- > 100 hrs to get familiar with the tools and their I/O

Millions of open-source repositories are updated/created every year.
Can we automatically accelerate research software understanding?
The dimensions of Research Software Understanding

Describe

Given a software project:
- What is it about?
- Examples?
- Relation to other resources (data, papers)?
- Metadata?

Compare

Given two or more tools:
- What are their similarities?
- Differences?
- Main features?

Reuse

How to quickly:
- run?
- repeat?
- reproduce?
- fix?
- combine?
User perspectives are crucial!

- How to...
  a. use a software component
  b. transform my data to use a software component?
  c. interpret the results?
  d. invoke the software component?
  e. configure the right parameters?
  f. compare against similar methods?

- How to...
  a. Ease capturing the dependencies and installation instructions of my software?
  b. Encapsulate my software so it can be used with other data?
  c. Describe my software so it can be used by others?
  d. Test if my software is ready to be used by others?
Three challenges for Research Software Understanding

1. Representing Research Software metadata

2. Knowledge capture from documentation and code

3. Automated encapsulation for reusability
Three challenges for Research Software Understanding

1. **Representing** Research Software metadata

2. **Knowledge capture** from documentation and code

3. **Automated encapsulation** for reusability
Research problem: Wide Research Software landscape

Versions and basic metadata? (creator, maintainer, license, citation…)

Contributions? Development process?

Dependencies and requirements?

Execution command and configuration

Supporting materials? Input data?

How to use WIDOCO

JAR execution

Download the latest .jar. WIDOCO available release (it will be something like widoco-VERSION.jar). Then just double click the .jar file.

You may also execute WIDOCO through the command line. Usage:

```
java -jar widoco-VERSION-jar-with-dependencies.jar [OPTIONS]
```
Representing Research Software Metadata: Scientists’ perspective

Crowdsourced Research Software Metadata Registry

- Complements code repositories to make them understandable
- Software metadata designed for scientists
- Metadata is curated by decentralized communities of users
- Training scientists on best practices

http://ontosoft.org

Comparing Tools

Compare Software

DrEICH algorithm, PIHM, PIHMgis, TauDEM, WBMsed

<table>
<thead>
<tr>
<th>PIHM</th>
<th>PIHMgis</th>
<th>DrEICH</th>
<th>TauDEM</th>
<th>WBMsed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geomorphology, Hydrological, Bedrock channel ero-</td>
<td>Basins, Continental</td>
<td>Basins, GIS</td>
<td>Hydrologically correctedDEM, Watershed</td>
<td>Sediment flux, Global model, Hydrological model</td>
</tr>
</tbody>
</table>

What are domain-specific keywords for this software? (e.g., hydrology, climate)

What Operating Systems can the software run on?

Unix/Linux

Unix/Windows/Linux/Mac OS

Unix/Windows/Linux/Mac OS

Unix/Linux

Is there any test data available for the software?

Test Data Location:
- http://sourceforge.net/projects/pihmmodel/

Test Data Description:
- Two test DEMs are included in the repository.
- Upper Juniata River 875 km²: see: http://source-
- The Logan River DEM is a small test dataset useful

Test Data Location:
- http://csdms.colorado.edu/wiki/Model: Tau- DEM#Testing

Test Data Description:
- Extensive input dataset is available on the CSDMS
Describing inputs, outputs and their structure

- Weather
- DEM
- Infiltration
- Penn State Hydrology Model (PIHM)
- Outflow
- Error

Adapt new sources
Run configured tool / model
Visualize result

Describing low granular metadata

FLDAS (climate)
Remote sensing
Low grained machine-readable Software Metadata:

- (From OntoSoft) Attribution, license, funding, usage examples...
- **Executable** software components
- **Software invocation**
- Input & output files, variables and units
- Containers used to **encapsulate** and run software component

**Input/output variables**

<table>
<thead>
<tr>
<th>Label</th>
<th>Long Name</th>
<th>Description</th>
<th>Standard Name</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bed</td>
<td>Bed Depth</td>
<td>Bed Depth</td>
<td>channel_bed__thickness</td>
<td>m</td>
</tr>
<tr>
<td>KsatV</td>
<td>Bed Hydraulic Conductivity</td>
<td>Bed Hydraulic Conductivity</td>
<td>soil_water__vertical_saturated_hydraulic_conductivity</td>
<td>m day⁻¹</td>
</tr>
<tr>
<td>Water table value</td>
<td>Water table of the IC</td>
<td>Water table of the IC</td>
<td></td>
<td>m</td>
</tr>
</tbody>
</table>
Extending:

- **Schema.org/Codemeta** (software metadata)
- **W3C Data Cubes** (Contents of inputs and outputs)
- **NASA QUDT** (Units)
- **DockerPedia** (Software images)
- **Scientific Variables Ontology** (Standard Variables)

Software images are created from configuration files (e.g., Dockerfiles)
- Execution of a tool
- Configuration
- Dependencies
- Infrastructure
Helpful for reproducibility

Initial effort transforming part of DockerHub: [https://dockerpedia.inf.utfsm.cl/](https://dockerpedia.inf.utfsm.cl/)

Notebooks contain crucial examples to understand scientific projects:
- Demos
- Tutorials
- Configuration

Basic metadata (e.g., title)
Configuration / queries needed
Intermediate results
Three challenges for Research Software Understanding

1. Representing Research Software metadata

2. Knowledge capture from documentation and code

3. Automated encapsulation for reusability
Research problem: Harvesting Research Software metadata

Research Software metadata is not abundant or machine readable

- Can you please describe your software component with metadata?
- I already did! Did you read the project readme?
- Did you see the online documentation?
- Perhaps you saw the paper?

Many domain-specific registries are curated by hand by experts
### Extracting knowledge from software projects

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

- **Code**
  - Static code analysis

#### File Details:

<table>
<thead>
<tr>
<th>Directory</th>
<th>Content</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>docs</td>
<td>update doc</td>
<td>13 days ago</td>
</tr>
<tr>
<td>experiments</td>
<td>Added pipeline missed in previous version of create_models</td>
<td>8 months ago</td>
</tr>
<tr>
<td>notebook</td>
<td>Fix #180</td>
<td>15 months ago</td>
</tr>
<tr>
<td>src/somemf</td>
<td>update version</td>
<td>13 days ago</td>
</tr>
<tr>
<td>.gitignore</td>
<td>Fix test and added env to .gitignore</td>
<td>29 days ago</td>
</tr>
<tr>
<td>.readthedocs.yml</td>
<td>documentation</td>
<td>2 years ago</td>
</tr>
<tr>
<td>CITATION.cff</td>
<td>Add citation file</td>
<td>4 months ago</td>
</tr>
<tr>
<td>Dockerfile</td>
<td>updating Docker image</td>
<td>4 months ago</td>
</tr>
<tr>
<td>LICENSE</td>
<td>initial cleanup</td>
<td>2 years ago</td>
</tr>
<tr>
<td>README.md</td>
<td>update doc</td>
<td>13 days ago</td>
</tr>
<tr>
<td>config.json</td>
<td>Created script to generate models and updated python version to 3.9</td>
<td>8 months ago</td>
</tr>
<tr>
<td>mkdocs.yml</td>
<td>Fix #178</td>
<td>15 months ago</td>
</tr>
<tr>
<td>pyproject.toml</td>
<td>minor package changes</td>
<td>4 months ago</td>
</tr>
<tr>
<td>setup.py</td>
<td>Fix #437</td>
<td>28 days ago</td>
</tr>
</tbody>
</table>
Accelerating Research Software Understandability Through Knowledge Capture. June, 2022

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

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

- File exploration
  - Notebooks
  - Dockerfiles
  - Documentation

- GitHub API

**SOMEF: Supervised classification**

- **Paragraph-based text classification**
- **Four main categories (binary classification):**
  - Installation
  - Citation
  - Description
  - Invocation

<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><strong>Total</strong></td>
<td></td>
<td><strong>1.0</strong></td>
<td><strong>547</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Best pipeline</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>CountVectorizer + LogisticRegression</td>
<td>0.85</td>
<td>0.79</td>
<td>0.82</td>
</tr>
<tr>
<td>Installation</td>
<td>TFIDFVectorizer + StochasticGradientDescent</td>
<td>0.92</td>
<td>0.9</td>
<td>0.91</td>
</tr>
<tr>
<td>Invocation</td>
<td>CountVectorizer + NaiveBayes</td>
<td>0.88</td>
<td>0.9</td>
<td>0.89</td>
</tr>
<tr>
<td>Citation</td>
<td>CountVectorizer + NaiveBayes</td>
<td>0.89</td>
<td>0.98</td>
<td>0.93</td>
</tr>
</tbody>
</table>

**Simple** classification pipelines yield nice results
SOMEF: File Exploration and Regular Expressions

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

**Installation**

Installation through Docker

```
docker pull uscis112/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 uscis112/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:
Using READMEs to categorize software

- Preprocessing is crucial
- Creating a methodology to recognize categories based on awesome lists
  - Text classification
  - Bi-LSTM networks

“Geosciences”
Method used (provenance):

- Supervised Classification (SC)
- Header Analysis and Synset comparison (HA)
- File Exploration (FE)
- Regular Expressions (RE)
- GitHub API (GA)
Residual Dense Network for Image Super-Resolution

The code is built on EDSR (Torch) and tested on Ubuntu 14.04 environment (Torch7, CUDA8.0, cuDNN5.1) with Titan X/1080Ti/Xp GPUs.

Extracting additional context from README files
- Transformer-based architectures (SciBERT)
- Inference for explicit and implicit relationships

https://github.com/oeg-upm/ner4soft/ (Work in progress)
Static code analysis in Python
- Extraction of available classes and functions
  - Documentation
- Requirements (reusing existing libraries)
- Call list
- Control flow (reusing existing libraries)
- Software invocation
  - Service
  - Package
  - Library
  - Script
    - Invocation command
- Output as a JSON file

GitHub: https://github.com/SoftwareUnderstanding/inspect4py

Early result: Automated software catalogs

Alpha available at: https://software.oeg.fi.upm.es/ Github: https://github.com/oeg-upm/soca
Extracting KGs from thousands of Open Source repositories
- Zenodo software (> 12000)
- Measuring best practices based on metadata
Three challenges for Research Software Understanding

1. **Representing** Research Software metadata

2. **Knowledge capture** from documentation and code

3. **Automated encapsulation** for reusability
Capturing different configurations of a complex tool requires significant knowledge.

How can we ensure an expert can share configured/calibrated models?

- Reduce complexity for novice users

Example: SWAT Hydrology model

- Model code is always the same
- Input files vary according to:
  - The region
  - Available information
Scientific Software encapsulation

Assistants + Guidelines

→

Machine-readable component specification

→

Tests

→

Portable Component

Workflow system

Software Metadata Registry

Software KG
Reusing software: model encapsulation methodology

- Input: software component
- Output:
  - Docker image
  - Wrapper script (GitHub)
  - Metadata (MINT model catalog)
- Powered by ReproZip (https://www.reprozip.org/) to automatically suggest I/O
Summing up
### Challenges and their dimensions

<table>
<thead>
<tr>
<th>Describe</th>
<th>Compare</th>
<th>Reuse</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Representing Research Software metadata</strong></td>
<td>![Describe Icon]</td>
<td>![Compare Icon]</td>
</tr>
<tr>
<td><strong>Knowledge capture from documentation and code</strong></td>
<td>![Describe Icon]</td>
<td>![Compare Icon]</td>
</tr>
<tr>
<td><strong>Automated encapsulation for reusability</strong></td>
<td>![Describe Icon]</td>
<td>![Compare Icon]</td>
</tr>
</tbody>
</table>
Research software is a critical asset for Open Science

Accelerating Software Understanding requires:

- Automated description
- Assisted comparison
- Easy reuse
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, Elena Montiel, Elvira Amador & all the students at UPM and USC who participated in the initiatives mentioned in this presentation.
Let's create **machine-actionable** software metadata to promote Open Science!
"WIDOCO helps you to publish and create an enriched and customized documentation of your classes, properties and data properties of the ontology, the OOPSI web service by Maria being used. In addition, we use WebVowl to visualize the ontology and have extended the documentation of the terms in your ontology (based on [LODE](http://www.esregueteate.it/)) annotation in JSON-LD snippets of the HTML produced.\n\nAssociation of a provenance page means to complete it on the fly when generating your ontology. Check the [best practices](https://widoco.org/guidelines) on the main sections that your document should have and how to do changeeloging of differences between the actual and the previous version of the ontology is then independently and replace only those needed.\n\n**Content negotiation and serialization:**

\n\n**0:**
\n**excerpt:**
```
"Wizard for documenting ontologies. WIDOCO is a step by step generator of HTML templates for the classes, properties and data properties of the ontology, the OOPSI web service by Maria being used. In addition, we use WebVowl to visualize the ontology and have extended the documentation of the terms in your ontology (based on [LODE](http://www.esregueteate.it/)) annotation in JSON-LD snippets of the HTML produced.\n\nAssociation of a provenance page means to complete it on the fly when generating your ontology. Check the [best practices](https://widoco.org/guidelines) on the main sections that your document should have and how to do changeeloging of differences between the actual and the previous version of the ontology is then independently and replace only those needed.\n\n**Content negotiation and serialization:**
```
\n**1:**
\n**excerpt:**
```
"For a complete list of the current improvements and next features, check the project documentation on [GitHub](https://github.com/MyProject)/ and the project documentation on [GitHub](https://github.com/MyProject)/
```
\n**2:**
\n**excerpt:**
```
"For a complete list of the current improvements and next features, check the project documentation on [GitHub](https://github.com/MyProject)/ and the project documentation on [GitHub](https://github.com/MyProject)/
```
\n**citation:**
```
@context: "https://doi.org/10.5281/schem/CodeMeta"  
@type: "SoftwareSourceCode"  
license: "https://raw.githubusercontent.com/dgarilo/Widoco/master/LICENSE"  
codeRepository: "git:https://github.com/dgarilo/Widoco.git"  
dateCreated: "2013-07-13"  
datePublished: "2020-12-14"  
dateModified: "2021-03-10"  
downloadUrl: "https://github.com/dgarilo/Widoco/releases"  
issueTracker: "https://github.com/dgarilo/Widoco/issue"  
name: "Widoco"  
version: "v1.4.15.1"  
\n\n**description:**
```
"Wizard for documenting ontologies. WIDOCO is a step by step generator of HTML templates for the classes, properties and data properties of the ontology, the OOPSI web service by Maria being used. In addition, we use WebVowl to visualize the ontology and have extended the documentation of the terms in your ontology (based on [LODE](http://www.esregueteate.it/)) annotation in JSON-LD snippets of the HTML produced.\n\nAssociation of a provenance page means to complete it on the fly when generating your ontology. Check the [best practices](https://widoco.org/guidelines) on the main sections that your document should have and how to do changeeloging of differences between the actual and the previous version of the ontology is then independently and replace only those needed.\n\n**Content negotiation and serialization:**
```
\n**0:**
```
"For a complete list of the current improvements and next features, check the project documentation on [GitHub](https://github.com/MyProject)/ and the project documentation on [GitHub](https://github.com/MyProject)/
```
\n**1:**
```
"For a complete list of the current improvements and next features, check the project documentation on [GitHub](https://github.com/MyProject)/ and the project documentation on [GitHub](https://github.com/MyProject)/
```
\n**2:**
```
"For a complete list of the current improvements and next features, check the project documentation on [GitHub](https://github.com/MyProject)/ and the project documentation on [GitHub](https://github.com/MyProject)/
```
\n**keywords:**
```
"ontology"  
"wizard"  
"metadata"  
"documentation"  
"ontology-diagram"  
"ontology-evaluation"
```
\n\n"WIDOCO helps you to publish and create an enriched and customized documentation of your classes, properties and data properties of the ontology, the OOPSI web service by Maria being used. In addition, we use WebVowl to visualize the ontology and have extended the documentation of the terms in your ontology (based on [LODE](http://www.esregueteate.it/)) annotation in JSON-LD snippets of the HTML produced.\n\nAssociation of a provenance page means to complete it on the fly when generating your ontology. Check the [best practices](https://widoco.org/guidelines) on the main sections that your document should have and how to do changeeloging of differences between the actual and the previous version of the ontology is then independently and replace only those needed.\n\n**Content negotiation and serialization:**
```
\n**0:**
```
"For a complete list of the current improvements and next features, check the project documentation on [GitHub](https://github.com/MyProject)/ and the project documentation on [GitHub](https://github.com/MyProject)/
```
\n**1:**
```
"For a complete list of the current improvements and next features, check the project documentation on [GitHub](https://github.com/MyProject)/ and the project documentation on [GitHub](https://github.com/MyProject)/
```
\n**2:**
```
"For a complete list of the current improvements and next features, check the project documentation on [GitHub](https://github.com/MyProject)/ and the project documentation on [GitHub](https://github.com/MyProject)/
```
\n**citation:**
```
@context: "https://doi.org/10.5281/schem/CodeMeta"  
@type: "SoftwareSourceCode"  
license: "https://raw.githubusercontent.com/dgarilo/Widoco/master/LICENSE"  
codeRepository: "git:https://github.com/dgarilo/Widoco.git"  
dateCreated: "2013-07-13"  
datePublished: "2020-12-14"  
dateModified: "2021-03-10"  
downloadUrl: "https://github.com/dgarilo/Widoco/releases"  
issueTracker: "https://github.com/dgarilo/Widoco/issue"  
name: "Widoco"  
version: "v1.4.15.1"  
\n\n**description:**
```
"Wizard for documenting ontologies. WIDOCO is a step by step generator of HTML templates for the classes, properties and data properties of the ontology, the OOPSI web service by Maria being used. In addition, we use WebVowl to visualize the ontology and have extended the documentation of the terms in your ontology (based on [LODE](http://www.esregueteate.it/)) annotation in JSON-LD snippets of the HTML produced.\n\nAssociation of a provenance page means to complete it on the fly when generating your ontology. Check the [best practices](https://widoco.org/guidelines) on the main sections that your document should have and how to do changeeloging of differences between the actual and the previous version of the ontology is then independently and replace only those needed.\n\n**Content negotiation and serialization:**
```
\n**0:**
```
"For a complete list of the current improvements and next features, check the project documentation on [GitHub](https://github.com/MyProject)/ and the project documentation on [GitHub](https://github.com/MyProject)/
```
\n**1:**
```
"For a complete list of the current improvements and next features, check the project documentation on [GitHub](https://github.com/MyProject)/ and the project documentation on [GitHub](https://github.com/MyProject)/
```
\n**2:**
```
"For a complete list of the current improvements and next features, check the project documentation on [GitHub](https://github.com/MyProject)/ and the project documentation on [GitHub](https://github.com/MyProject)/
"