FROM SOFTWARE METADATA REGISTRIES TO KNOWLEDGE GRAPHS: ONTOSOFT AND OKG-SOFT

Daniel Garijo, Maximiliano Osorio, Deborah Khider,
Varun Ratnakar and Yolanda Gil

University of Southern California,
Information Sciences Institute

@dgarijov

Scientific Software Registry Collaboration Workshop (SSRCW)
November, 13th 2019

The importance of Scientific Software

- Software helps understand data
  - Provenance, reproducibility
- Software helps understanding methods
  - Assumptions, limitations

Open data

figshare

DRYAD

The Dataverse Network™ Project

Open source software

github SOCIAL CODE HOSTING

Apache

sourceforge

Open publications

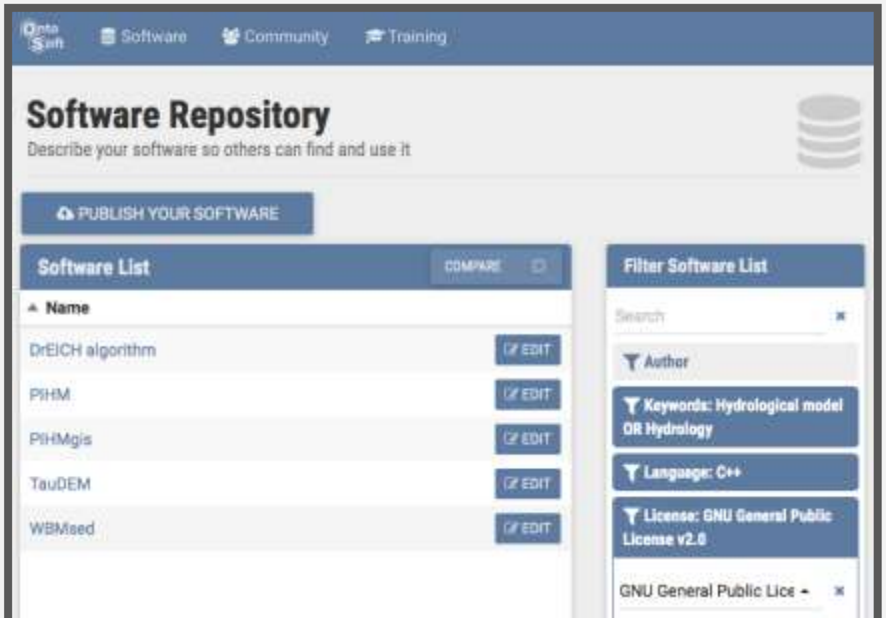
PLOS

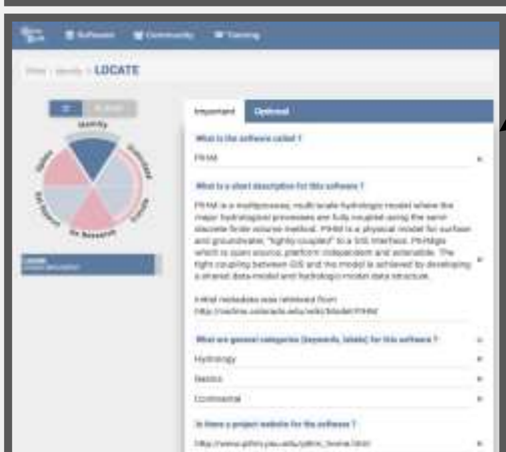
Policy Guidelines for the Development and Promotion of OPEN ACCESS

wikigenes collaborative publishing

Software registries help search, access and understand Scientific Software.

Prior Work: OntoSoft Software Metadata Registry





Finding Software

OntoSoft

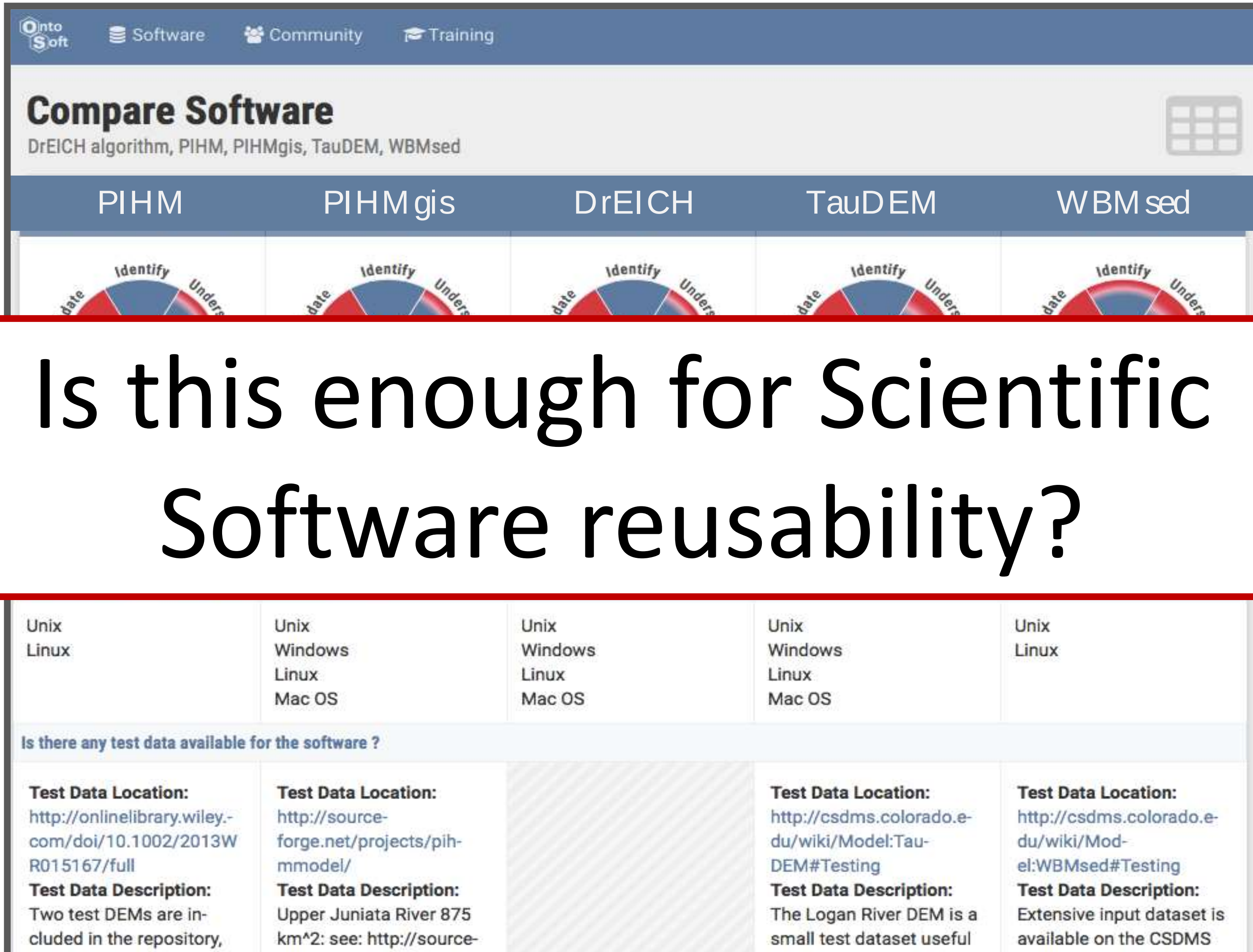
Distributed Software Metadata Registry

- 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

[Gil et al 2015]: OntoSoft: Capturing Scientific Software Metadata Eighth ACM International Conference on Knowledge Capture, Palisades, NY, 2015

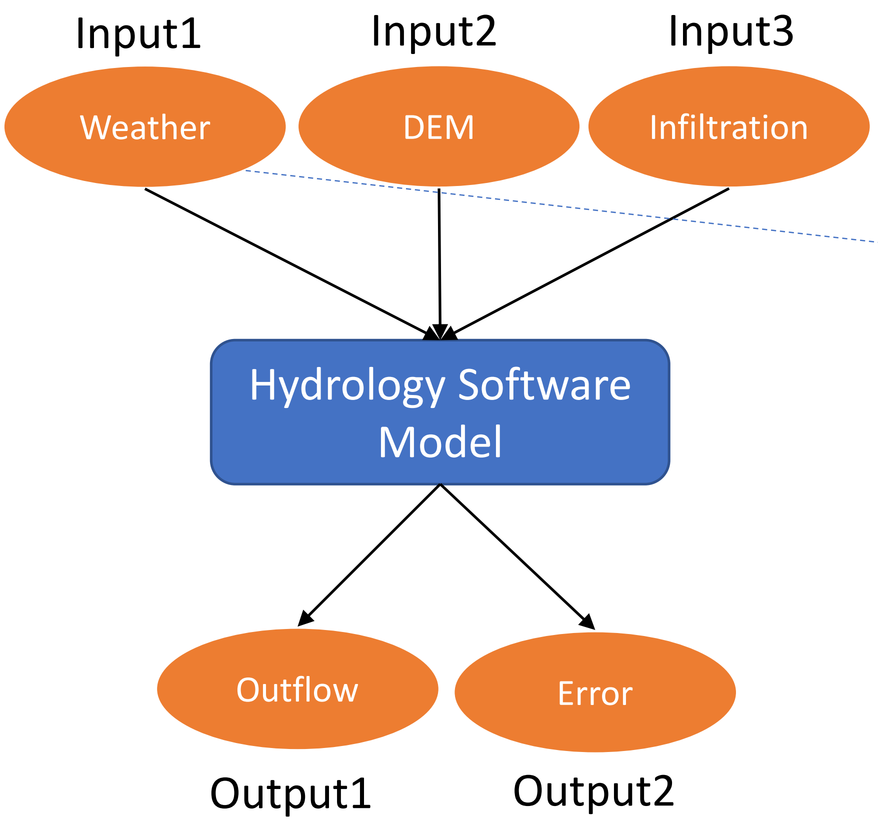
Prior Work: OntoSoft Software Metadata Registry

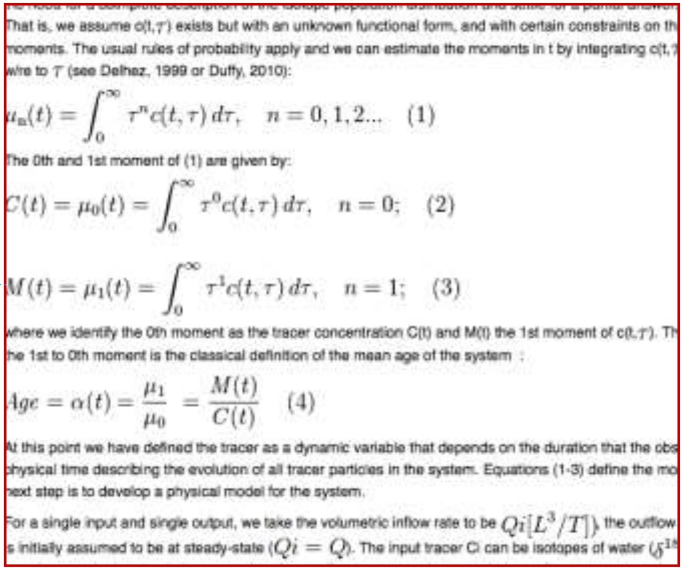


Is this enough for Scientific Software reusability?

Requirements for Software Reusability

1. Exposing software inputs, outputs and their corresponding variables





That is, we assume $c(t,\tau)$ exists but with an unknown functional form, and with certain constraints on th
moments. The usual rules of probability apply and we can estimate the moments in t by integrating $c(t,\tau$
wrt to τ (see Delhez, 1999 or Duffy, 2010):

$$\mu_n(t) = \int_0^\infty \tau^n c(t,\tau)\,d\tau, \quad n = 0,1,2... \quad (1)$$

The 0th and 1st moment of (1) are given by:

$$C(t) = \mu_0(t) = \int_0^\infty \tau^0 c(t,\tau)\,d\tau, \quad n = 0; \quad (2)$$

$$M(t) = \mu_1(t) = \int_0^\infty \tau^1 c(t,\tau)\,d\tau, \quad n = 1; \quad (3)$$

where we identify the 0th moment as the tracer concentration C(t) and M(t) the 1st moment of $c(t,\tau)$. Th
he 1st to 0th moment is the classical definition of the mean age of the system :

$$Age = \alpha(t) = \frac{\mu_1}{\mu_0} = \frac{M(t)}{C(t)} \quad (4)$$

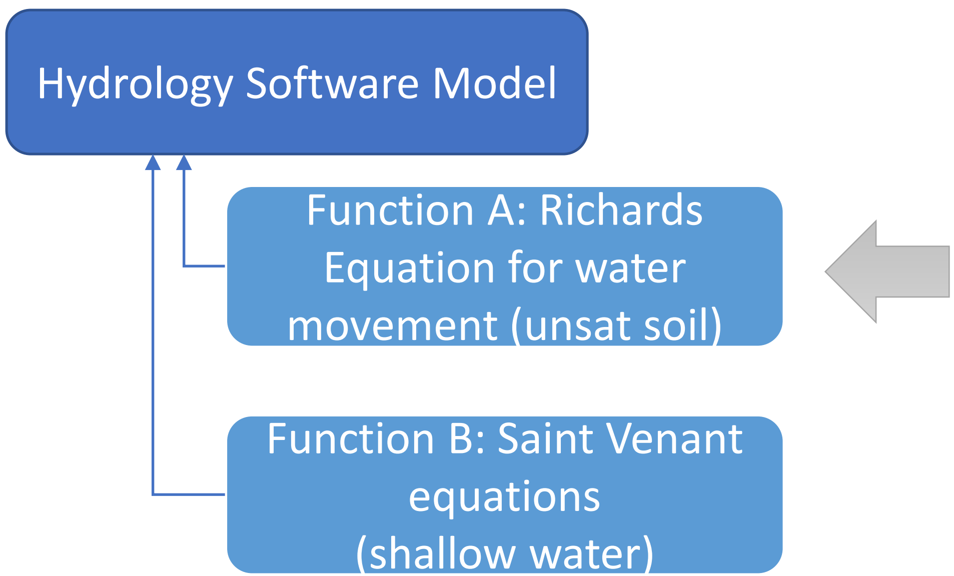
At this point we have defined the tracer as a dynamic variable that depends on the duration that the obs
physical time describing the evolution of all tracer particles in the system. Equations (1-3) define the mo
next step is to develop a physical model for the system.

For a single input and single output, we take the volumetric inflow rate to be $Qi[L^3/T]$, the outflow
s initially assumed to be at steady-state ($Qi = Q$). The input tracer Ci can be isotopes of water (δ^{18}

- Land surface temperature (degC)
- Precipitation rate (mm/h)
- Land surface wind speed (m/day)
- Net radiation (MJ/(day m^2))

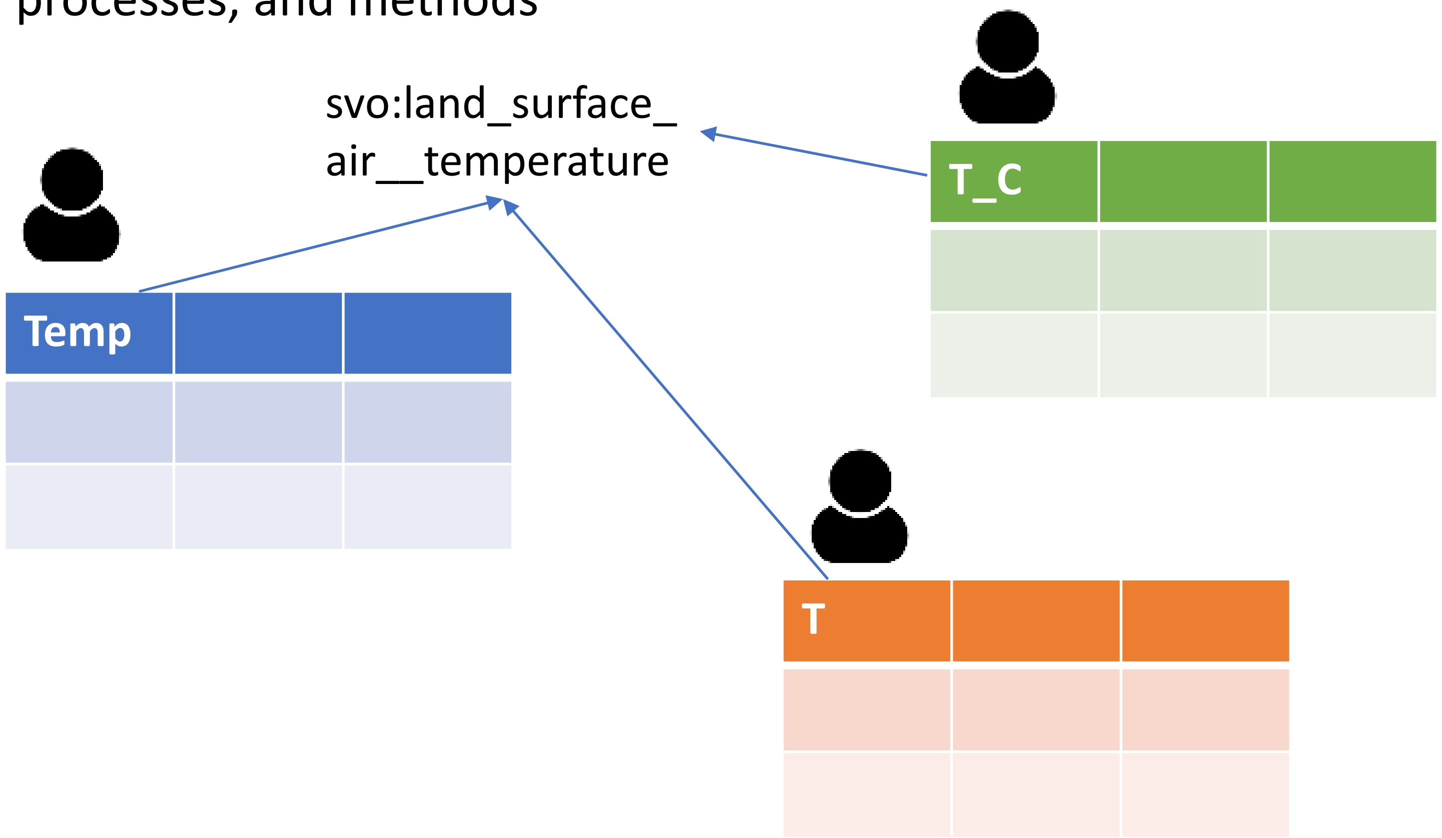
Requirements for Software Reusability

1. Exposing software inputs, outputs and their corresponding variables
2. Capturing the functions of the software component being used



Requirements for Software Reusability

1. Exposing software inputs, outputs and their corresponding variables
2. Capturing the functions of the software component being used
3. Using principled ontologies with structured names for model variables, processes, and methods

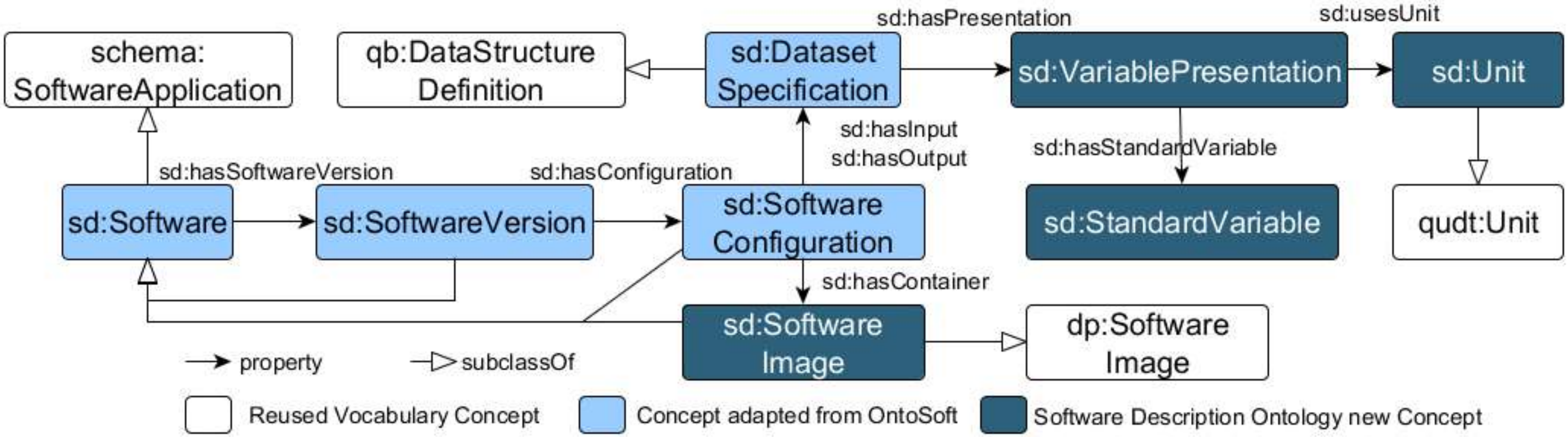


Requirements for Software Reusability

1. Exposing software inputs, outputs and their corresponding variables
2. Capturing the functions of the software component being used
3. Using principled ontologies with structured names for model variables, processes, and methods
4. Capture the semantic structure of software invocations

Dependencies?
Sample runs?
Invocation command?
Is data supposed to be in the same folder?
Default arguments/Configuration files?
Volumes?
Do I have to log in in the image

Evolving OntoSoft: Software Description Ontology



Extensions:

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

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

OKG-SOFT: Framework

Software Model Catalog contains:

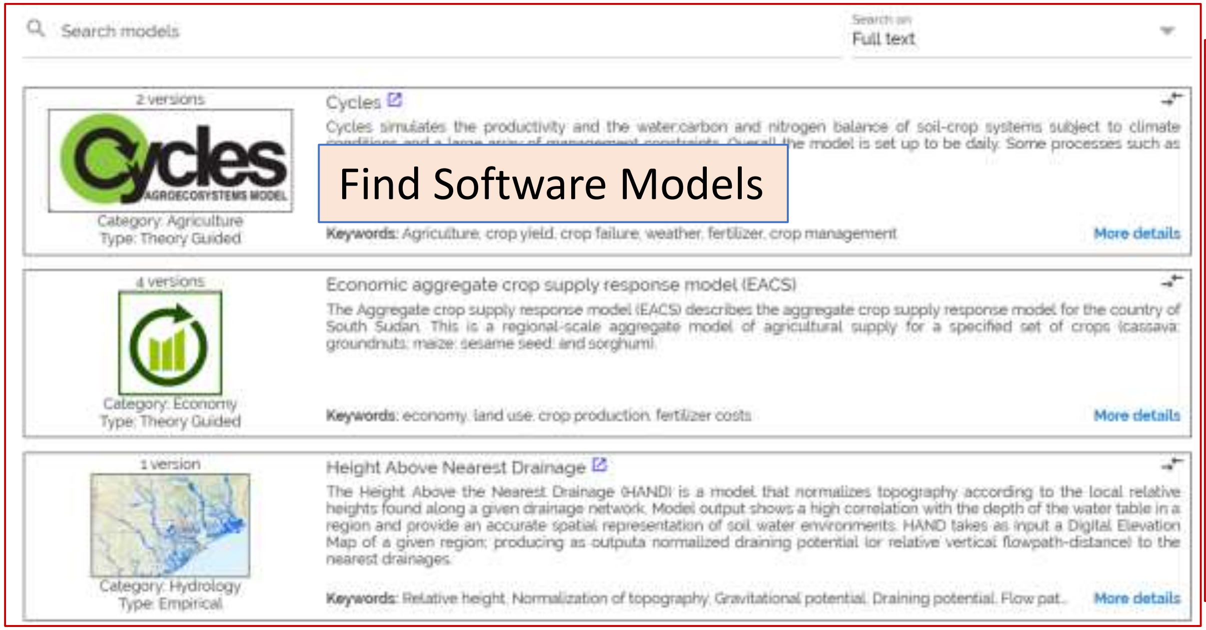
- Models from hydrology, agriculture and economy, their versions and model configurations.
  - More than 200 variables mapped to SVO.
  - All models are executable through scientific workflows
  - Most contents are added manually (expert users) **collaboratively**

- Automated unit transformations
- Automated software image description
- Semi-automated Wikidata linking

APIs:

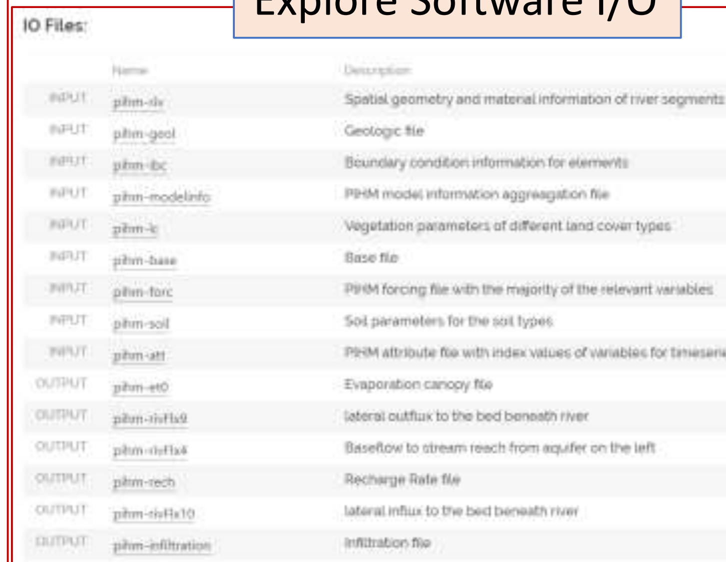
- SPARQL endpoint
- REST APIs (GET/POST) https://query.mint.isi.edu/api/mintproject/MINT-ModelCatalogQueries#/
- Python clients

Exploitation: Exploring Scientific Software Model Metadata



Find Software Models

Explore Software I/O



pihm-riv Spatial geometry and material information of river segments

Explore variables

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

Compare models

http://models.mint.isi.edu

Summary

Scientific Software is crucial to understand

- Existing data
- Published methods

Scientific Software Metadata registries help search and understand software

- Enough for software reusability?

Requirements for scientific software reusability:

- Describing inputs, outputs, variables and software invocation details

Our approach for capturing and structuring scientific software