



Dziedzinowo zorientowane
usługi i zasoby infrastruktury
PL-Grid dla wspomagania
Polskiej Nauki w Europejskiej
Przestrzeni Badawczej

Lightweight Metadata and Data Management with DataNet

Daniel Hareźlak¹, Marek Kasztelnik¹, Maciej Pawlik¹,
Bartosz Wilk¹, Marian Bubak^{1,2}

¹ACC Cyfronet AGH

²AGH University of Science and Technology, Institute of Computer
Science AGH

Konferencja Użytkowników Komputerów Dużej Mocy, March 12-14, 2014, Zakopane, Poland



- Motivation behind DataNet
- Metadata Management Requirements
- Architecture Description
- PL-Grid Deployment
- Conclusions

■ Rationale

- Current data **discoverability** and **reproducibility** is poor
- Data management as a common requirement in computational sciences
- Workflow and scripting engines provide only little support (GridSpace, Taverna)
- Each application is different and requires a dedicated metadata/data model

■ Objectives

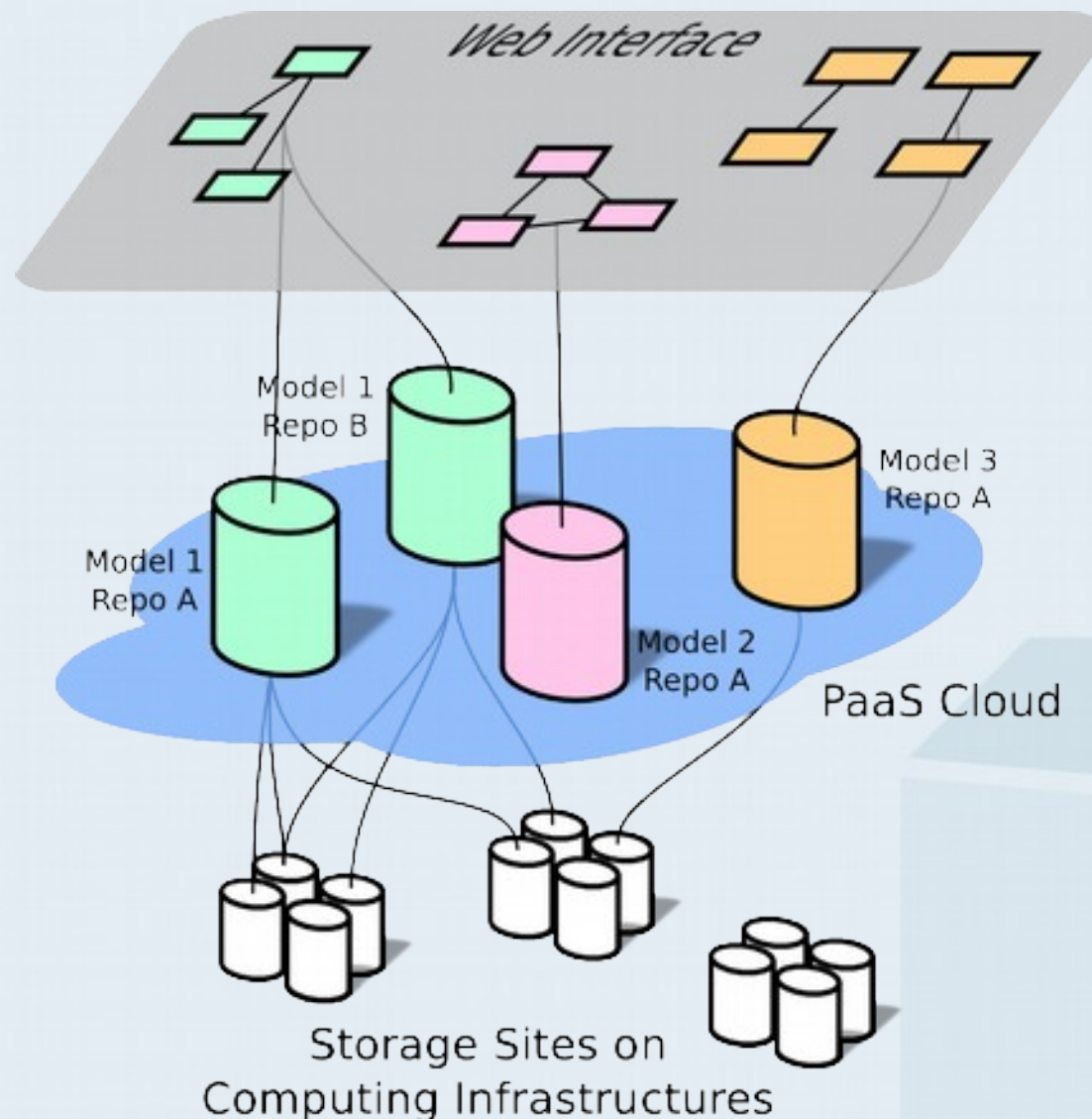
- Provide means for **ad-hoc metadata model creation** and deployment of corresponding storage facilities
- Create a research space for **metadata model exchange and discovery** with associated data repositories with access restrictions in place
- Support **different types of storage sites** and **data transfer protocols**
- Support the exploratory paradigm by making the models evolve together with data

- PLGrid infrastructure – supporting different e-Science domains
 - Various applications coming from different scientific communities
 - Common computational and storage resources

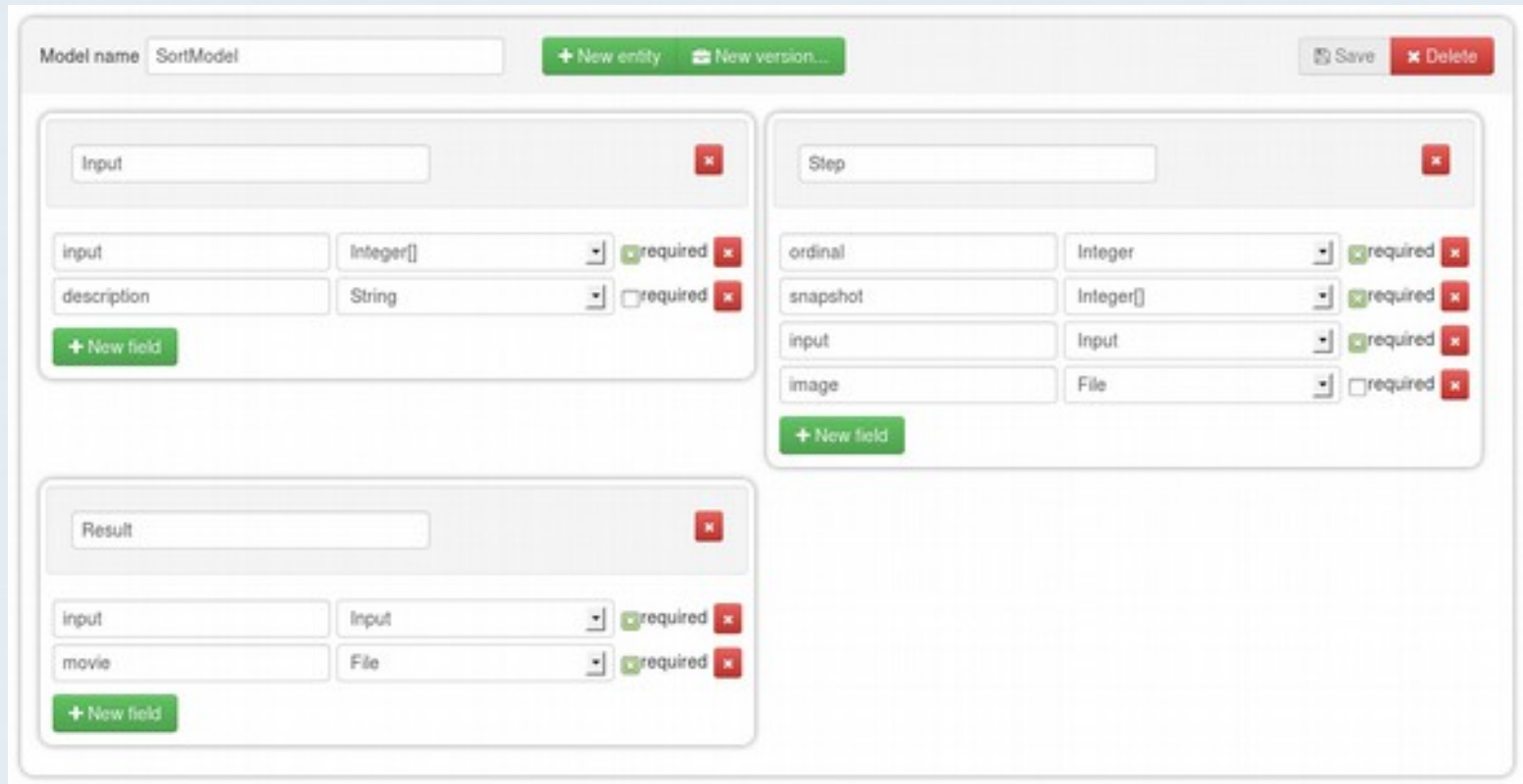
- Deployment of model data as repositories in PL-Grid cloud
 - Robust enablement of a dedicated interface
 - Access control capabilities
 - Exploitation of available storage infrastructure

- Universal availability of the repository
 - Platform independent
 - Facilitated by existing standards

- **Web Interface** is used by users to create, extend and discover metadata models
- Model repositories are deployed in the **PaaS Cloud** layer for scalable and reliable access from computing nodes through REST interfaces
- Data items from **Storage Sites** are linked from the model repositories



- Set of entities with fields
 - Simple types
 - Array types
 - File type
 - Relations



The screenshot displays the DataNet Data Model editor interface for a model named "SortModel". The interface includes a header with the model name, a "New entity" button, a "New version..." button, and "Save" and "Delete" buttons. The main area is divided into three panels, each representing an entity with its fields and data types.

Entity: Input

Field Name	Data Type	Required
input	Integer[]	checked
description	String	unchecked

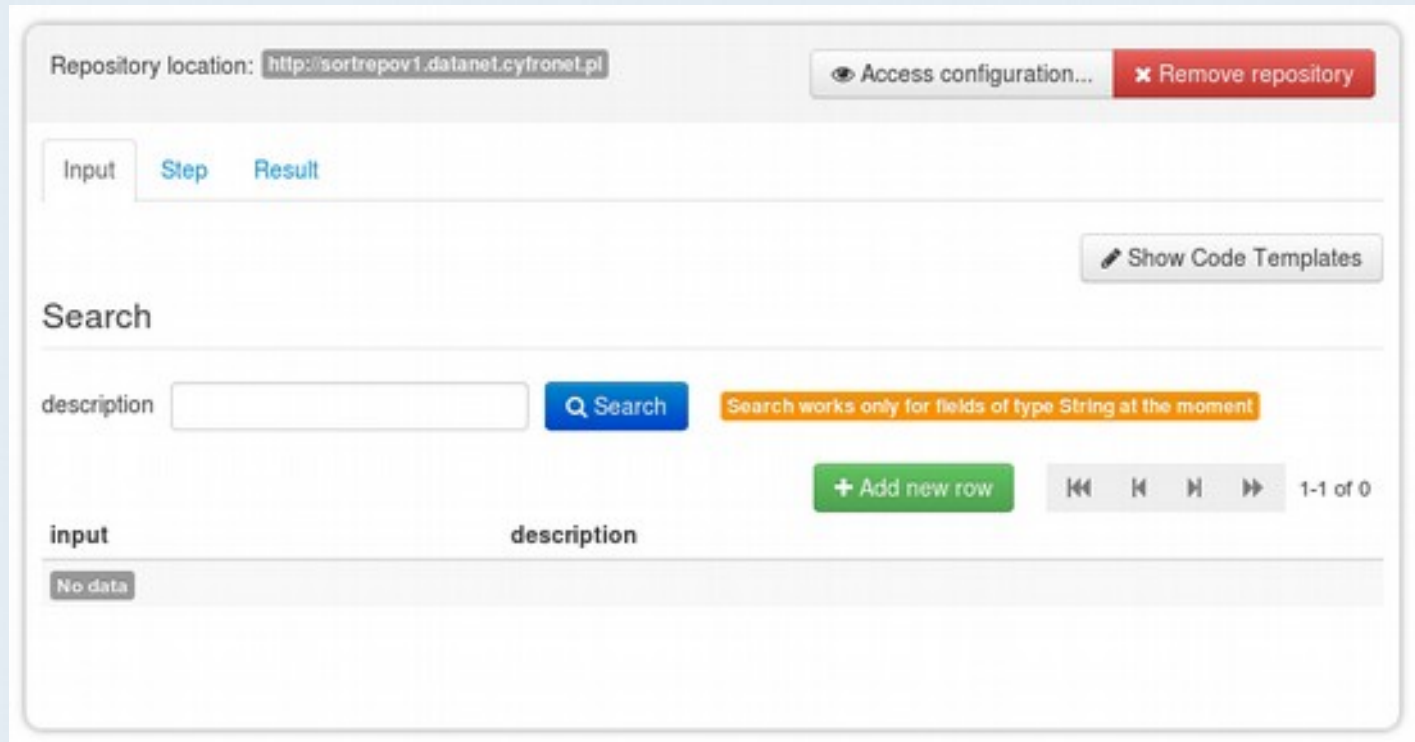
Entity: Step

Field Name	Data Type	Required
ordinal	Integer	checked
snapshot	Integer[]	checked
input	Input	checked
image	File	unchecked

Entity: Result

Field Name	Data Type	Required
input	Input	checked
movie	File	checked

- Repositories are accessed through REST
 - Data view through a web application
 - Configurable Access control
 - Public
 - Private (within a group of users)



Repository location: [Access configuration...](#) [Remove repository](#)

[Input](#) [Step](#) [Result](#)

[Show Code Templates](#)

Search

description [Search](#) Search works only for fields of type String at the moment

[+ Add new row](#) [⏪](#) [⏴](#) [⏵](#) [⏩](#) 1-1 of 0

input	description
No data	

■ Data sent over with JSON or FORM

- REST methods
 - POST – submit new data
 - PUT – modify data
 - DELETE – remove data
 - GET- retrieve data
 - Queries with URL

```
require 'rest-client'
require 'json'

datanet=RestClient::Resource.new('http://a:a@repo.datanet.cyfronet.pl')
datanet.get
def get_user(first_name, last_name)
  {first_name: first_name, last_name: last_name}.to_json
end
get_user "marek", "kasztelnik"
datanet['user'].post get_user("Marek", "Kasztelnik")
datanet["user"].get
10.times {datanet['user'].post get_user("Marek", "Kasztelnik")}
datanet["user"].get
datanet["user/519dbfed2fbb0c79f400000b"].delete
datanet["user"].get
```

```
import requests as req
import json

headers = {'content-type': 'application/json'}
resp = req.post('http://test5.datanet.cyfronet.pl/Hello',
               data = json.dumps({'name': 'hello1'}), auth = ('', ''), headers = headers)
```


■ PLGrid Users

- Access with regular **PLGrid account**
- **REST** interface
- **Web Application** for simple use cases

■ Domain application interoperability

- User **proxy delegation** retrieved from PLGrid OpenID provider
- Access from browsers possible through **CORS** (direct uploads)

■ DataNet as a Service is in preparation

- Security audit in progress

■ DONEs

- Custom CloudFoundry environment was setup as a PaaS platform to ensure quick deployments of required application and storage services
- Schema for metadata model creation was elaborated and was evaluated for NoSQL storage service MongoDB
- Storage site access libraries were implemented and tested
- Deployment of a web-based tool to create, discover and manage metadata models
- Integrated storage site access libraries with PLGrid OpenID proxy extension

■ TODOs

- Support various types of metadata storage services to fulfil different application requirements
- Prototype a utility for data migration between model versions

Thank You!



■ Acknowledgements

- This research has been partially supported by the European Regional Development Fund program no. POIG.02.03.00-00-096/10 as part of the PL-Grid PLUS project

■ Contact us and help make DataNet better

■ Visit <http://dice.cyfronet.pl> for more information

■ See DataNet in action at <https://datanet.cyfronet.pl> (PLGrid account required)

