The SPARQLing Unicorn

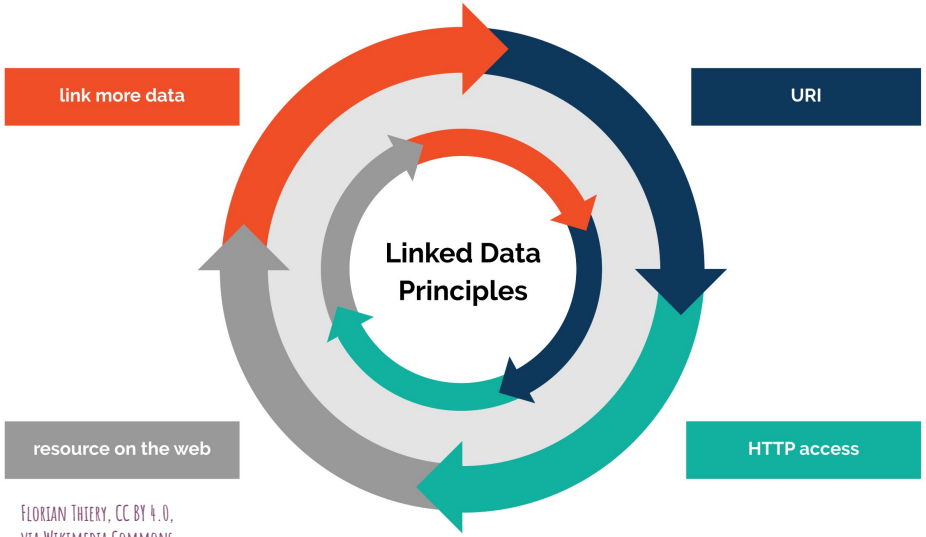
A Research Tool for Linked Open Data in QGIS and git-action-based ontology documentation

Timo Homburg & Florian Thiery

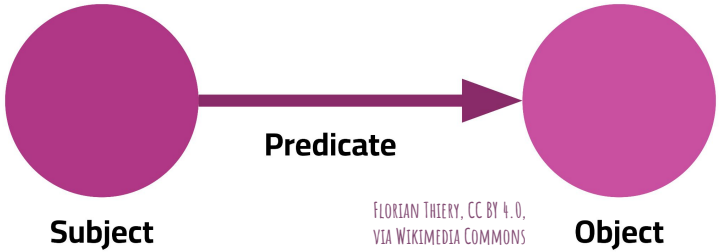
deRSE24 - Conference for RSEng in Germany
JMU Würzburg | 05. - 07. March 2024
Session: Metadata for Research Software

Introduction: Research Data

- At the end of many research projects stands the publication of research:
  - Research papers
  - **Data publications**
  - Software publications
- The publication of research data comes with many questions:
  - Which data formats?
  - Accessibility of data?
  - Long term storage and hosting?
  - Long term data provision using APIs?
  - Which metadata?
  - FAIR data?
  - Where to publish data?
  - How to generate views on data?
- How can we make research data long term usable with the least possible maintenance?



Florian Thiery, CC BY 4.0, via Wikimedia Commons



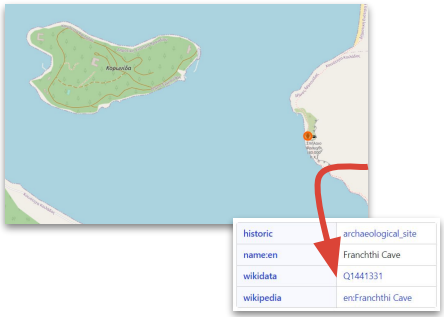
Florian Thiery, CC BY 4.0, via Wikimedia Commons

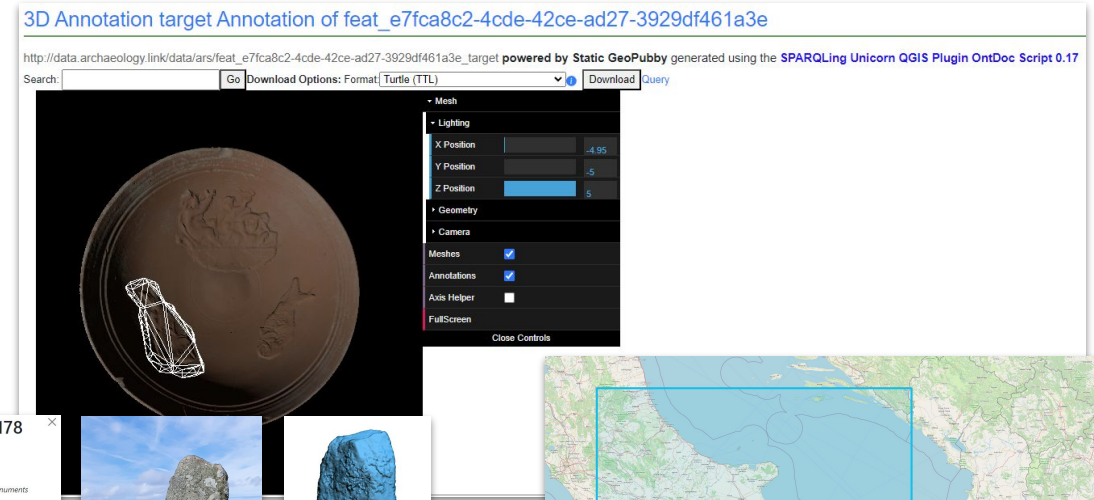
For the linking of data and FAIRification, the Linked Open Data (LOD) is the method of choice.

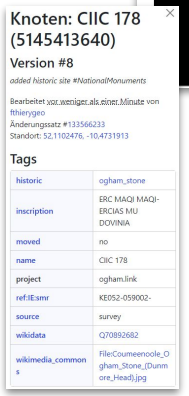
Publishing Research Data as LOD

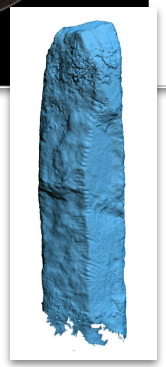
Introduction: Research Data as LOD

- Research projects typically yield a variety of different research data as their results
- Data come in different formats and fulfil different functions
- Research data points to further research data
- To capture these relations, people tend to publish research data as linked open data

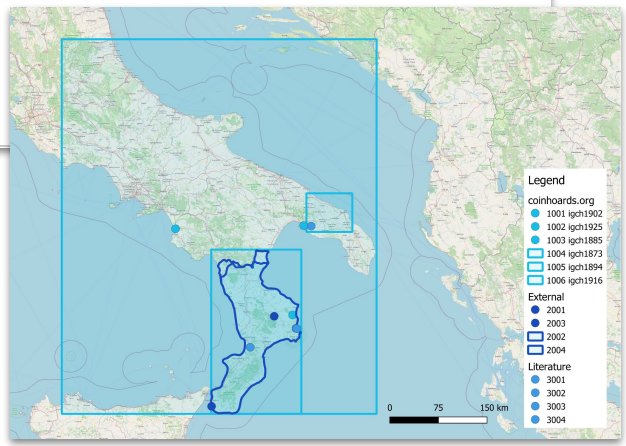


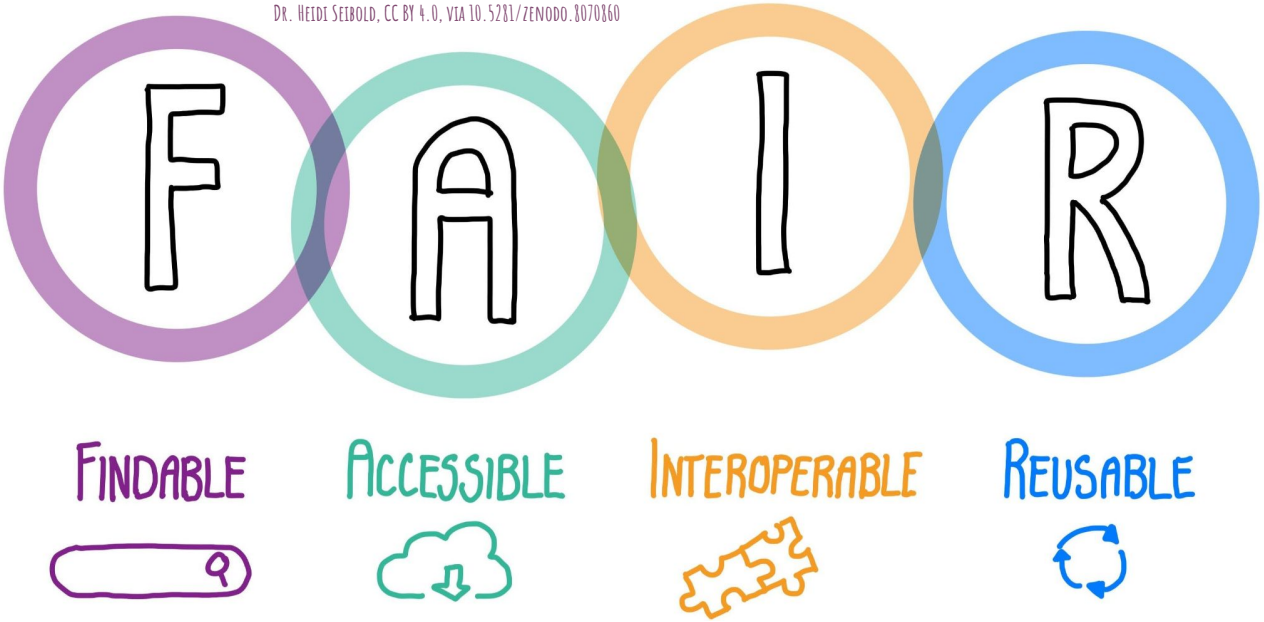






↑ Ogham in 3D Project, DIAS, CC BY-NC-SA 3.0 Ireland

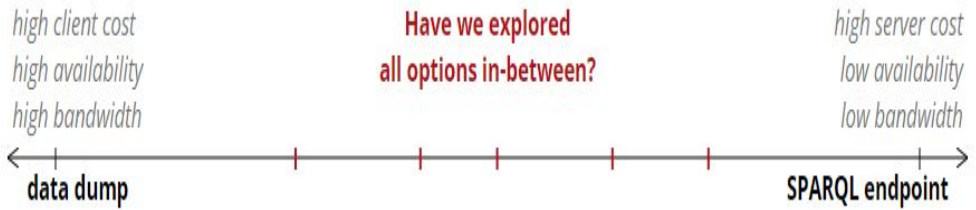




In order to create FAIR, reproducible and understandable data for re-use, LOD must be documented and Published.

Publishing Linked Open Data

- When publishing Linked Open Data, we have several expectations:
  - 5-star principles of Linked Open Data
  - FAIR publication
  - Open license
  - Reusing common vocabularies
  - Long term storage of the dataset, e.g. with a DOI
- Publish vocabularies independently of data
- Different ways to publish Linked Open Data:
  - Data Dumps
  - Linked Data Fragments
  - SPARQL Endpoint
- Assumptions for this talk:
  - No server resources exist after the research project is over
  - The research institute cannot provide long term hosting
  - The National Data Infrastructure is
    - Not in place
    - Not suitable for hosting the data
    - Not willing to accept the data



Linked Data Dumps

- Longterm hosting
- Sustainable
- Not easily discoverable by many research communities
- Experts knowledge is required to
- Linked Data Dumps are not necessarily Linked Open Usable Data
- How can we improve this situation?

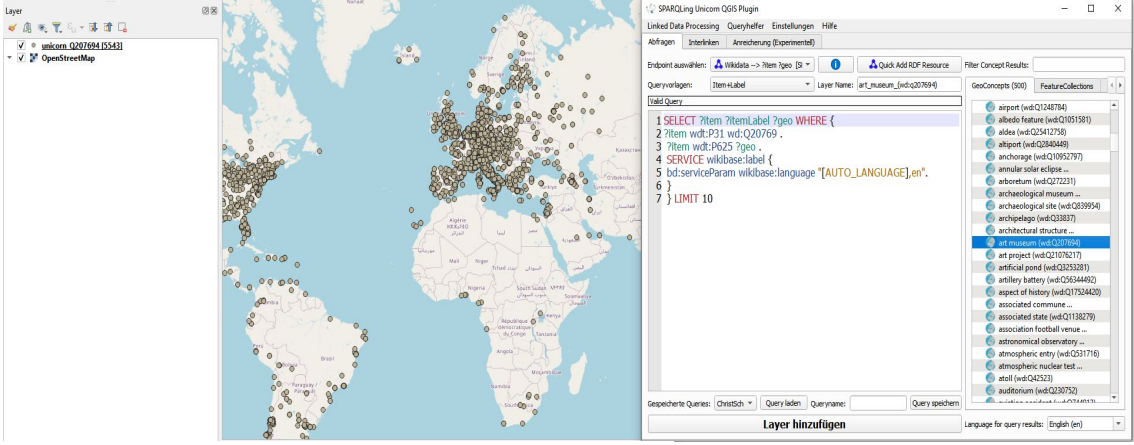
Motivation:

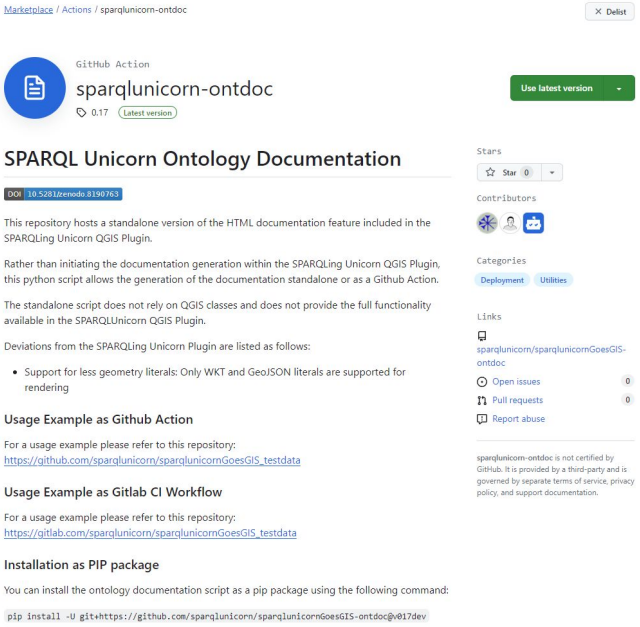
- Create Linked Open Usable Data (LOUD) Dumps
- What can we achieve in hosting data alone?
- How can we improve Linked Data Dump publications to be more user-friendly?

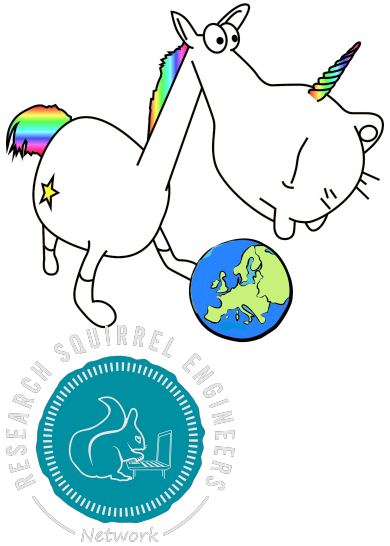
SPARQL Unicorn Ontology Documentation Tool

Documentation Tool

- An extension of the SPARQLing Unicorn QGIS Plugin
- Idea: Convert an RDF Dump to an enriched HTML Deployment
- The result should be ready to host on platforms such as Github and Gitlab Pages
- The Documentation Process should be usable as Continuous Integration components
- The Deployment should be in such a way that it is useful for a maximum of research communities

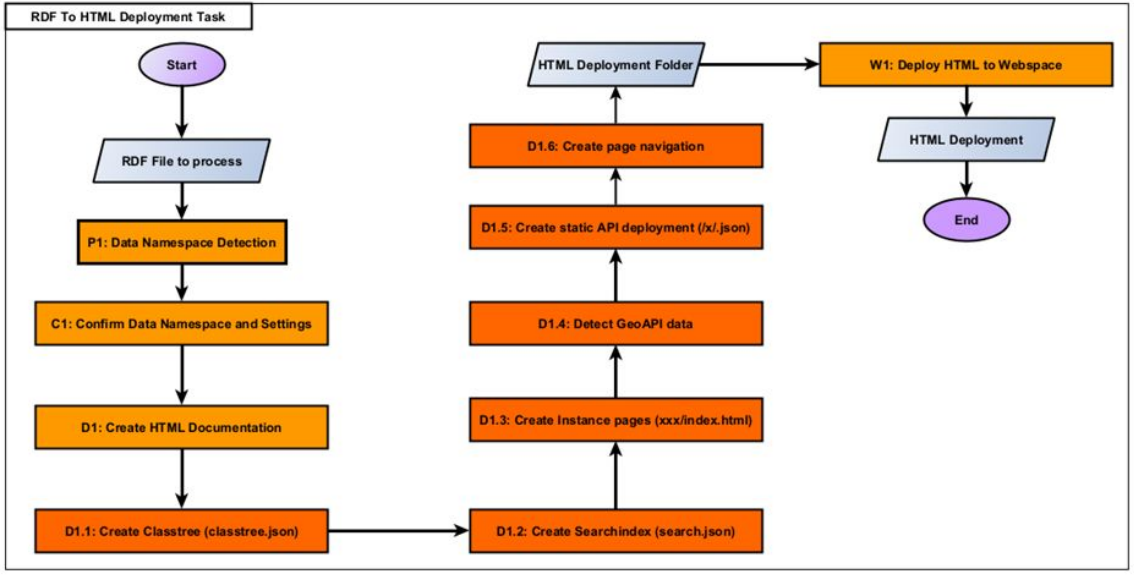


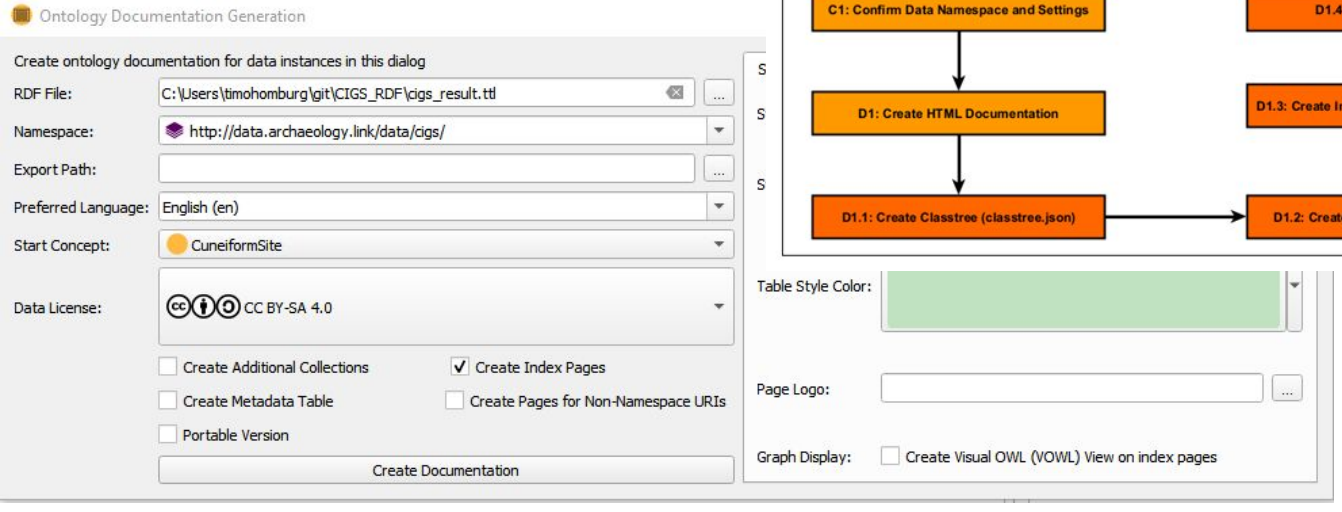




Documentation Tool: How to Use it

1. Load an RDF Data Dump File
2. Select one or more data namespaces which should be documented as HTML
3. Select a target folder and customize your export
4. Start HTML Generation to a local folder
5. Optional Deployment to a webspace

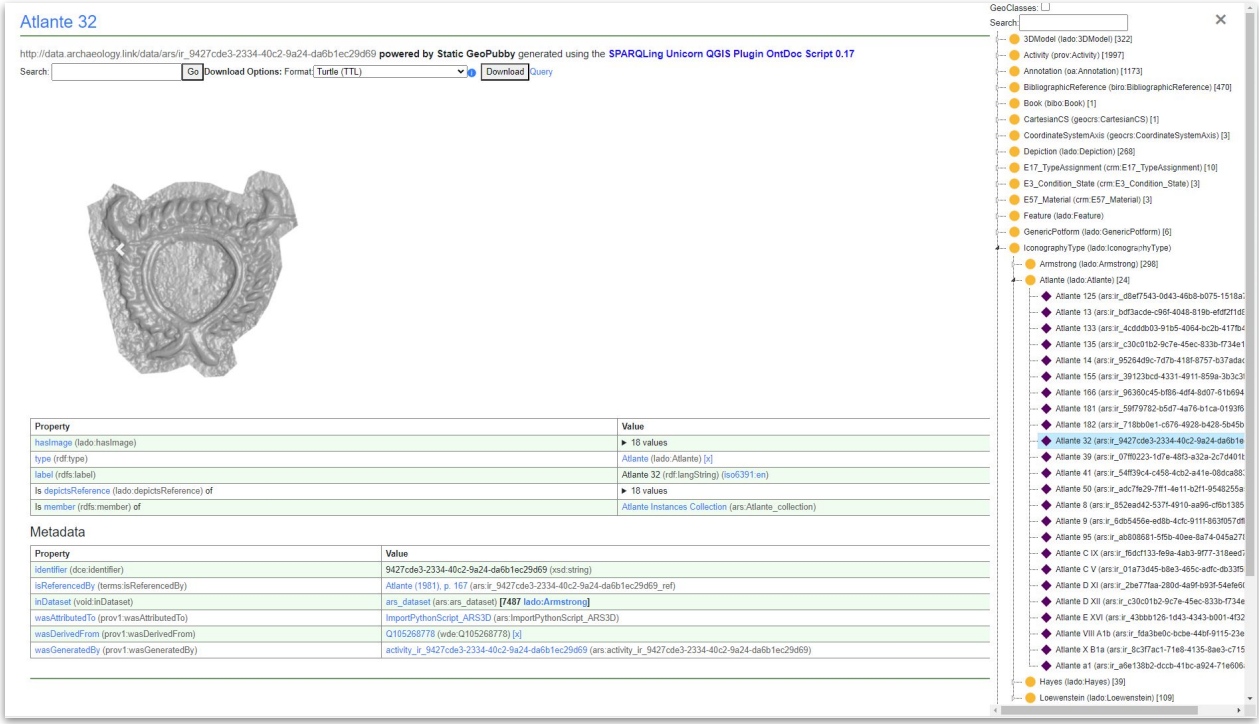




Documentation Tool: HTML+RDFa Rendering

- Idea: Create static web pages that mimic already established Linked Open Data Browsers
- Generate a Protegé-inspired class tree for navigation
- Create statistics of the dataset and publish them using VOID files
- Detect a variety of common vocabularies to create customized page widgets and pages:
  - 3D Model Viewer
  - Leaflet View for Geometries
  - Dictionary View for Ontolex-Lemon dictionaries

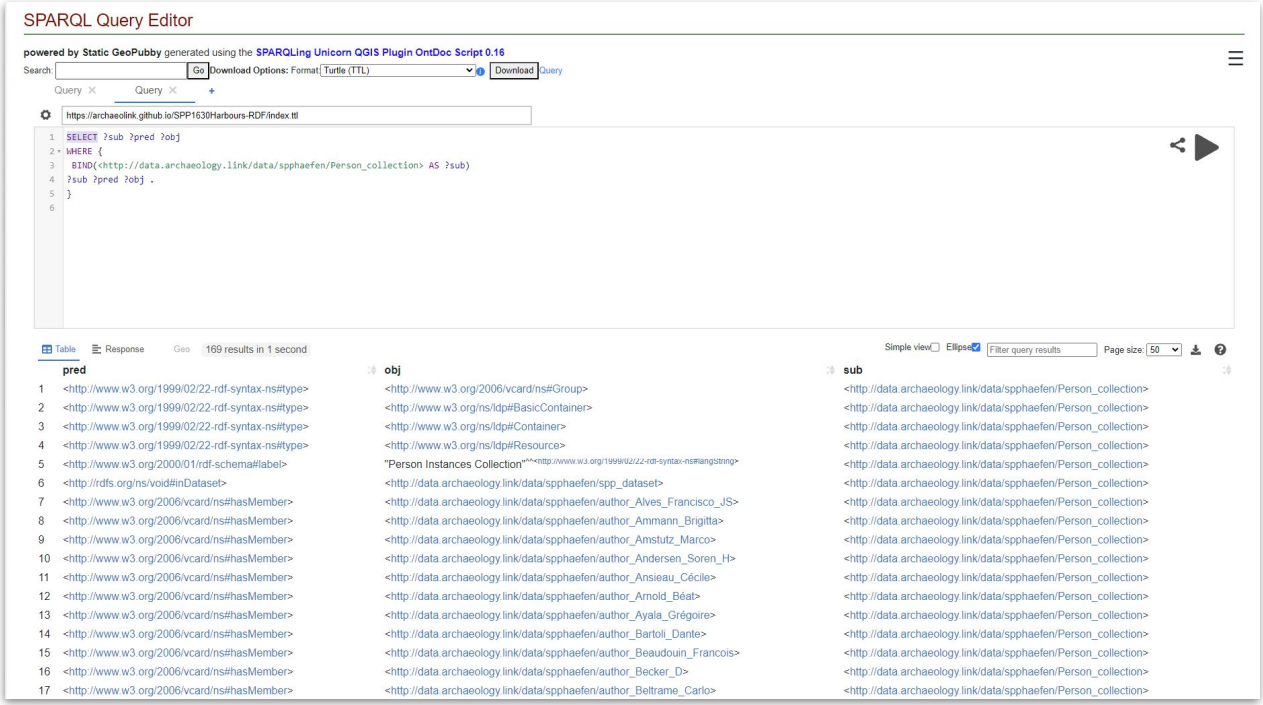
Example: ARS-LOD Data



Documentation Tool: SPARQLing Data Dumps in JS

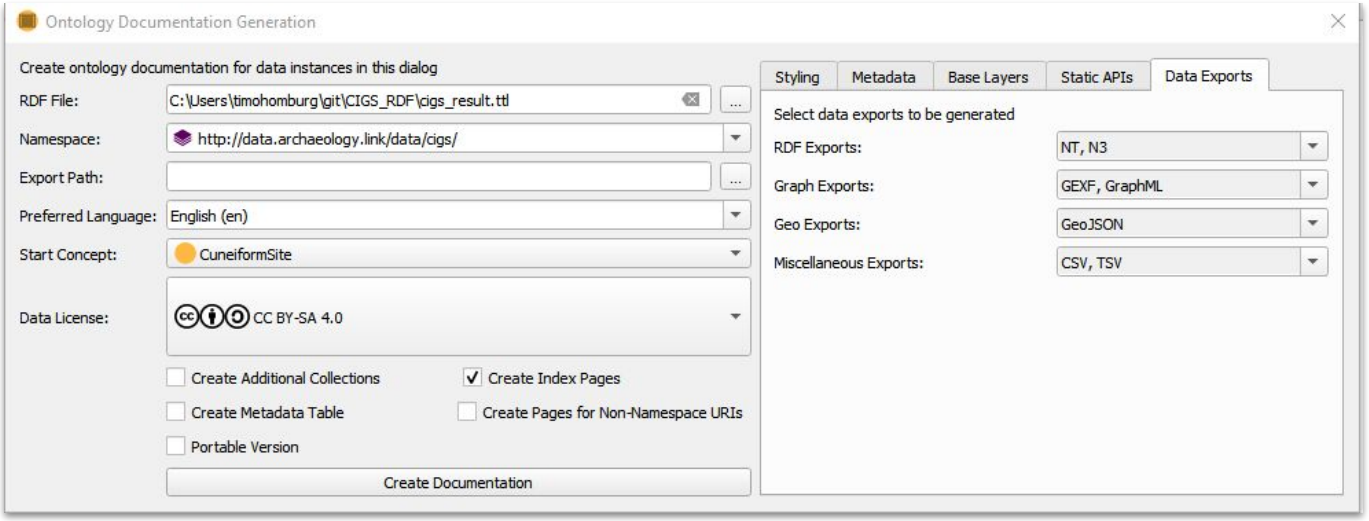
- Static Deployments cannot provide SPARQL endpoints
- Data Dumps can be queried in-browser using JavaScript
- Data Dumps can be queried by external tools loading the Linked Data Dump File

Example



Documentation Tool: Data Exports

- Different research communities are used to data being provided differently
- The SPARQL Unicorn Ontology Documentation Tool may create data exports for the following communities:
  - Different RDF serializations (.TTL, .JSON-LD, .N3)
  - Graph Analysis Data (.graphml, .gexf among others)
  - GeoExports (GeoJSON, KML, GML)
  - Relational Data Exports (.CSV, .TSV, .JSON)



Datadump Metadata - VOID(ext)

- Machine-readable discovery of the dataset
  - Automatically generated topics from vocabulary usage (e.g. GeoSPARQL: dbp:geodata)
  - Automatically derived topics from vocabulary statements
  - Statistics about the dataset (how many triples, subjects, predicates, objects etc.)
  - Which vocabularies are used?
  - Connections to other datastores

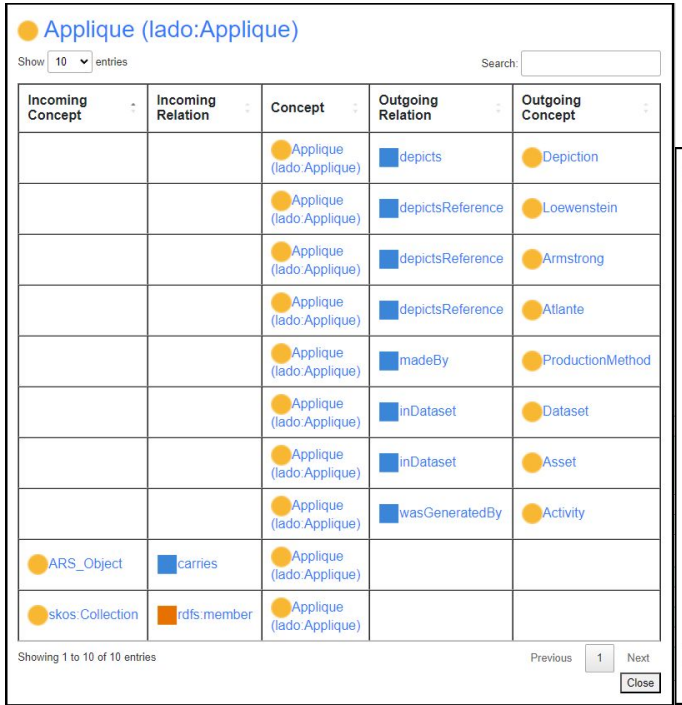
Example

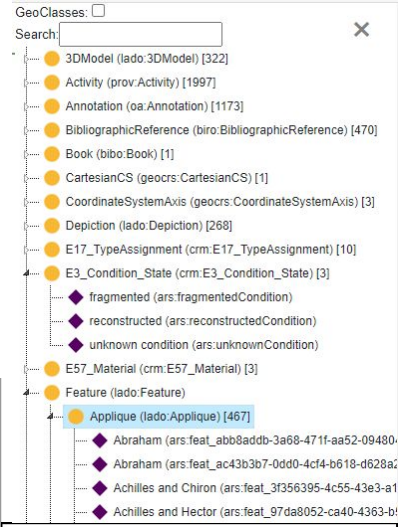
▼ Dataset Statistics [VOID]

Property	Value
classes	7 (xsd:integer)
entities	17159 (xsd:integer)
distinctObjects	53326 (xsd:integer)
distinctSubjects	18678 (xsd:integer)
properties	20 (xsd:integer)
triples	115216 (xsd:integer)
propertyClasses	0 (xsd:integer)
averagePropertyIRILength	38 (xsd:integer)
languages	1 (xsd:integer)
distinctBlankNodes	0 (xsd:integer)
datatypes	0 (xsd:integer)
distinctLiterals	35576 (xsd:integer)
averageSubjectIRILength	77 (xsd:integer)
averageObjectIRILength	62 (xsd:integer)
averageLiteralLength	17 (xsd:integer)
distinctIRIReferences	72024 (xsd:integer)
distinctRDFNodes	107600 (xsd:integer)

Datadump Metadata - ClassTree (Vocabulary)

- Human-readable Discovery
  - Classtree navigation
  - Additional generation of Collection instances for better navigation
  - Data Schema Views on Class Items
  - Views on the position of classified items in the graph
  - Classtree is captured as its own vocabulary (https://purl.org/vocab/classtree)





Static APIs

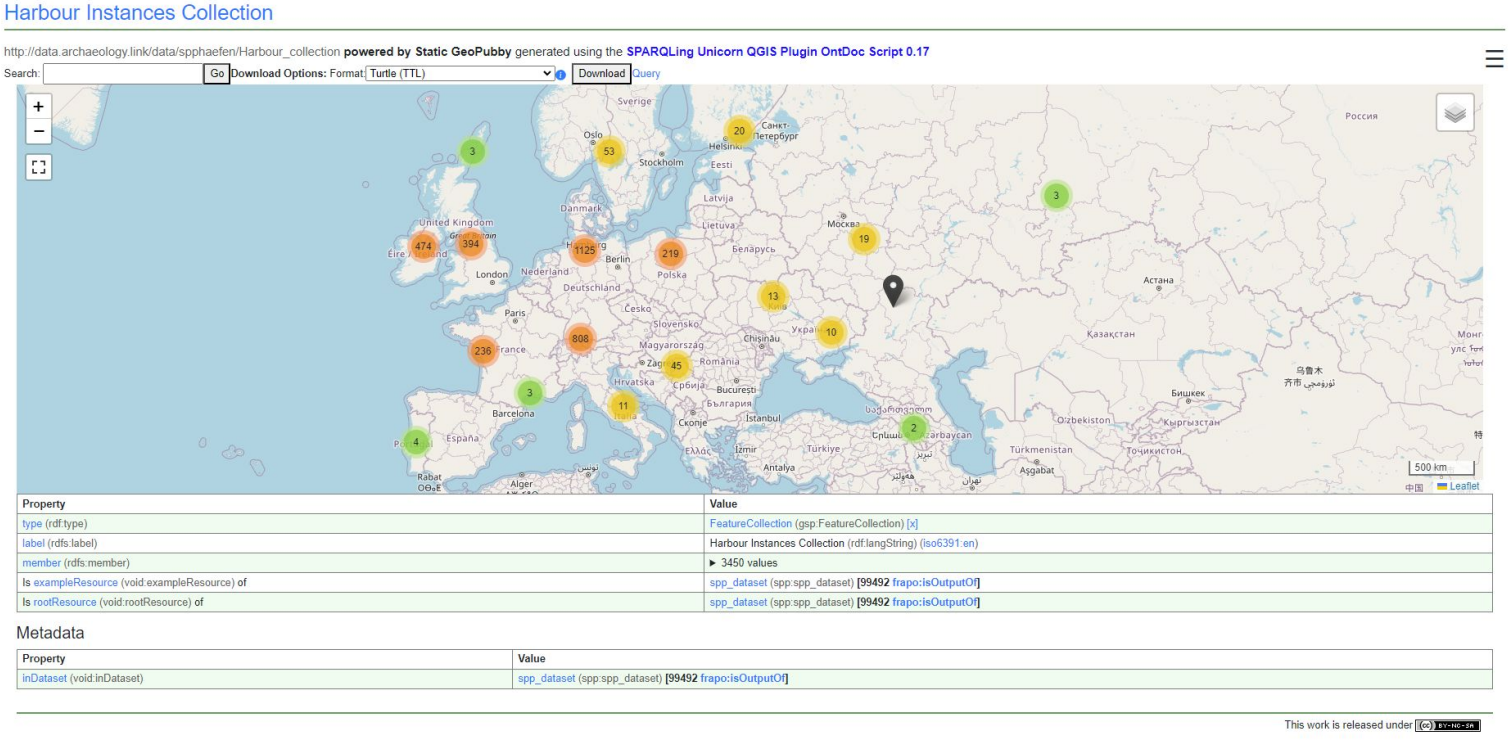
- *"A collection of flat (JSON?) files that live on a webspace"*
- Static APIs may be used to mimic existing APIs with a sacrifice of features that real APIs provide
- We can use static APIs to provide data in RDF Dumps to research communities in ways they are used to
- Do other (research-)communities provide static API definition that are useful?
    - Do they have to?
    - Can we adopt supposedly non-static APIs and publish them statically?

Which APIs, in their static form, provide enough functionality to be useful?

Static APIs: Example QGIS and OGC API Features

SPPHarbour Dataset:

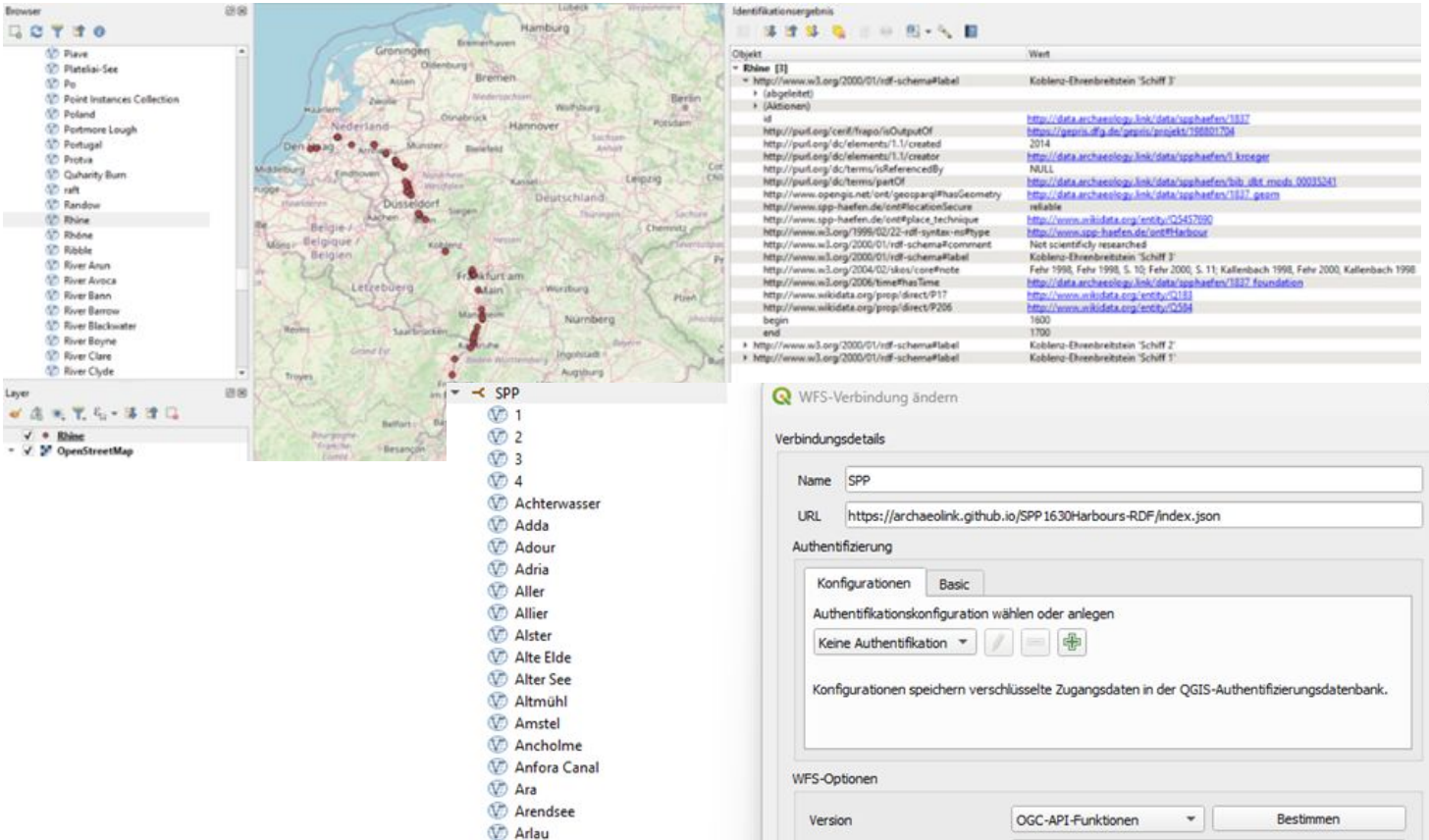
- A dataset of medieval Harbours in Europe
- GeoData modeled with the GeoSPARQL vocabulary
- Static Deployment as a Github Page



Static APIs: Example QGIS and OGC API Features

In **QGIS**:

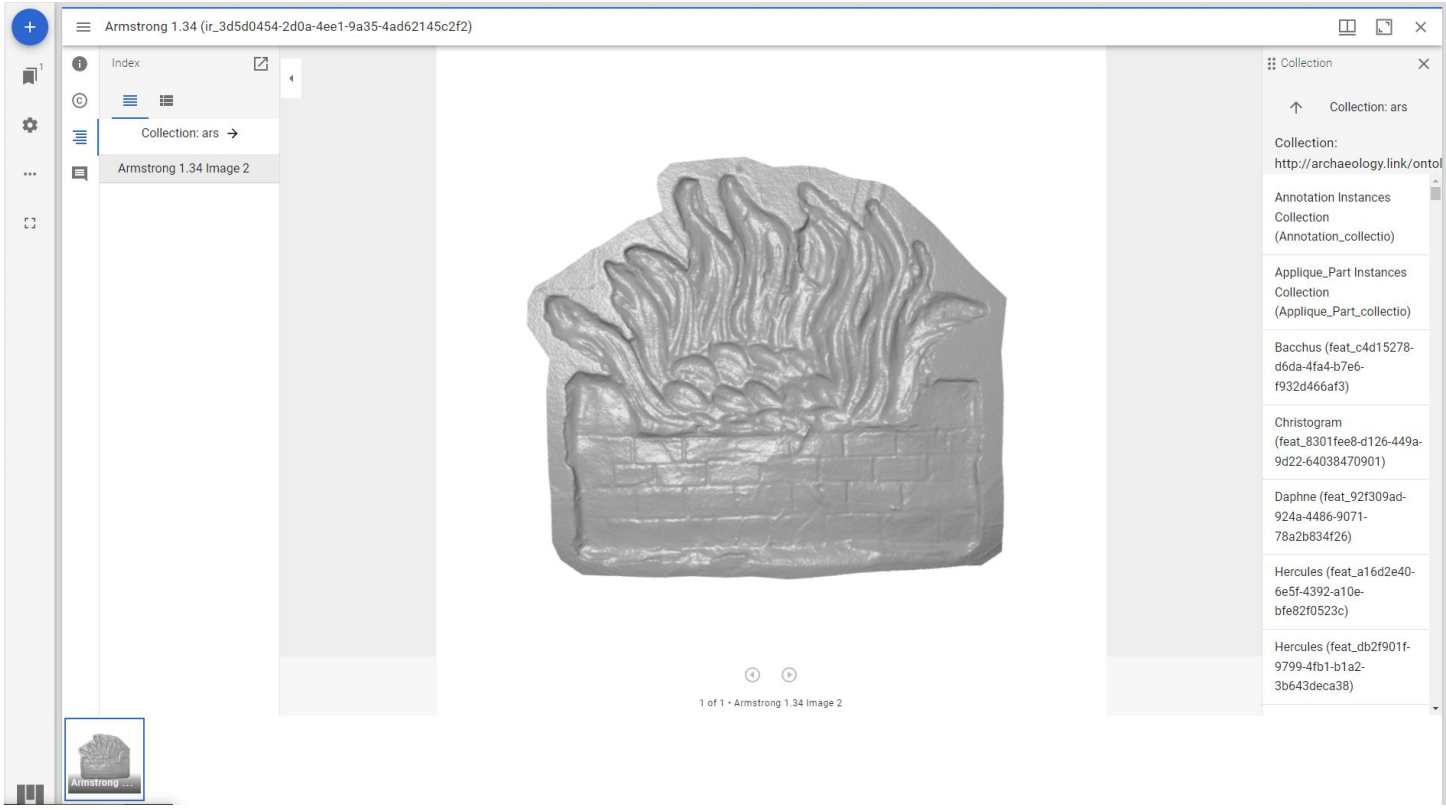
- Add static deployment as OGC API Features service
- Classes in RDF graph become FeatureCollection defintions in QGIS
- Only full FeatureCollections can be loaded
- No serverside filtering/searching



Static APIs: Example IIIF

- Expose Image data hosted on Zenodo using a static IIIF Deployment
- Use the JS Viewer Mirador (deployed with the HTML Dump) or any other IIIF View to access the data
- Only image loading is supported
- Image resizing, rotating etc. would require a server

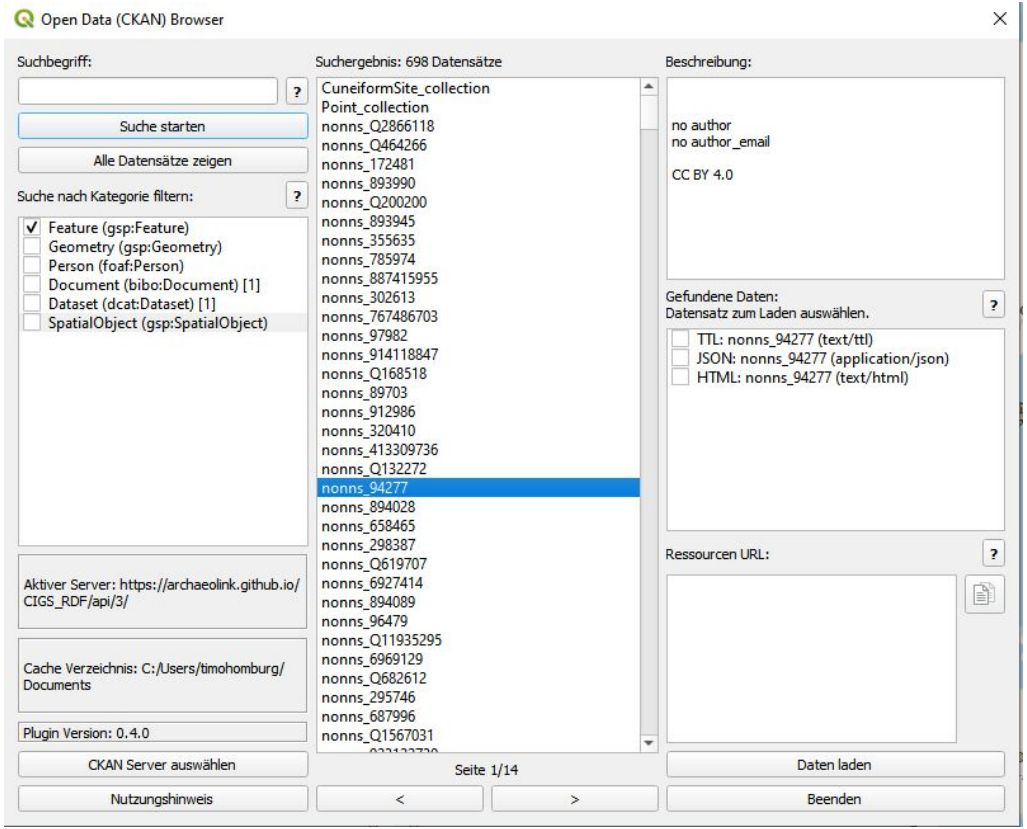
Example



Static APIs: Example CKAN

- Expose generated data export formats via the well-known CKAN API
- Datasets become accessible for a variety of CKAN clients without losing their linked open data context
- Searching via CKAN does not work
- Listing all CKAN datasets works

Example
https://archaeolink.github.io/CIGS_RDF/api/3/



SPARQL Unicorn Ontology Documentation in RDM

- SPARQL Unicorn Ontology Documentation **helps with**:
  - Publishing Research Data
  - Archiving Research Data
- SPARQL Unicorn Ontology Documentation **enables**:
  - Research Data Discovery
  - Research Data Reuse
- SPARQL Unicorn Ontology Documentation **simplifies**:
  - Research Data Analysis
  - Research Data Processing

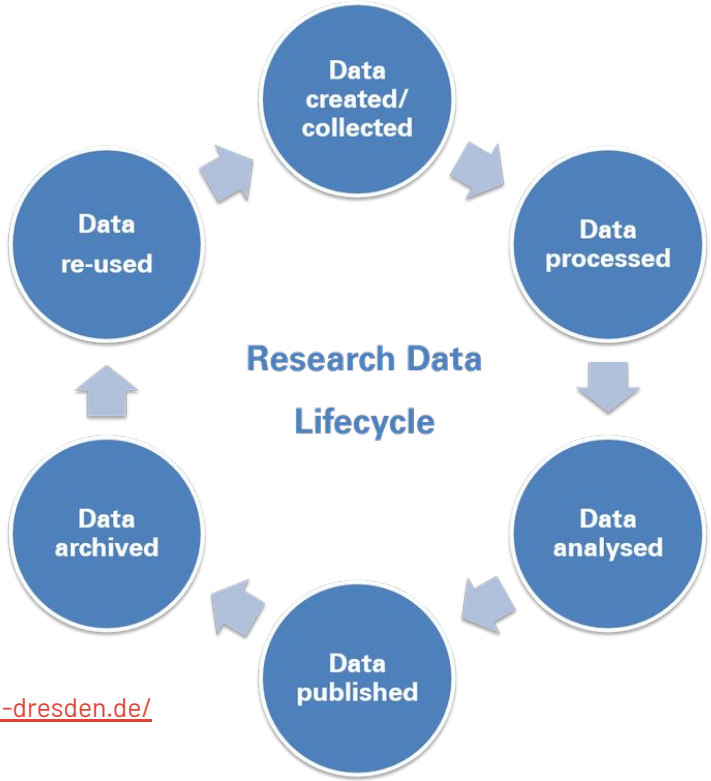


Image by https://tu-dresden.de/

Conclusions

- Research projects producing linked open data are often constrained to publishing LOD Dumps
- Discoverability of RDF Dumps by different communities can be enhanced by
  - Providing HTML Renderings
  - Additional data exports
  - Data access via static APIs
- We believe that easily generated access to LOUD Dumps can be a way to enhance the exposure and reuse of research data

Future Work

- Publishing RDF Dump Contents as Public Solid Pods
- Detection and visualization of further vocabulary contents
- Integration of further common (static) APIs
- HTML Templating

Finis!

Questions?