Accelerating Research Software Understandability Through Knowledge Capture

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

daniel.garijo@upm.es

@dgarijov

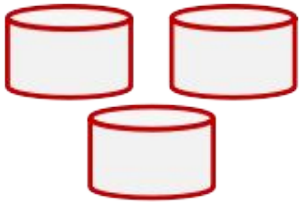
Research Software is one of the pillars of Open Science

Research Data

Research Software

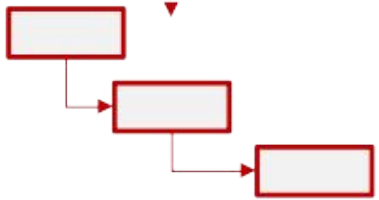
Research Methods

Research Software is one of the pillars of Open Science



Research Data

Research Software



Research Methods

figshare

github SOCIAL CODE HOSTING

Apache

PLOS

DRYAD

The Dataverse Network Project

GitLab

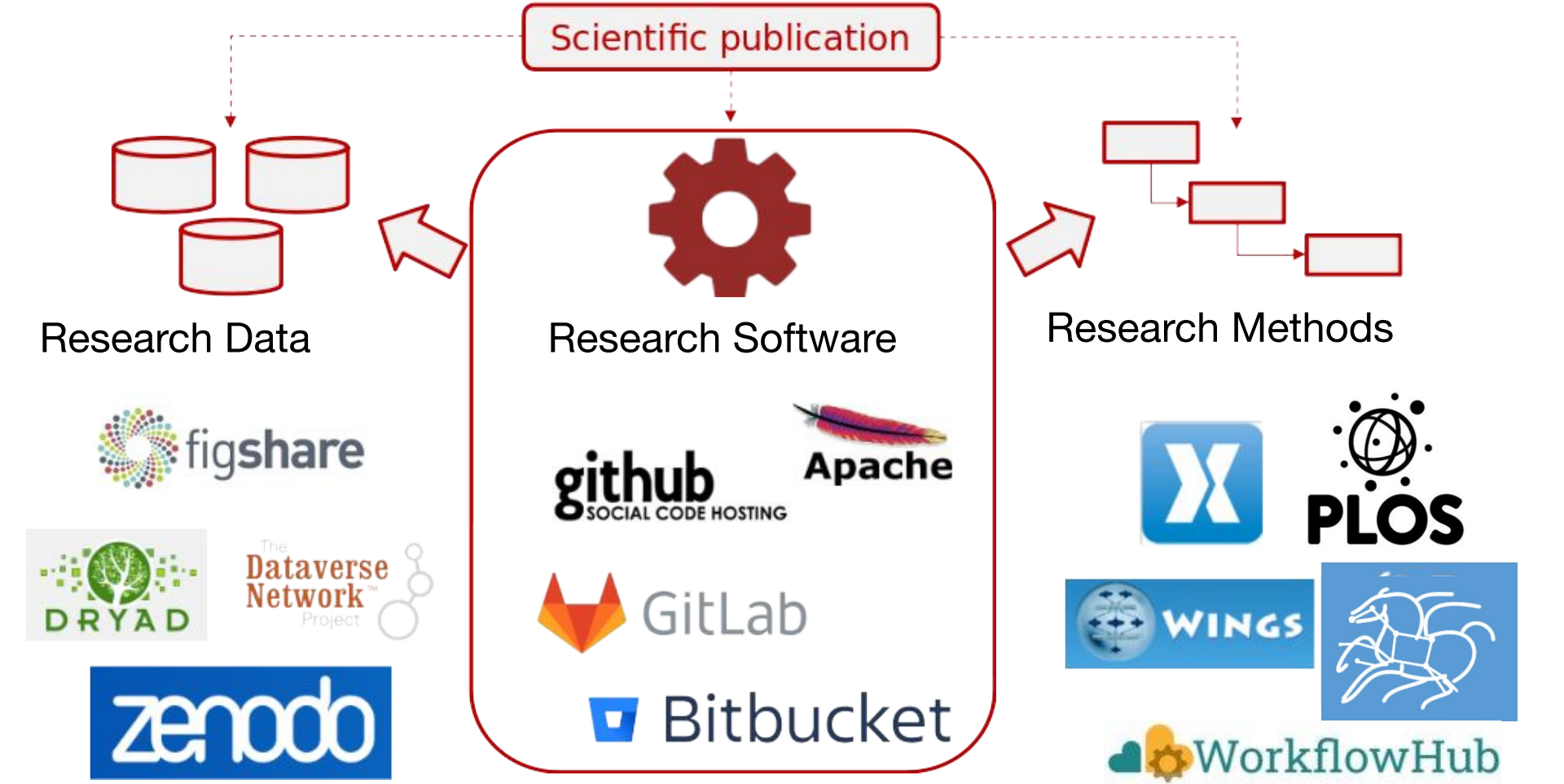
WINGS

zenodo

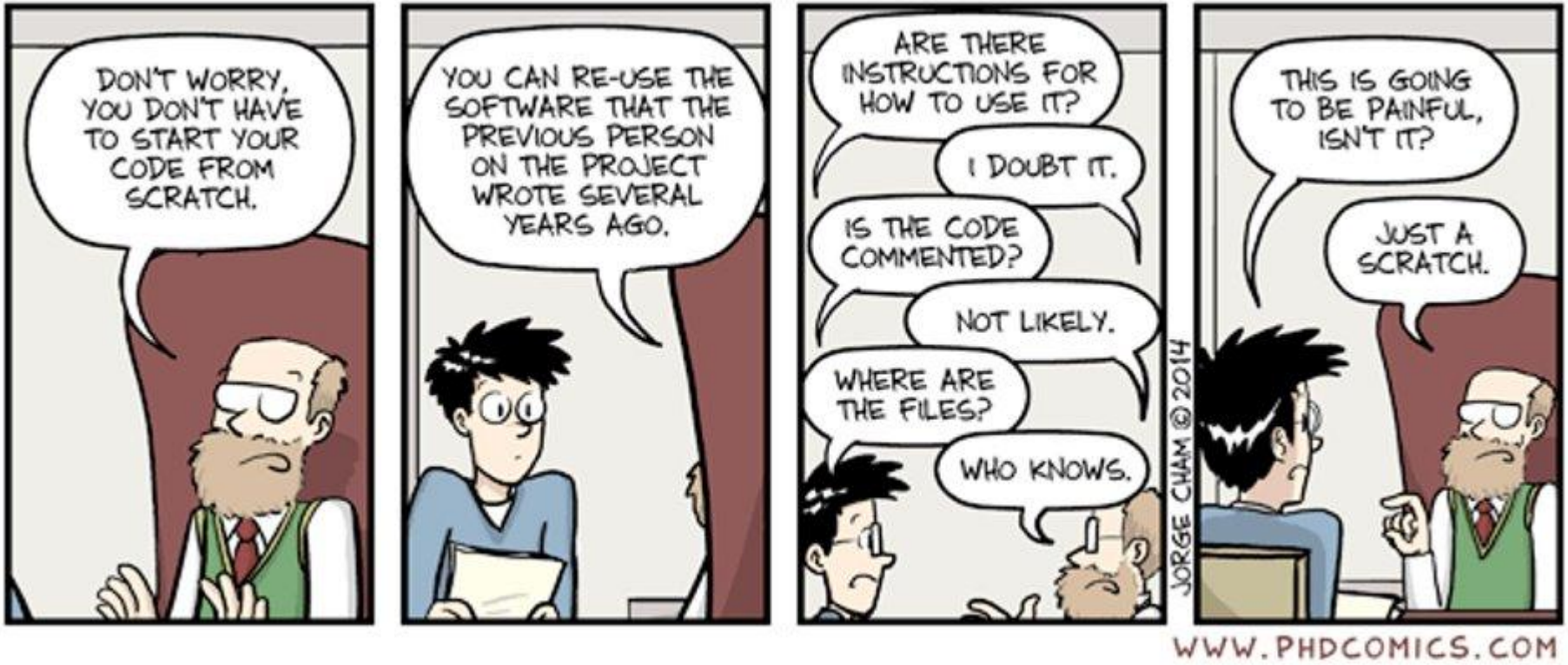
Bitbucket

WorkflowHub

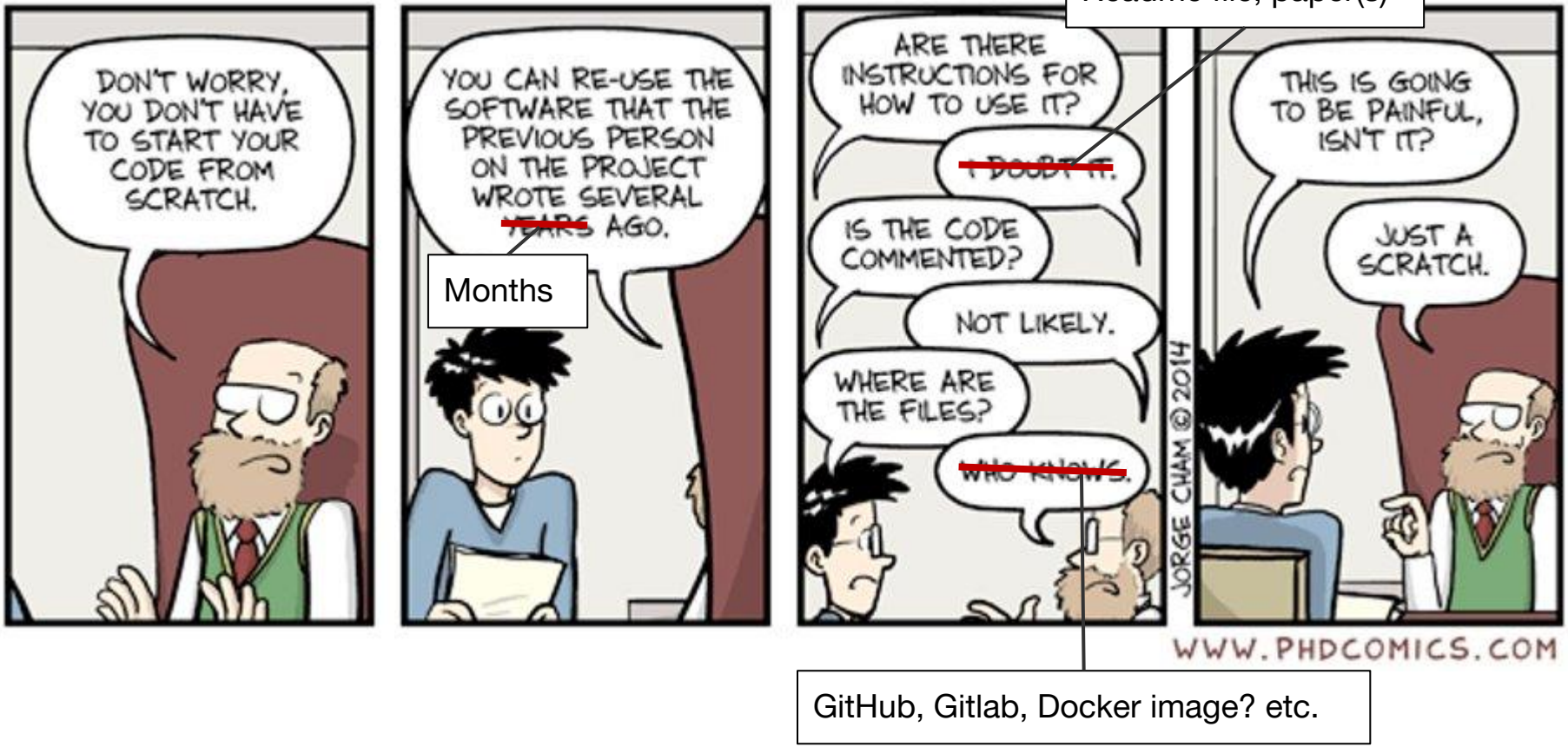
Research Software is one of the pillars of Open Science



Is this situation familiar?



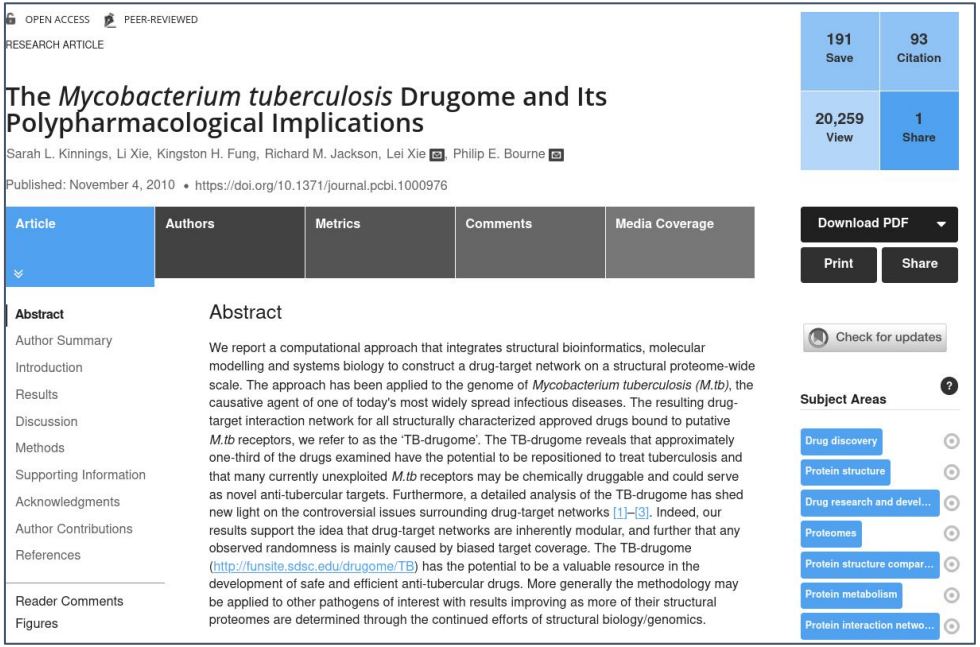
Or this one?


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.
Months
ARE THERE INSTRUCTIONS FOR HOW TO USE IT?
~~I DOUBT IT.~~
IS THE CODE COMMENTED?
NOT LIKELY.
WHERE ARE THE FILES?
~~WHO KNOWS.~~
JORGE CHAM © 2014
THIS IS GOING TO BE PAINFUL, ISN'T IT?
JUST A SCRATCH.
WWW.PHDCOMICS.COM
Readme file, paper(s)
GitHub, Gitlab, Docker image? etc.

Reusability takes time, even when sources are available

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)



OPEN ACCESS PEER-REVIEWED

RESEARCH ARTICLE

The *Mycobacterium tuberculosis* Drugome and Its Polypharmacological Implications

Sarah L. Kinnings, Li Xie, Kingston H. Fung, Richard M. Jackson, Lei Xie, Philip E. Bourne

Published: November 4, 2010 • https://doi.org/10.1371/journal.pcbi.1000976

191 Save | 93 Citation | 20,259 View | 1 Share

Article | Authors | Metrics | Comments | Media Coverage

Download PDF | Print | Share

Abstract
Author Summary
Introduction
Results
Discussion
Methods
Supporting Information
Acknowledgments
Author Contributions
References
Reader Comments
Figures

Abstract

We report a computational approach that integrates structural bioinformatics, molecular modelling and systems biology to construct a drug-target network on a structural proteome-wide scale. The approach has been applied to the genome of *Mycobacterium tuberculosis (M.tb)*, the causative agent of one of today's most widely spread infectious diseases. The resulting drug-target interaction network for all structurally characterized approved drugs bound to putative *M.tb* receptors, we refer to as the 'TB-drugome'. The TB-drugome reveals that approximately one-third of the drugs examined have the potential to be repositioned to treat tuberculosis and that many currently unexploited *M.tb* receptors may be chemically druggable and could serve as novel anti-tubercular targets. Furthermore, a detailed analysis of the TB-drugome has shed new light on the controversial issues surrounding drug-target networks [1]–[3]. Indeed, our results support the idea that drug-target networks are inherently modular, and further that any observed randomness is mainly caused by biased target coverage. The TB-drugome (http://funsite.sdsc.edu/drugome/TB) has the potential to be a valuable resource in the development of safe and efficient anti-tubercular drugs. More generally the methodology may be applied to other pathogens of interest with results improving as more of their structural proteomes are determined through the continued efforts of structural biology/genomics.

Check for updates

Subject Areas

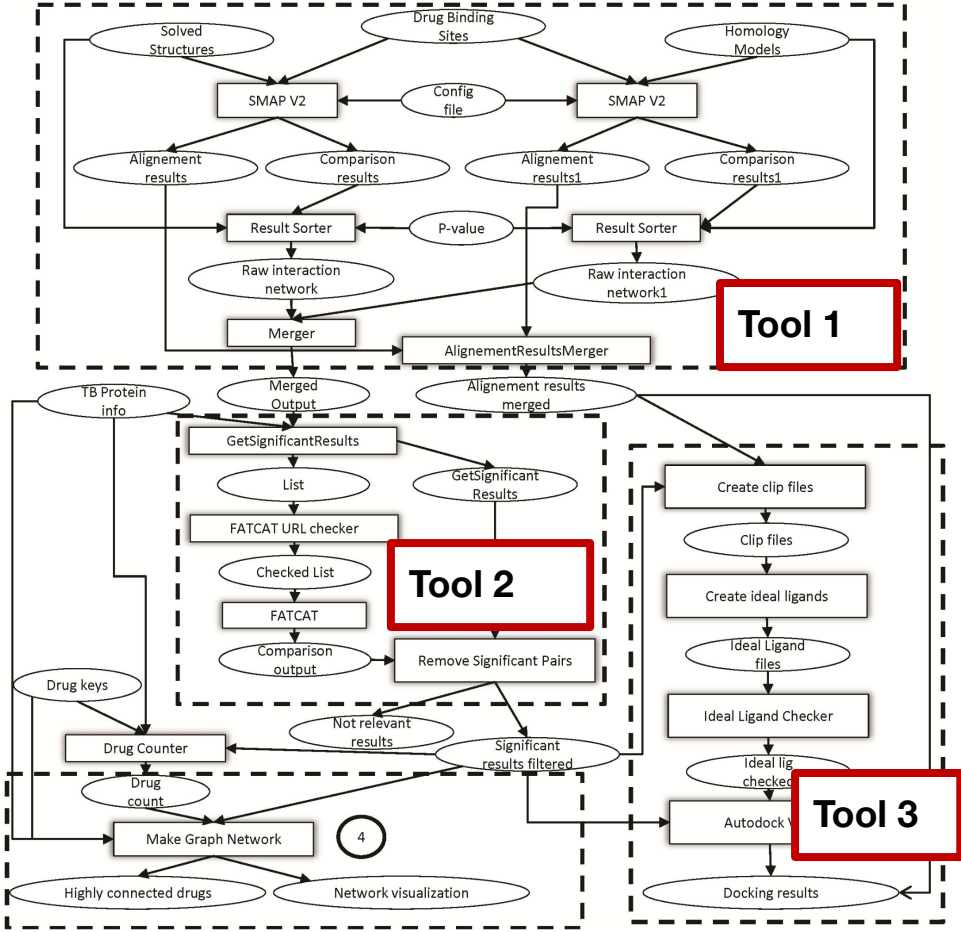
Drug discovery | Protein structure | Drug research and devel... | Proteomes | Protein structure compar... | Protein metabolism | Protein interaction netwo...

[1] Garijo, D., Kinnings, S., Xie, L., Xie, L., Zhang, Y., Bourne, P. E., & Gil, Y. (2013). Quantifying reproducibility in computational biology: the case of the tuberculosis drugome. *PloS one*, *8*(11), e80278.

Reusability takes time, even when sources are available

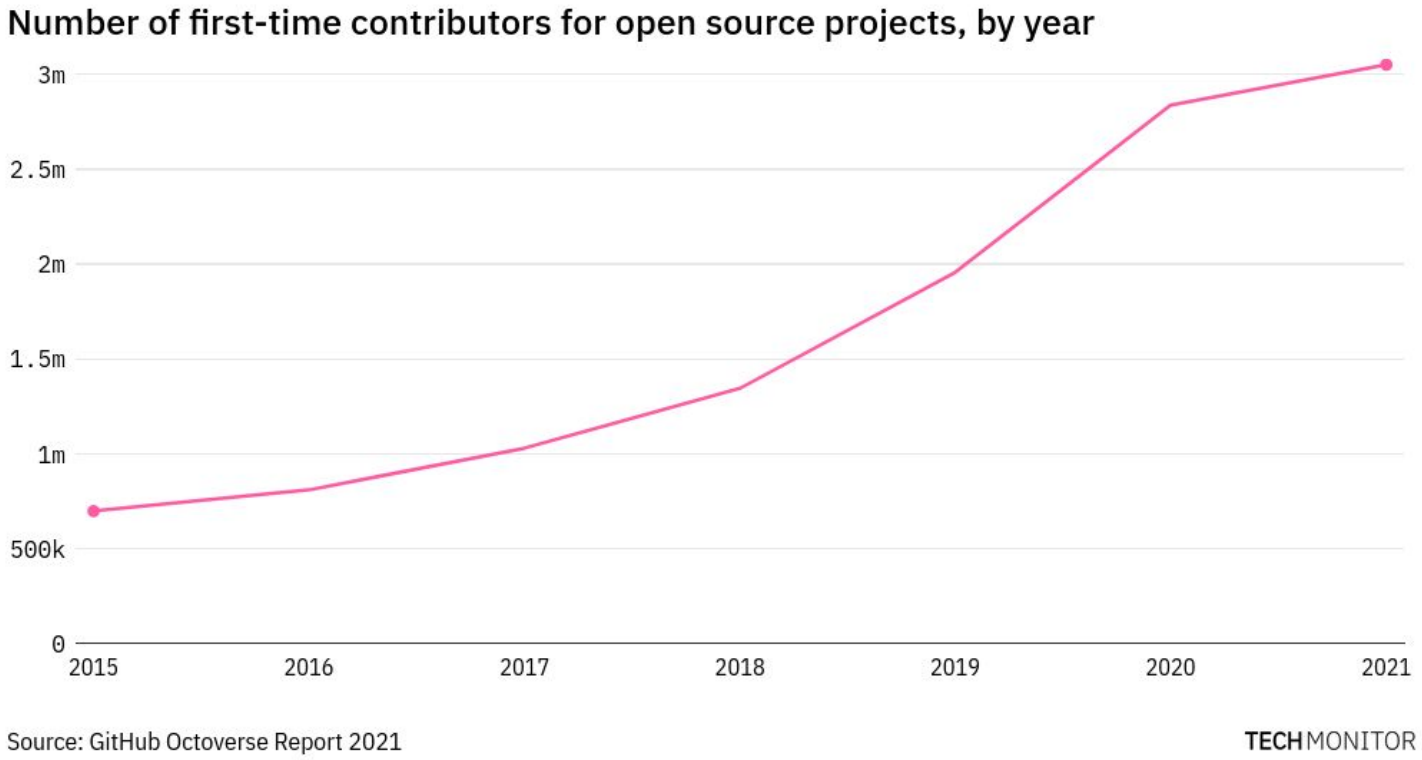
In [1] we tried to reproduce an effort from **one** year before.

- 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



[1] Garijo, D., Kinnings, S., Xie, L., Xie, L., Zhang, Y., Bourne, P. E., & Gil, Y. (2013). Quantifying reproducibility in computational biology: the case of the tuberculosis drugome. *PloS one*, *8*(11), e80278.

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:

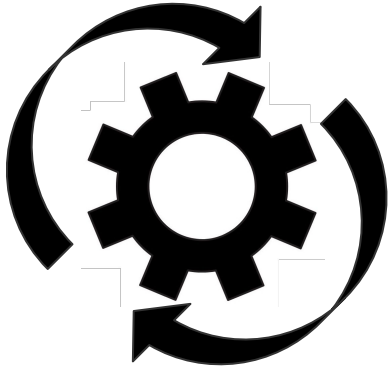
- What is it about?
- Examples?
- Relation to other resources (data, papers)?
- Metadata?

Compare

Given two or more tools:

- What are their similarities?
- Differences?
- Main features?

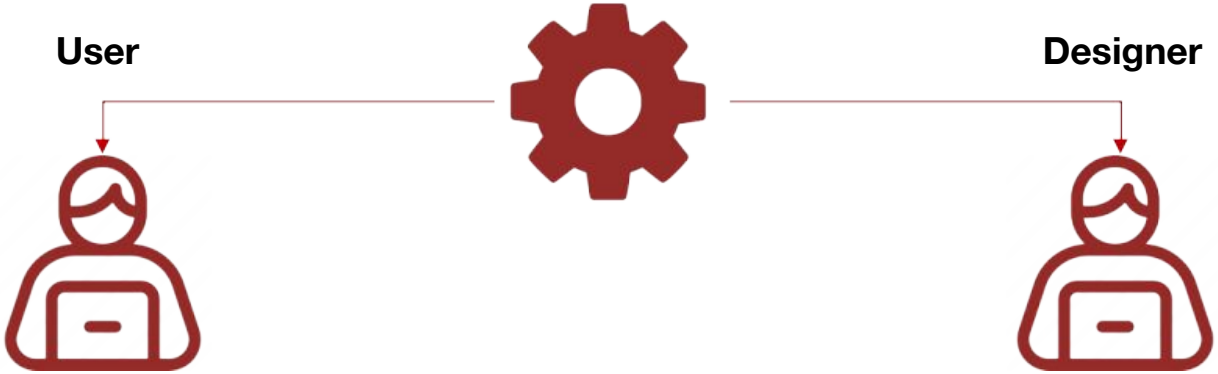
Reuse



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?

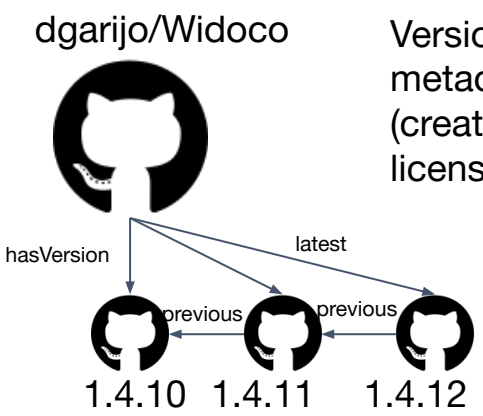
- How to…
  a. Ease capturing the dependencies and installation instructions of my software?
  b. Encapsulate my software so it can be used with other data?
  c. Describe my software so it can be used by others?
  d. Test if my software is ready to be used by others?

Three challenges for Research Software Understanding

1. Representing Research Software metadata
2. Knowledge capture from documentation and code
3. Automated encapsulation for reusability

1. **Representing** Research Software **metadata**
2. Knowledge capture from documentation and code
3. Automated encapsulation for reusability

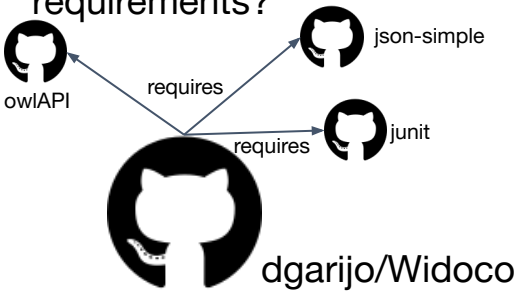
Research problem: Wide Research Software landscape



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

Contributions?
Development process?

Dependencies and requirements?



Execution command and configuration

How to use WIDOCO

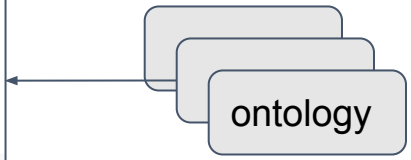
JAR execution

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

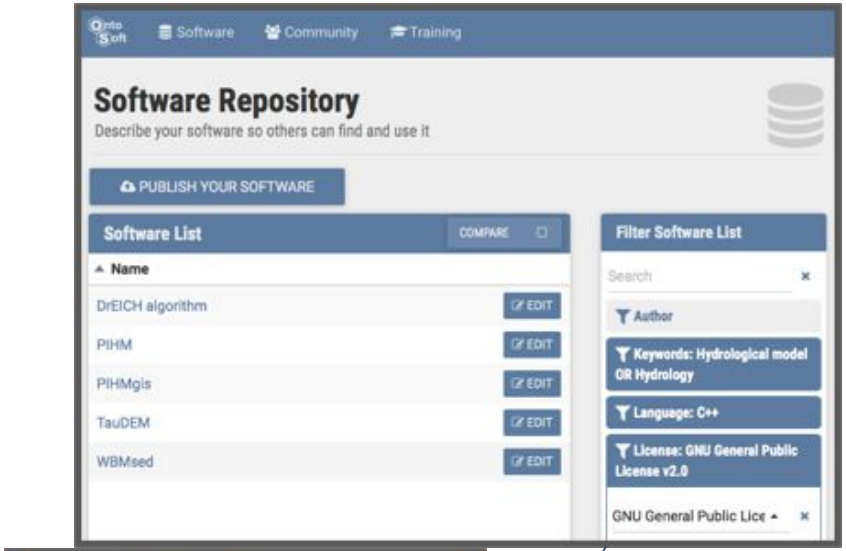
You may also execute WIDOCO through the command line. Usage:

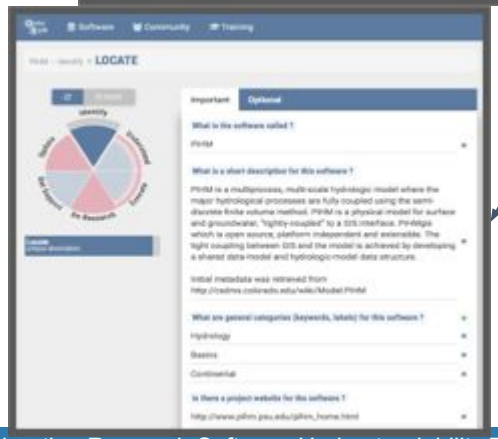
```
java -jar widoco-VERSION-jar-with-dependencies.jar [OPTIONS]
```

Supporting materials?
Input data?



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

OntoSoft: Capturing Scientific Software Metadata. Eighth ACM International Conference on Knowledge Capture, Palisades, NY, 2015

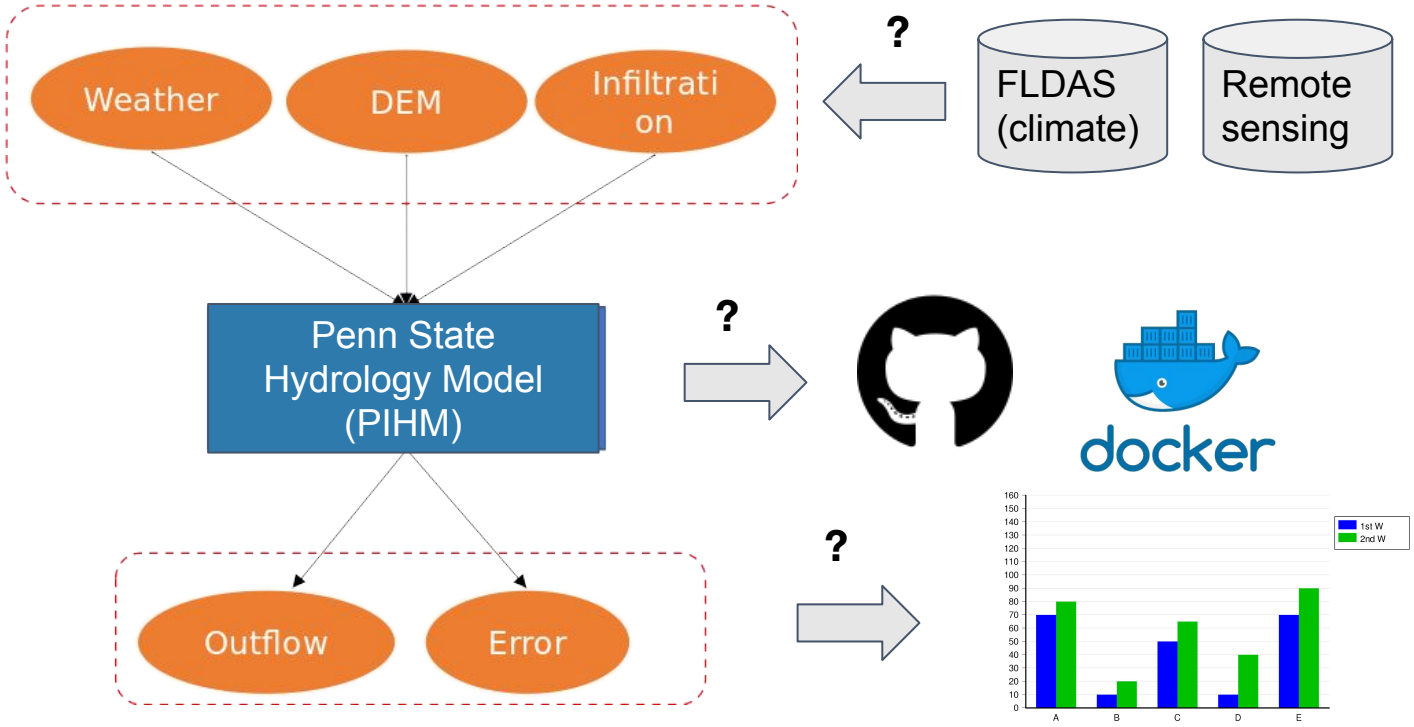
Onto Soft | Software | Community | Training

Compare Software

DrEICH algorithm, PIHM, PIHMgis, TauDEM, WBMsed

	PIHM	PIHMgis	DrEICH	TauDEM	WBMsed
					
What are domain specific keywords for this software ? (eg: hydrology, climate)					
	Geomorphology, Hydrological, Bedrock channel ero-	Basins, Continental	Basins, GIS	Hydrologically corrected DEM, Watershed	Sediment flux, Global model, Hydrological model
What Operating Systems can the software run on ?					
	Unix Linux	Unix Windows Linux Mac OS	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://onlinelibrary.wiley.-com/doi/10.1002/2013WR015167/full **Test Data Description:** Two test DEMs are included in the repository,	**Test Data Location:** http://source-forge.net/projects/pih-mmodel/ **Test Data Description:** Upper Juniata River 875 km^2: see: http://source-		**Test Data Location:** http://csdms.colorado.e-du/wiki/Model:Tau-DEM#Testing **Test Data Description:** The Logan River DEM is a small test dataset useful	**Test Data Location:** http://csdms.colorado.e-du/wiki/Mod-el:WBMsed#Testing **Test Data Description:** Extensive input dataset is available on the CSDMS

Describing inputs, outputs and their structure



Adding structure to software Metadata: MINT (Model Integration)

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

IO Files:

	Name	Description
INPUT	pihm-riv	Spatial geometry and material information of river segments
INPUT	pihm-geol	Geologic file
INPUT	pihm-ibc	Boundary condition information for elements
INPUT	pihm-modelinfo	PIHM model information aggreagation file
INPUT	pihm-lc	Vegetation parameters of different land cover types
INPUT	pihm-base	Base file
INPUT	pihm-forc	PIHM forcing file with the majority of the relevant variables
INPUT	pihm-soil	Soil parameters for the soil types
INPUT	pihm-att	PIHM attribute file with index values of variables for timeseries
OUTPUT	pihm-et0	Evaporation canopy file
OUTPUT	pihm-rivFlx9	lateral outflux to the bed beneath river
OUTPUT	pihm-rivFlx4	Baseflow to stream reach from aquifer on the left
OUTPUT	pihm-rech	Recharge Rate file
OUTPUT	pihm-rivFlx10	lateral influx to the bed beneath river
OUTPUT	pihm-infiltration	Infiltration file

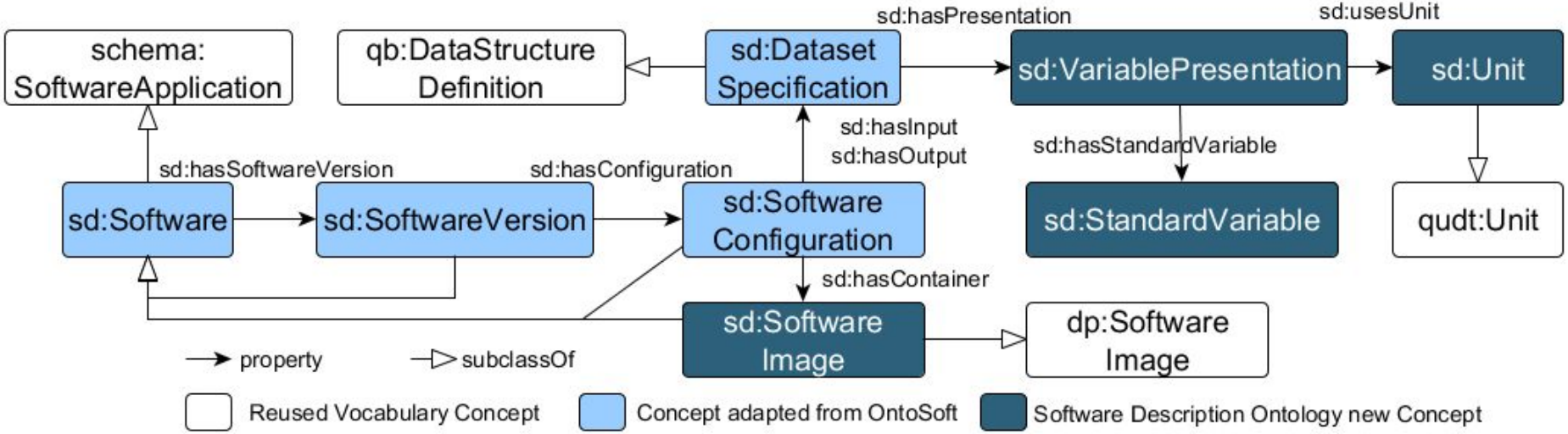
pihm-riv

Input/output variables

ormation of river segments

Label	Long Name	Description	Standard Name	Units
Bed	Bed Depth	Bed Depth	channel_bed__thickness	m
KsatV	Bed Hydraulic Conductivity	Bed Hydraulic Conductivity	soil_water__vertical_saturated_hydraulic_conductivity	m day-1
Water table value	Water table of the IC	Water table of the IC		m

Software Description Ontology



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)

CodeMeta Schema.org

https://w3id.org/okn/o/sd

Garijo, D., Osorio, M., Khider, D., Ratnakar, V., & Gil, Y. (2019, September). OKG-Soft: An open knowledge graph with machine readable scientific software metadata. In *2019 15th International Conference on eScience (eScience)* (pp. 349-358). IEEE.

Beyond software projects: Containers

Software images are created from configuration files (e.g., Dockerfiles)

- Execution of a tool
- Configuration
- Dependencies
- Infrastructure

Helpful for **reproducibility**

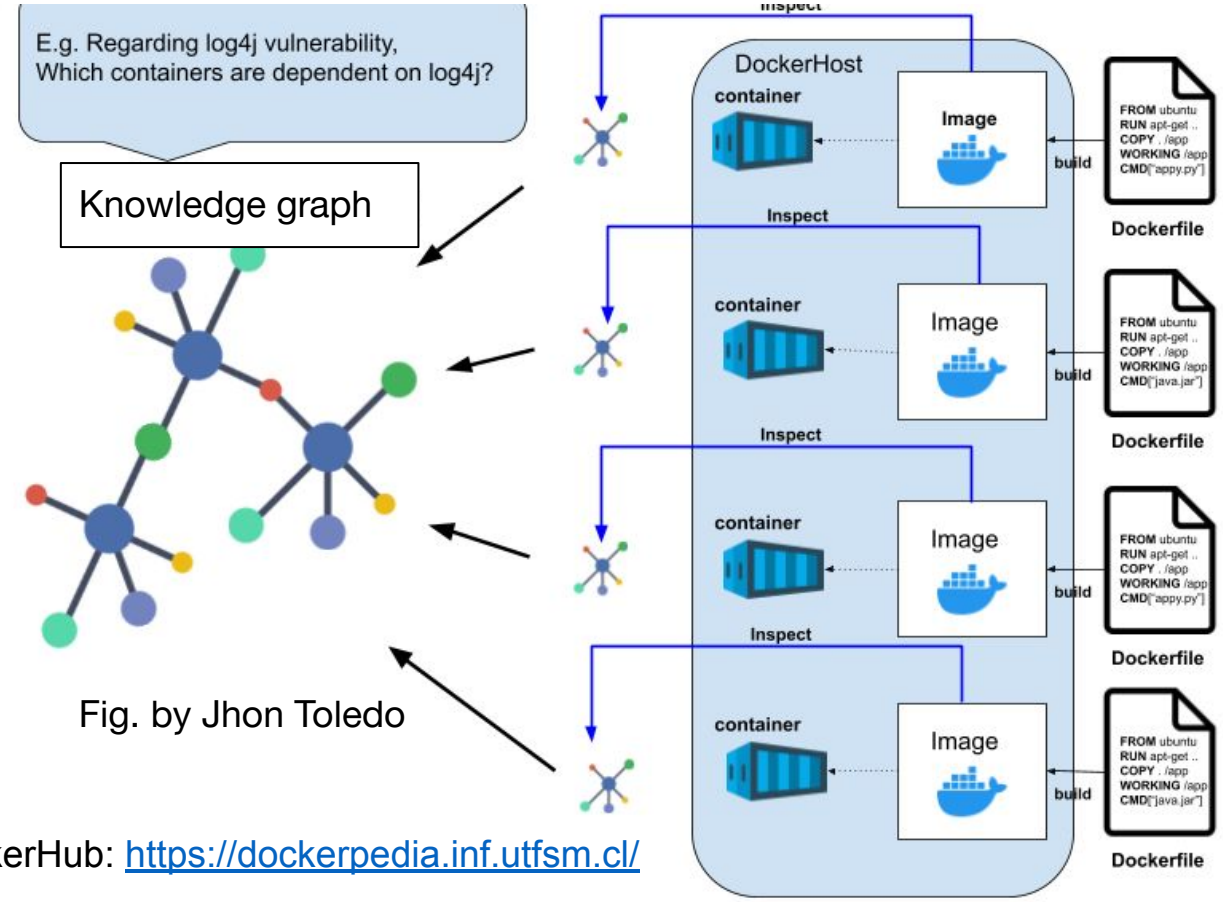


Fig. by Jhon Toledo

Initial effort transforming part of DockerHub: https://dockerpedia.inf.utfsm.cl/

Osorio, M., Buil-Aranda, C., Santana-Perez, I., & Garijo, D. (2022). DockerPedia: A Knowledge Graph of Software Images and Their Metadata. *International Journal of Software Engineering and Knowledge Engineering*, *32*(01), 71-89.

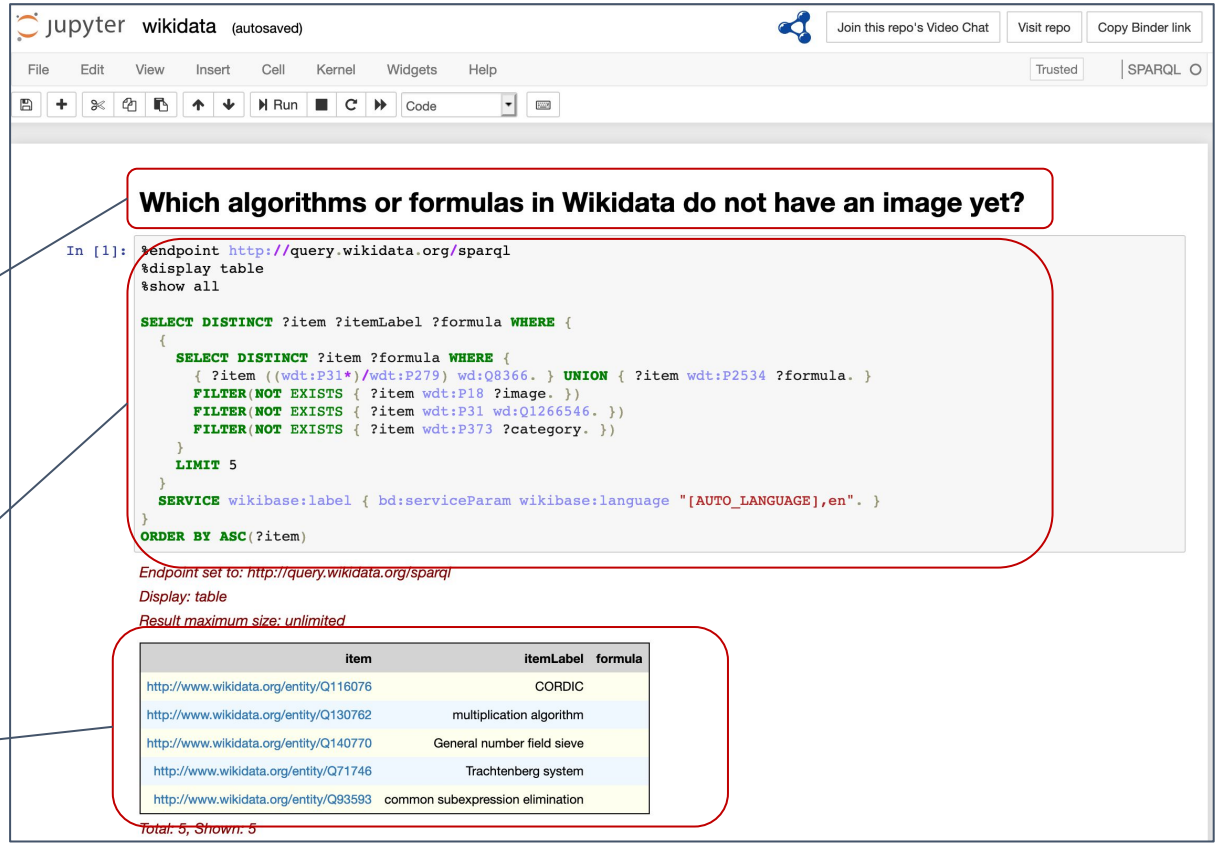
Notebooks contain crucial examples to understand scientific projects:

- Demos
- Tutorials
- Configuration

Basic metadata (e.g., title)

Configuration / queries needed

Intermediate results



1. Representing Research Software metadata
2. **Knowledge capture from documentation and code**
3. Automated encapsulation for reusability

Research Software metadata is not ~~abundant~~ machine readable

Can you please describe your software component with metadata?

I already did! Did you read the project readme?

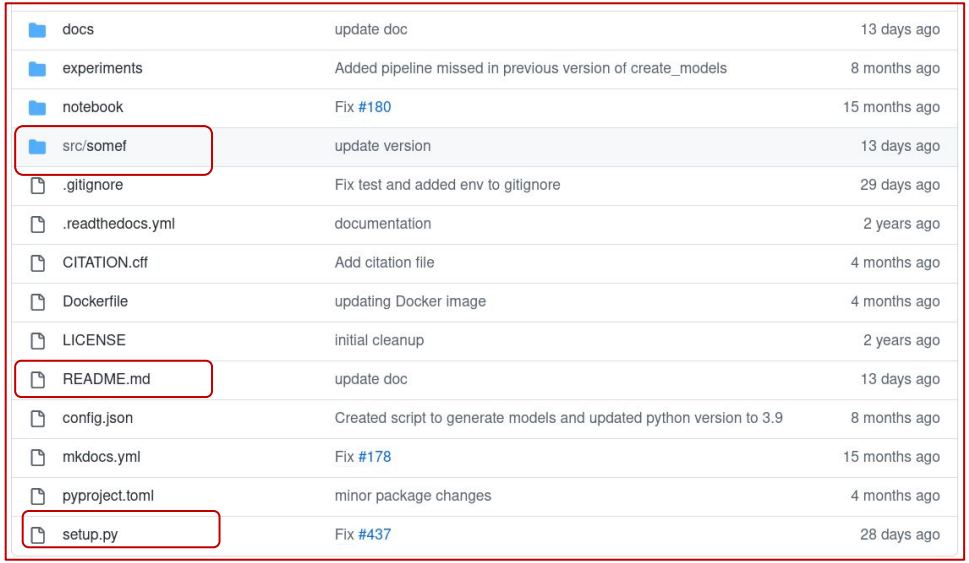
Did you see the online documentation?

Perhaps the you saw the paper?

Many domain-specific registries are curated by hand by experts

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

- Code
  - Static code analysis

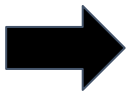


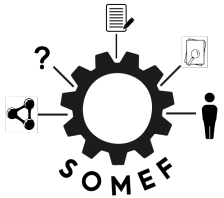
docs	update doc	13 days ago
experiments	Added pipeline missed in previous version of create_models	8 months ago
notebook	Fix #180	15 months ago
src/somef	update version	13 days ago
.gitignore	Fix test and added env to gitignore	29 days ago
.readthedocs.yml	documentation	2 years ago
CITATION.cff	Add citation file	4 months ago
Dockerfile	updating Docker image	4 months ago
LICENSE	initial cleanup	2 years ago
README.md	update doc	13 days ago
config.json	Created script to generate models and updated python version to 3.9	8 months ago
mkdocs.yml	Fix #178	15 months ago
pyproject.toml	minor package changes	4 months ago
setup.py	Fix #437	28 days ago

Text classification: Software Metadata Extraction Framework

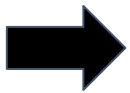
https://github.com/KnowledgeCaptureAndDiscovery/somef/

Repository

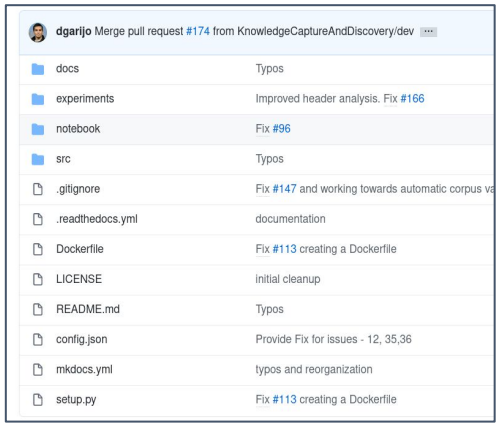




Extraction



Results (Metadata)



- **Readme Analysis**
  - Supervised classification
  - Regular expressions
  - Header analysis
- **File exploration**
  - Notebooks
  - Dockerfiles
  - Documentation
- **GitHub API**

CodeMeta

Kelley, A., & Garijo, D. (2021). A framework for creating knowledge graphs of scientific software metadata. *Quantitative Science Studies*, 1-37.

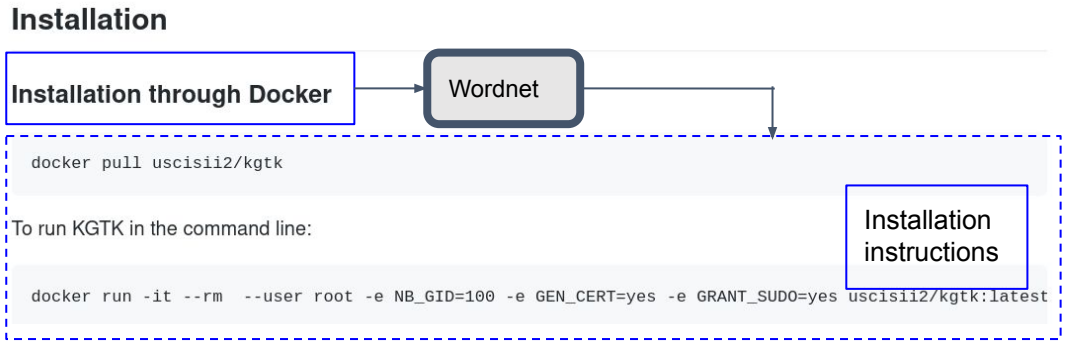
- Paragraph-based text classification
- Four main categories (binary classification):
  - Installation
  - Citation
  - Description
  - Invocation

Truth Value	Category	Apprx. Ratio	Count
True	Description	0.5	275
False	Installation	0.125	68
	Invocation	0.125	68
	Citation	0.125	68
	Treebank	0.125	68
	Total	1.0	547

Classifier	Best pipeline	Precision	Recall	F-Measure
Description	CountVectorizer + LogisticRegression	0.85	0.79	0.82
Installation	TFIDFVectorizer + StochasticGradientDescent	0.92	0.9	0.91
Invocation	CountVectorizer + NaiveBayes	0.88	0.9	0.89
Citation	CountVectorizer + NaiveBayes	0.89	0.98	0.93

Simple classification pipelines yield nice results

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



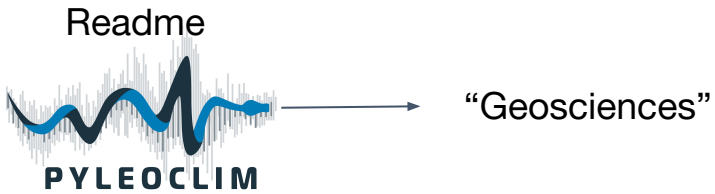
Regular expressions, based on common practices (e.g., DOI, .bib, etc.)

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:

SOMEF: Classifying software types based on README files

Using READMEs to categorize software

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



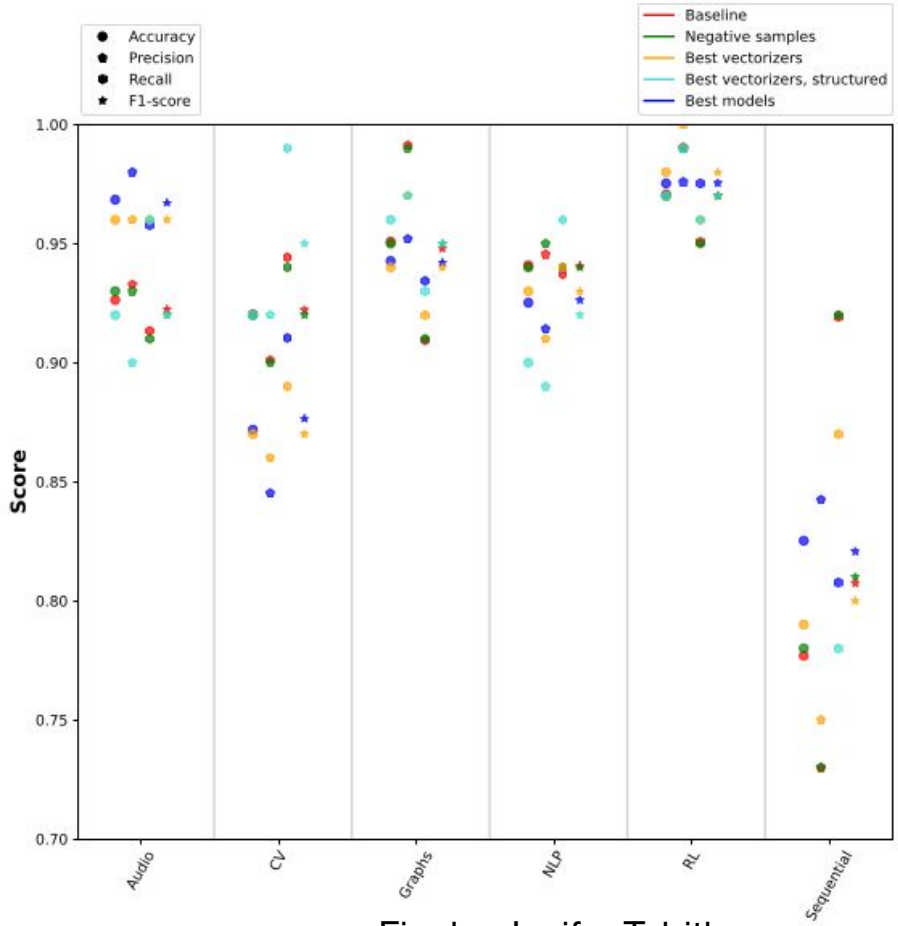


Fig. by Jenifer Tabitha

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)

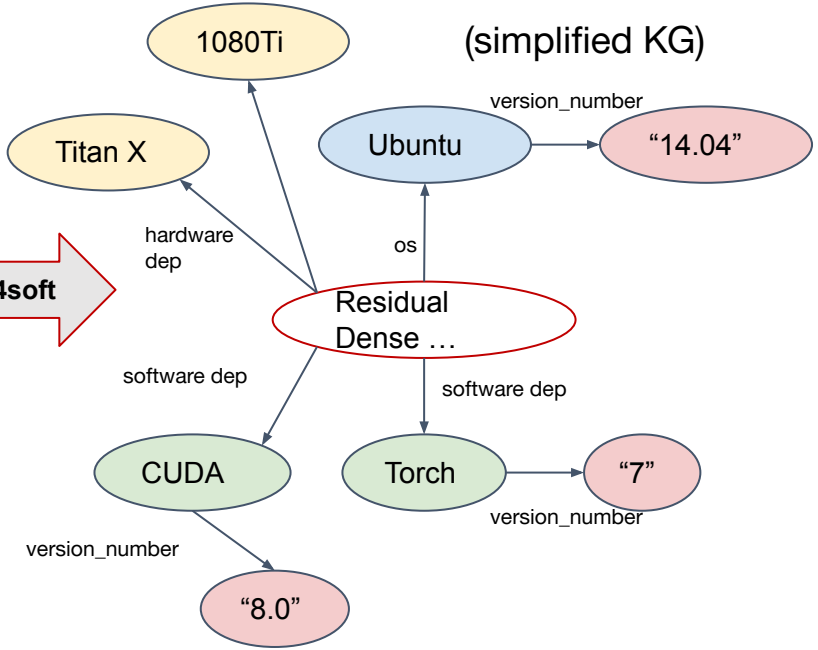
- **Contact** (HA)
- **Download URL** (HA, GA)
- **DOI** (RE)
- **DockerFile** (FE)
- **Notebooks** (FE)
- **Executable notebooks (Binder, Collab)** (RE)
- **Owner**: (GA)
- **Keywords** (GA)
- **Source code** (GA)
- **Releases** (GA)
- **Changelog** (GA)
- **Issue tracker** (GA)
- **Programming languages** (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)**
- **…**

Readme file

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.

ner4soft



- Hardware dep.
- Software dep.
- Operative System
- Version.

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)

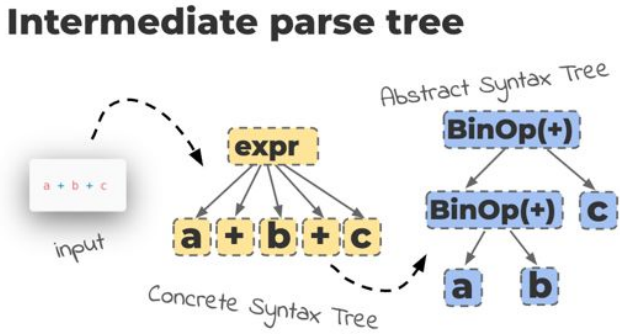
Static code analysis in Python

- 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

Inspect4py

Rosa Filgueira

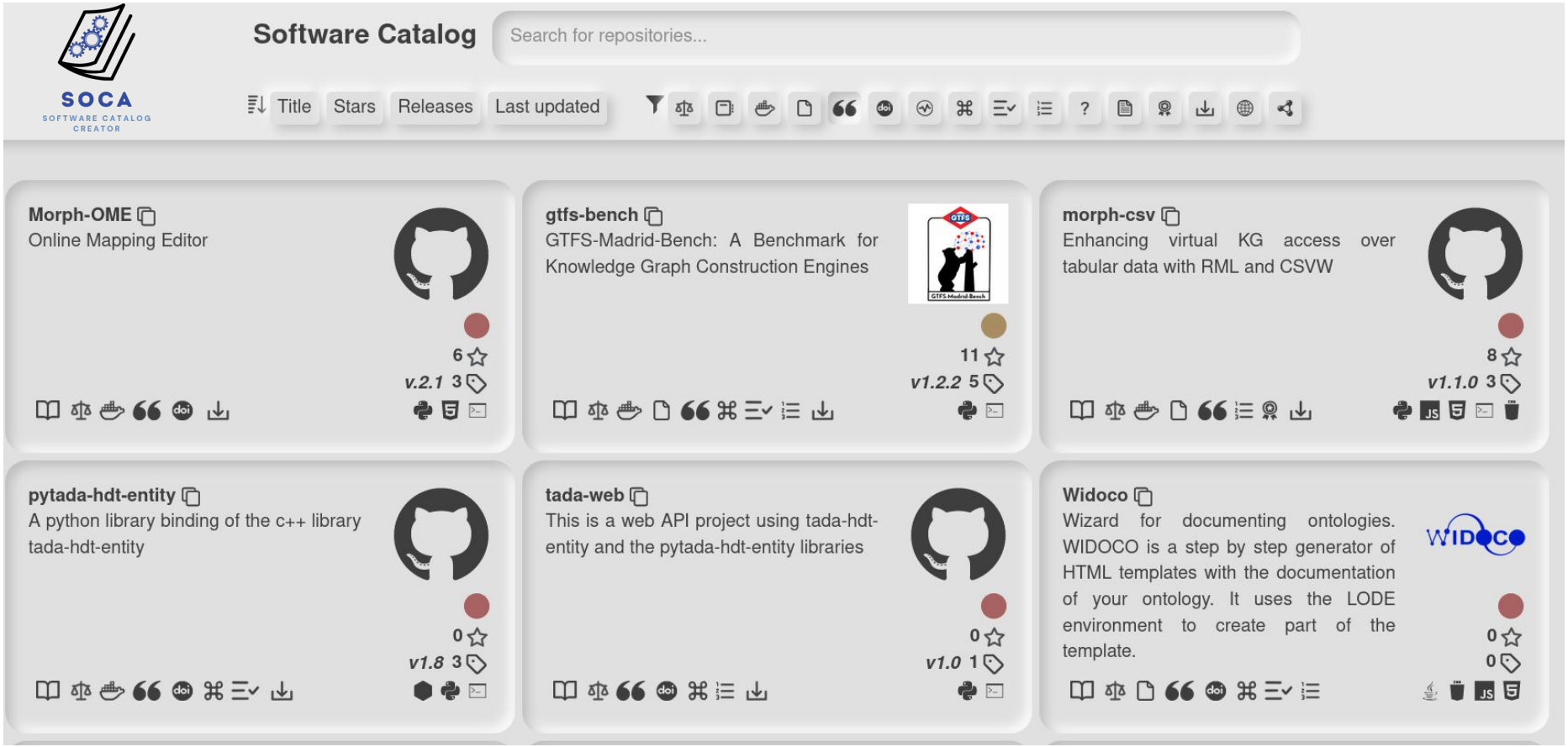
University of St Andrews



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

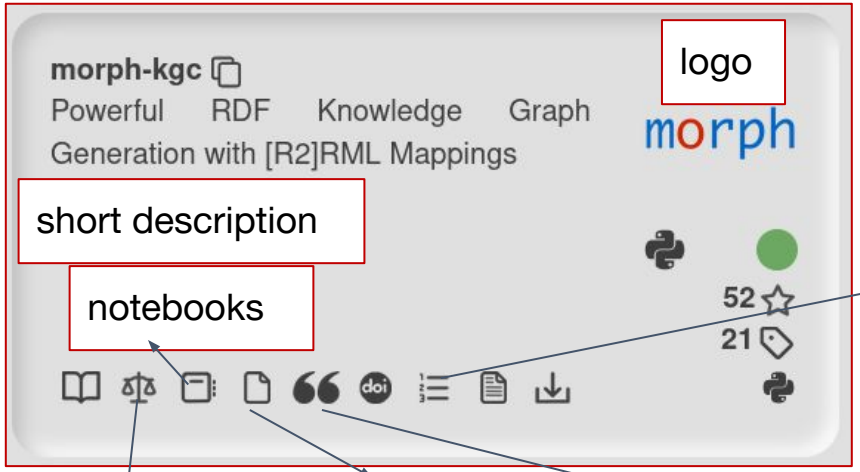
Filgueira, R. and Garijo, D. (2022). Inspect4py: A Knowledge Extraction Framework for Python Code Repositories. To appear in Mining Software Repositories, 2022 (demo)

Early result: Automated software catalogs



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

A software repository at a glance



Usage

Learn quickly with the tutorial in **Google Colaboratory**!
PyPi is the fastest way to install Morph-KGC:
pip install morph-kgc
We recommend to use **virtual environments** to install Morph-KGC.
To run the engine via **command line** you just need to execute the following:
python3 -m morph_kgc config.ini
Check the **documentation** to can see how to generate the configuration **INI file**.
Here you can also see an example INI file.
It is also possible to run Morph-KGC as a **library** with **RDFLib** and **Oxigraph**:

```
import morph_kgc

# generate the triples and load them to an RDFLib graph
g_rdflib = morph_kgc.materialize('/path/to/config.ini')
# work with the RDFLib graph
q_res = g_rdflib.query(' SELECT DISTINCT ?classes WHERE { ?s a ?classes } ')

# generate the triples and load them to Oxigraph
g_oxigraph = morph_kgc.materialize_oxigraph('/path/to/config.ini')
# work with Oxigraph
q_res = graph.query(' SELECT DISTINCT ?classes WHERE { ?s a ?classes } ')

# the methods above also accept the config as a string
config = """
        [DataSource1]
        mappings: /path/to/mapping/mapping_file.rml.ttl
        db_url: mysql+pymysql://user:password@localhos
    """
g_rdflib = morph_kgc.materialize(config)
```

How to use it
python /morph-kgc/oeg-upm_morph-kgc/morph-kgc-main/src/morph_kgc/**main**.py

invocation

{Morph-KGC: Scalable Knowledge Graph Materialization with Mapping Partitions}

License

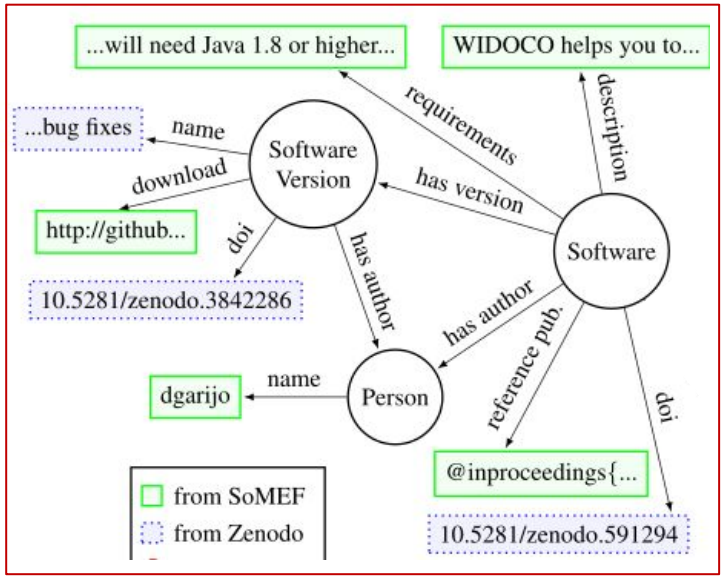
Apache License 2.0
Description:
A permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code.
Permissions:
1. Commercial-use
2. Modifications
3. Distribution
4. Patent-use
5. Private-use

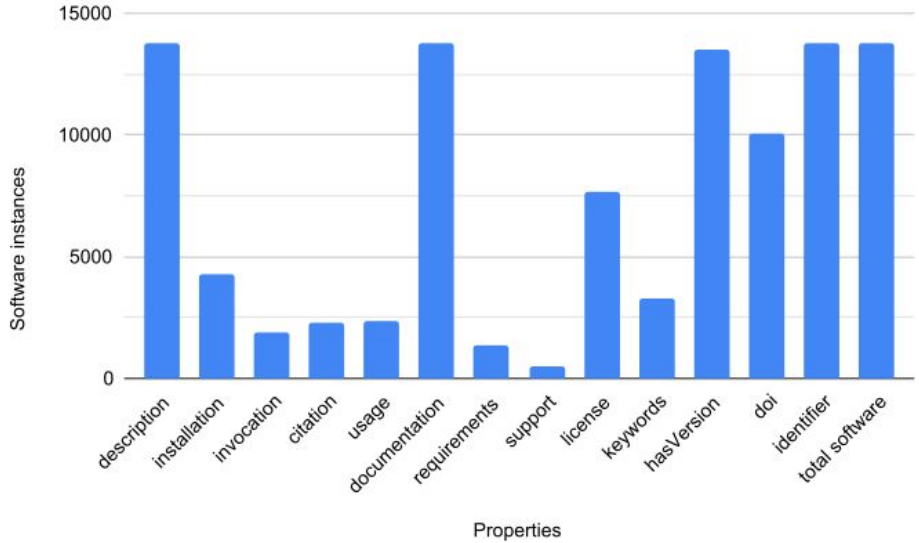
Citation

```
@article{arenas2022morph,
  title   = {{Morph-KGC: Scalable Knowledge Graph Materialization with
  author  = {Arenas-Guerrero, Julián and Chaves-Fraga, David and Toledo
  journal = {Semantic Web},
  year    = {2022},
  url     = {http://www.semantic-web-journal.net/system/files/swj3135.p
}
```

Extracting KGs from thousands of Open Source repositories

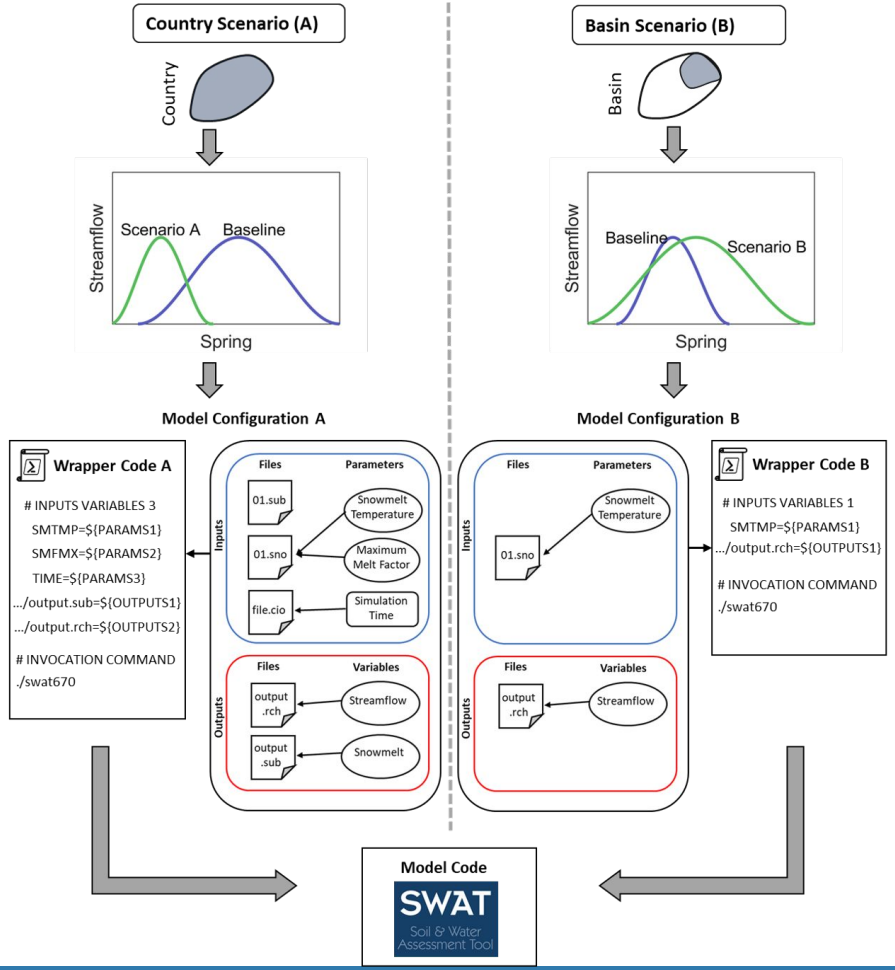
- Zenodo software (> 12000)
- Measuring best practices based on metadata





1. Representing Research Software metadata
2. Knowledge capture from documentation and code
3. **Automated encapsulation for reusability**

Encapsulating configured experiments



Capturing different configurations of a complex tool **requires significant knowledge.**

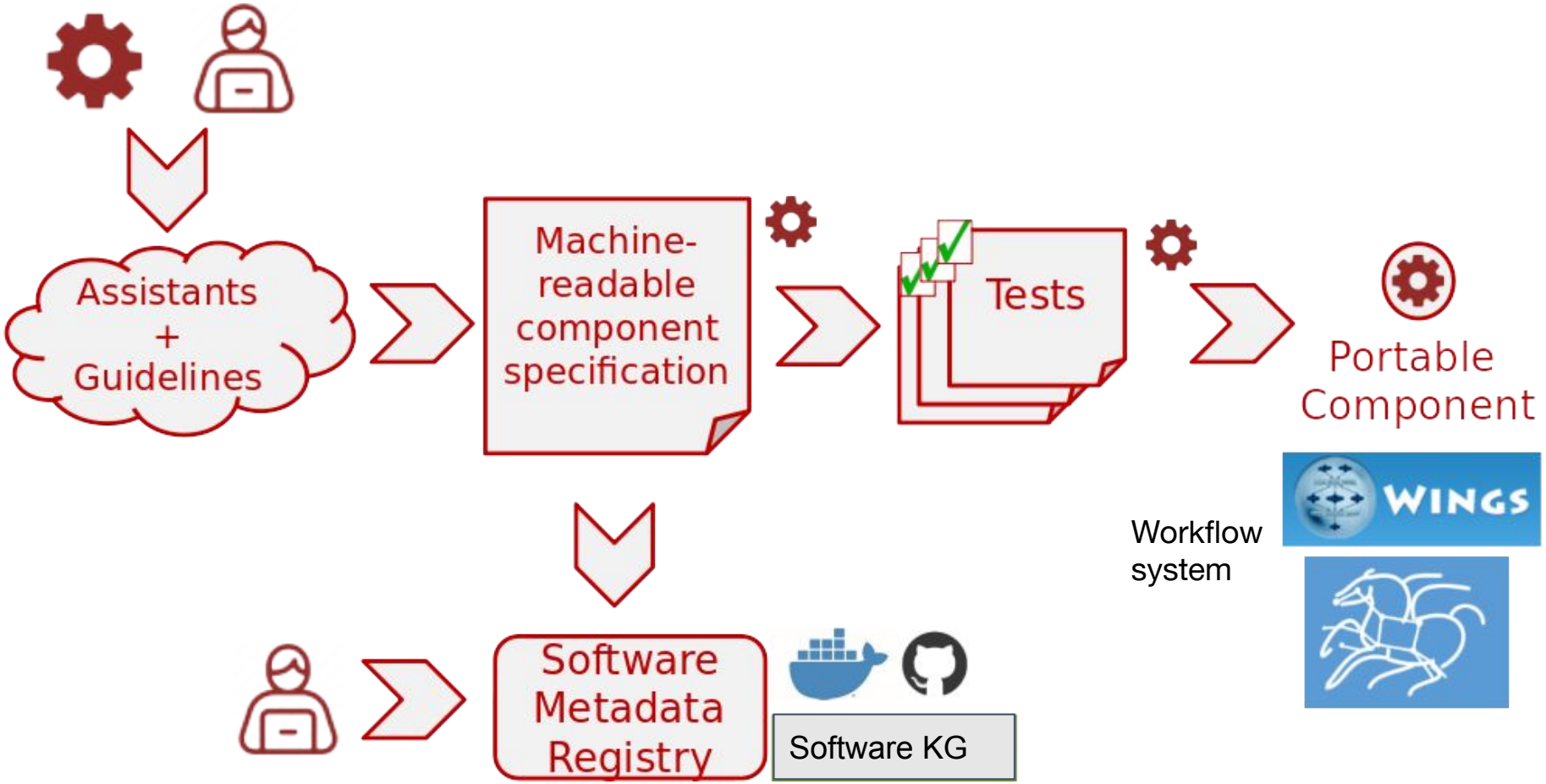
How can we ensure an expert can share configured/calibrated models?

- Reduce complexity for novice users

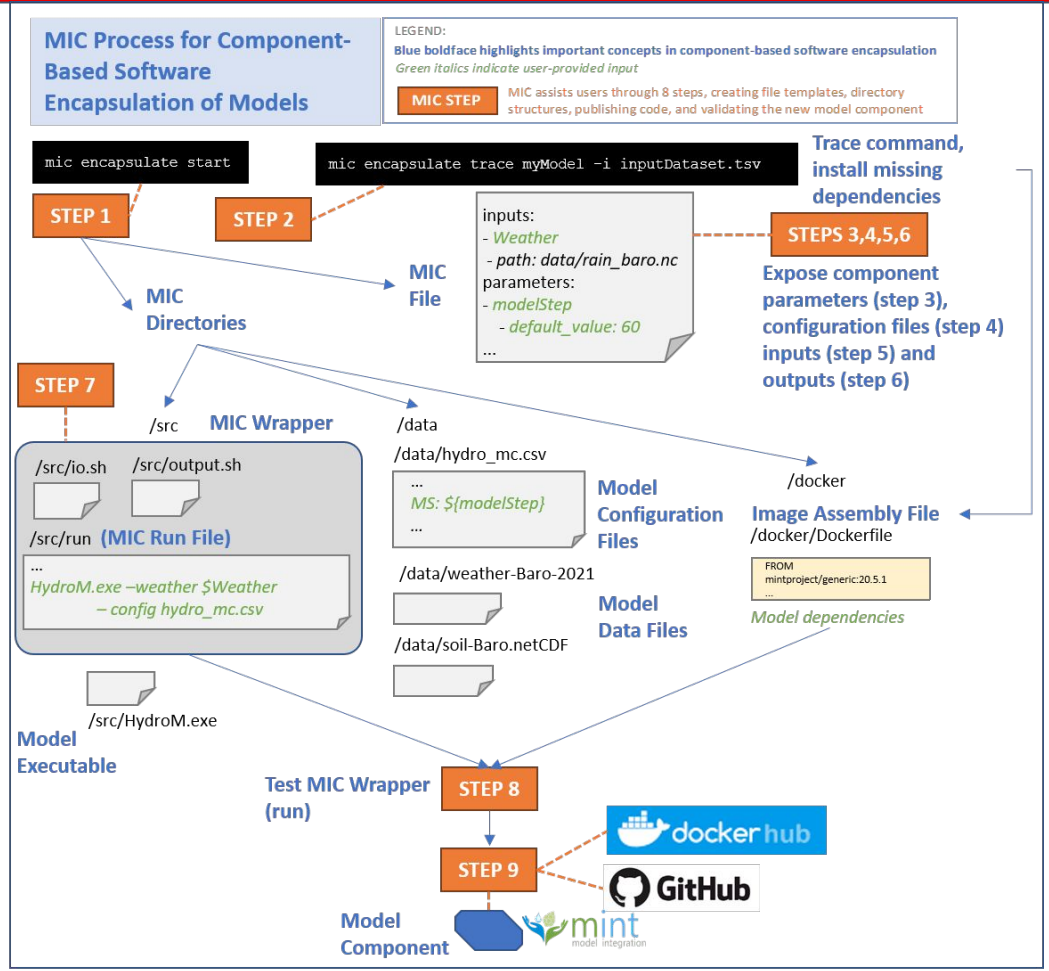
Example: SWAT Hydrology model

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

Scientific Software encapsulation



Reusing software: model encapsulation methodology



Software 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

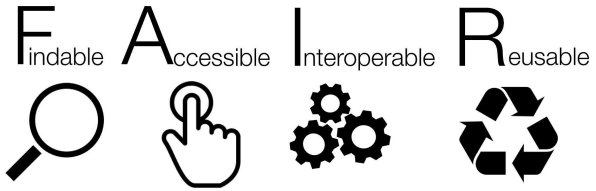
Summing up

	Describe	Compare	Reuse
Representing Research Software metadata			
Knowledge capture from documentation and code			
Automated encapsulation for reusability			

Research software is a critical asset for **Open Science**

Accelerating Software Understanding requires:

- Automated description
- Assisted comparison
- Easy reuse



Acknowledgements





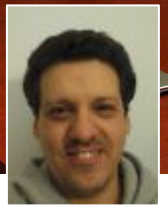






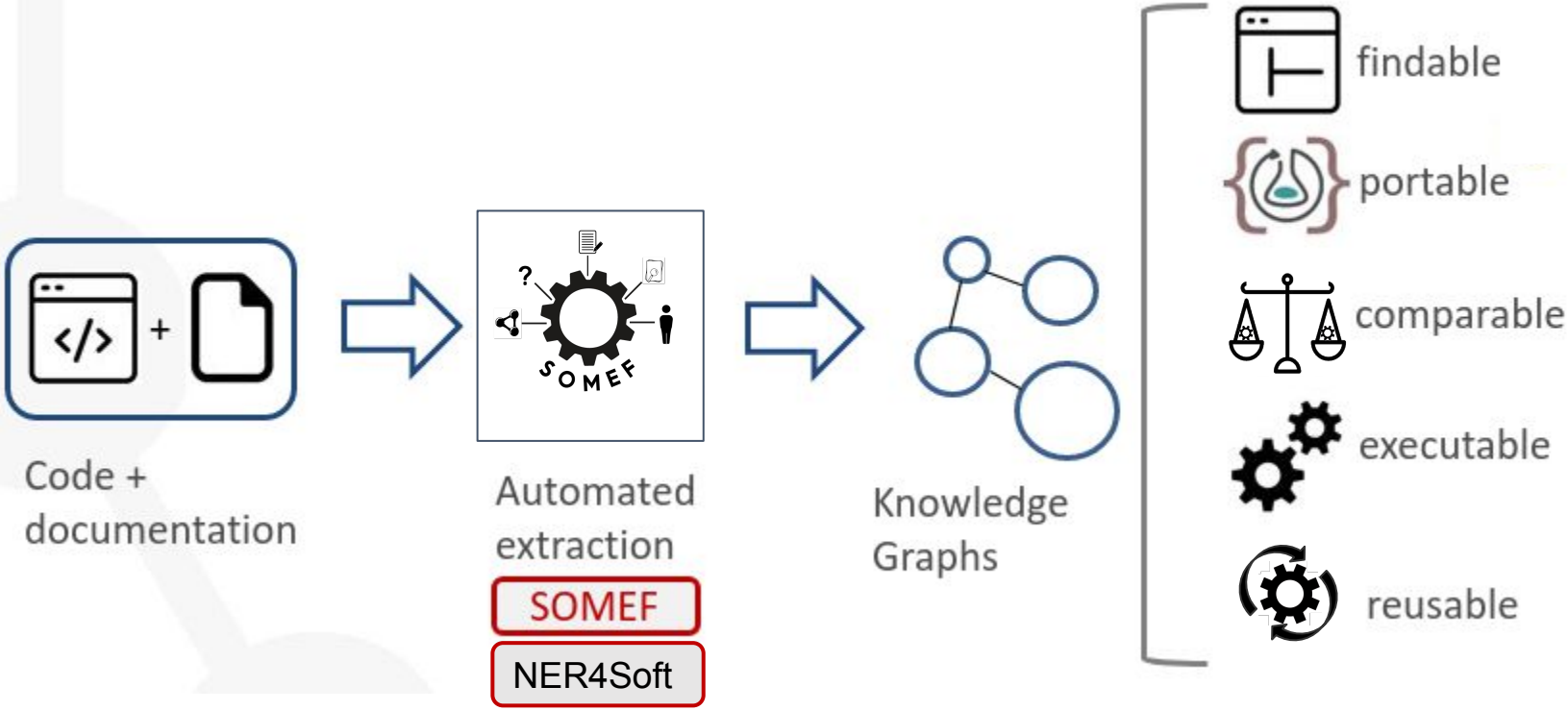





Thanks to Yolanda Gil, Varun Ratnakar, Maximiliano Osorio, Hernán Vargas, Deborah Khider, Allen Mao, Aidan Kelley, Haripriya Dharmala, Jiajing Wang, Rosa Filgueira, Pablo Calleja, Oscar Corcho, Laura Camacho, Jhon Toledo, Miguel Angel García, Esteban Gonzalez, Elena Montiel, Elvira Amador & all the students at UPM and USC who participated in the initiatives mentioned in this presentation

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 to promote Open Science!

Extra slides

Example Output

CodeMeta


```
description:
  0:
    excerpt:      "WIDOCO helps you to publish and create an enriched and customized documentation of you
                  classes, properties and data properties of the ontology, the OOPS! webservice by María
                  being used. In addition, we use WebVowl to visualize the ontology and have extended Bub
                  documentation of the terms in your ontology (based on [LODE](http://www.essepuntato.it/
                  annotation in JSON-LD snippets of the html produced.\n* Association of a provenance pag
                  means to complete it on the fly when generating your ontology. Check the [best practice
                  WIDOCO.\n* Guidelines on the main sections that your document should have and how to co
                  changelog of differences between the actual and the previous version of the ontology (b
                  them independently and replace only those needed.\n* Content negotiation and serializat
    confidence:
      0:          1
    technique:    "wordnet"
  1:
    excerpt:      "For a complete list of the current improvements and next features, check the project o
    confidence:
      0:          0.8231493588525339
    technique:    "classifier"
  2:
    excerpt:      "Wizard for documenting ontologies. WIDOCO is a step by step generator of HTML template
    confidence:
      0:          1
    technique:    "metadata"
citation:
  0:
    excerpt:      "@inproceedings{garijo2017widoco,\n  title={WIDOCO: a wizard for documenting ontologies
                  organization={Springer, Cham},\n  doi = {10.1007/978-3-319-68204-4_9},\n  funding = {US
    confidence:
      0:          1
    technique:    "classifier"
```

```
@context:       "https://doi.org/10.5063/schema/codemeta-2.0"
@type:          "SoftwareSourceCode"
license:        "https://raw.githubusercontent.com/dgarijo/Widoco/master/LICENSE"
codeRepository: "git+https://github.com/dgarijo/Widoco.git"
dateCreated:    "2013-07-15"
datePublished:  "2020-12-14"
dateModified:   "2021-03-16"
downloadUrl:    "https://github.com/dgarijo/Widoco/releases"
issueTracker:   "https://github.com/dgarijo/Widoco/issues"
name:           "Widoco"
version:        "v1.4.15_1"
description:
  0:            "Wizard for documenting ontologies. WIDOCO is a step by step generat
  1:            "WIDOCO helps you to publish and create an enriched and customized d
                the classes, properties and data properties of the ontology, the OOP
                URI and title being used. In addition, we use WebVowl to visualize t
                WIDOCO:\n* Automatic documentation of the terms in your ontology (ba
                /index.html))\n* Automatic annotation in JSON-LD snippets of the htm
                extraction from the ontology plus the means to complete it on the fl
                to know more about the terms recognized by WIDOCO.\n* Guidelines on
                (http://vowl.visualdataweb.org/webvowl/)).\n* Automatic changelog of
                /)).\n* Separation of the sections of your html page so you can writ
                practices\n\n"
  2:            "For a complete list of the current improvements and next features,
releaseNotes:   "This pre-release fixes issues regarding namespace prefixes (now the
                settings in your visualization)\r\n\r\nMore information on the addre
keywords:
  0:            "ontology"
  1:            "wizard"
  2:            "metadata"
  3:            "documentation"
  4:            "ontology-diagram"
  5:            "ontology-evaluation"
```