Scuola universitaria professionale della Svizzera italiana
Dipartimento ambiente costruzioni e design
Istituto scienze della Terra

SUPSI

Open geospatial standards and reproducible research

Massimiliano Cannata[1], Gregory Giuliani[2], Jens Ingensand[3], Olivier Ertz[3], and Maxime Collombin[3]

[1]SUPSI, Istituto scienze della Terra, DACD, Canobbio, Switzerland (massimiliano.cannata@supsi.ch)

[2]University of Geneva, Institute for Environmental Sciences/enviroSPACE, geneva, Switzerland

[3]HEIG-VD, Yverdon-les-Bains, Switzerland

https://doi.org/10.5194/egusphere-egu23-14845

Disclaimer

In view of best practice of Open Science principle
we're sharing a project idea we're currently exploring at IST- SUPSI to stimlate
feedbacks, **further ideas**, **collaborations**.

OSGeo **@istSOS** team
we're starting **creation a proof of concept**
to implement this idea within the SensorThingsAPI

Please refer to this presentation with this DOI:
https://doi.org/10.5194/egusphere-egu23-14845

Geospatial research flow is just like any other research

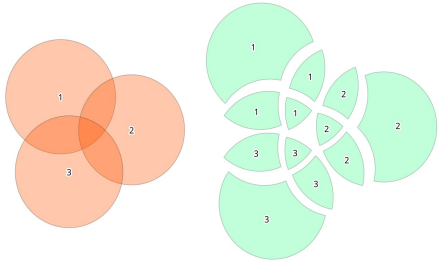
Geodata

01 **file data** (shp, gpkg, gml, GeoJSON…)

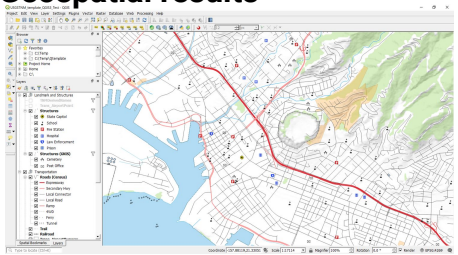
02 **databases** (postGIS, MongoDB…)

03 **geoservices** (ESRI Feature Service, OGC services, REST APIs…)

Geoprocessing

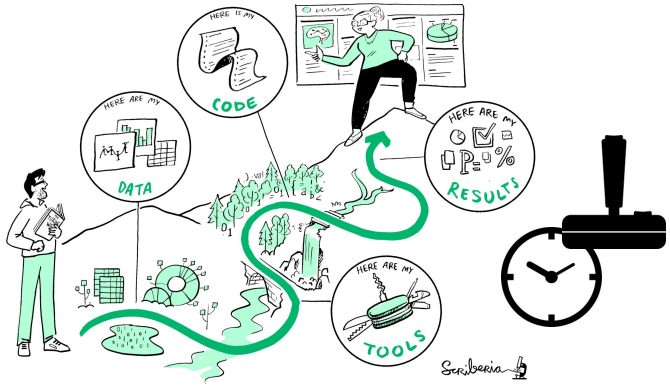


Geospatial results



and traditional archiving is an option to pursue reproducible research

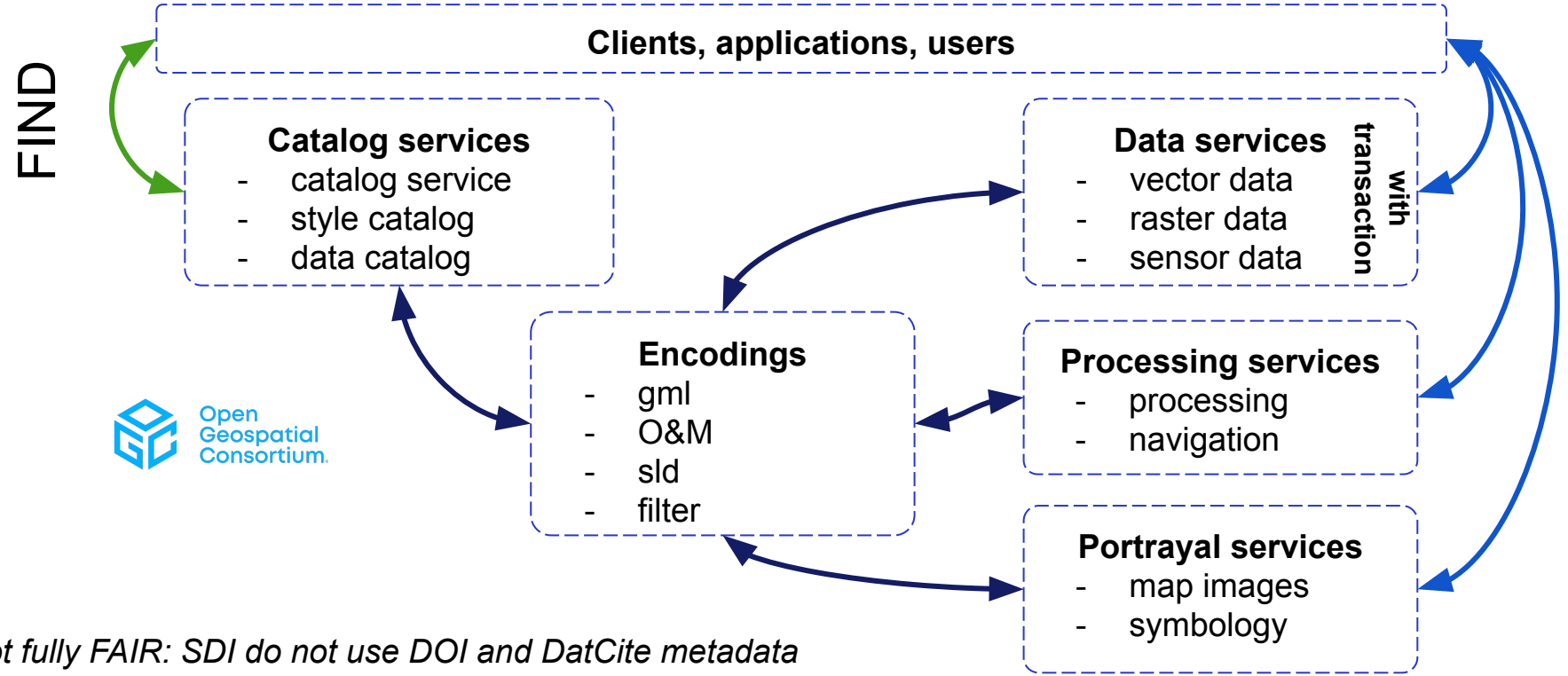
1. archive data in an open repository (e.g.: shp, geoJSON, GeoPackage)
2. define and reference archived processing tools (e.g.: github release link, doi)
3. archive your implemented processing routines (e.g.: scripts, GIS workflow, notebook)
4. document and archive your results (e.g.: paper, reports, blog)



img: 10.5281/zenodo.3332807

In Geospatial interoperability is implemented with OGC Services

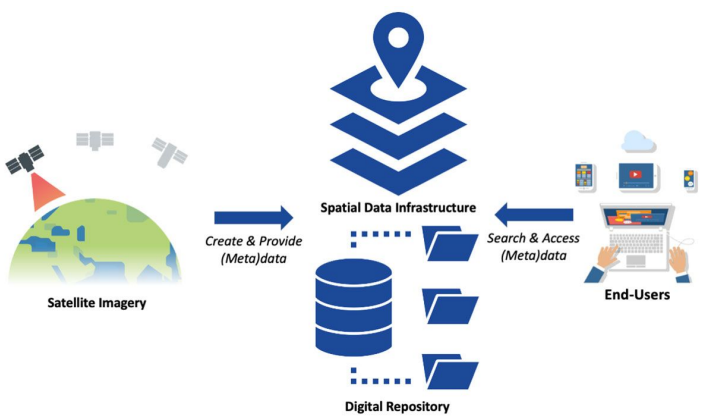
geospatial data discoverability, accessibility, sharing and re-use is commonly achieved through Spatial Data Infrastructures



not fully FAIR: SDI do not use DOI and DatCite metadata

SDI & long-term preservation (coupling OGC & FAIR repo)

Giuliani et. al (2021) proposed a solution to enhance traditional SDI with a long-term preservation digital repository ensuring full compliance with FAIR principles while at the same time benefiting from geospatial services capabilities.



GeoNetwork opensource

GeoServer

metadata

data

Geospatial Standards
(OGC, ISO19xxx)

YARETA

FAIR compliant archive
(doi, datacite)

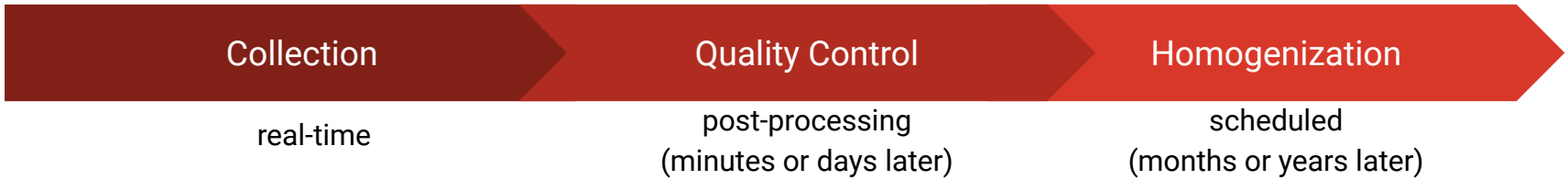
Giuliani, G., Cazeaux, H., Burgi, P. Y., Poussin, C., Richard, J. P., & Chatenoux, B. (2021). SwissEnveo: A FAIR national environmental data repository for earth observation open science. *Data Science Journal*, *20*(1). https://doi.org/10.5334/dsj-2021-022

How to do with dynamic “live” data source with Data Management?

Reference a web service with used parameters to get data is with no guarantee:

tomorrow data may vary from today --> not reproducible results

EXAMPLE: Monitoring Network from metoffice
study: climate change
data web service: OGC SensorThingsAPI
data: 50 years of 10 min data from 250 stations and 5 parameters (~10GB)

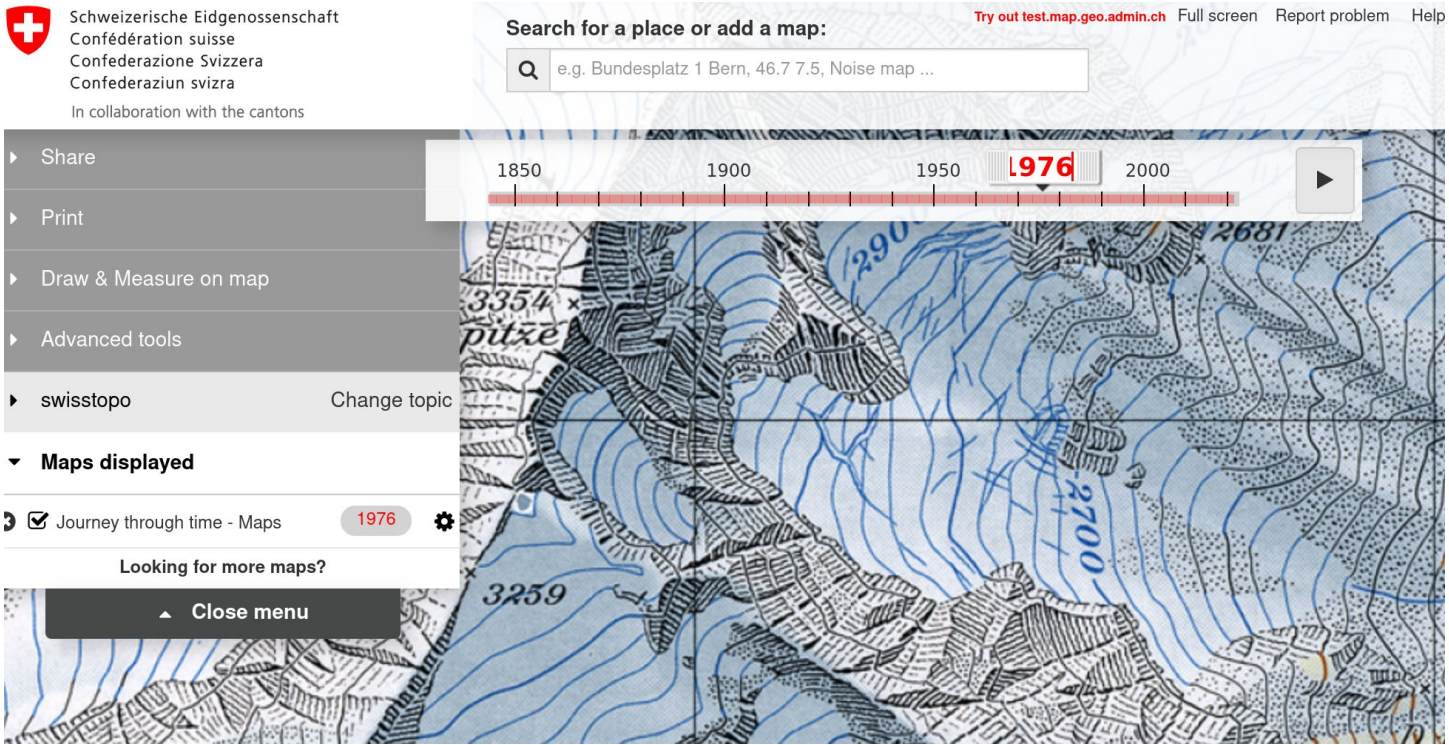


When data size is large and the dataset is highly dynamic (spatio-temporal) saving a snapshot of the dataset at each scientific publication may be:

- **expensive** (10 GB * N publications)
- **inefficient** (download and upload, metadata recreation, etc..)
- **difficult to reproduce** (notebook not working without setup of the same web services)

A Journey through time

Would it be possible to reference data status of a service only using date time?



RDB - SQL “Tempora Data Support”

From 2011 SQL “Tempora Data Support” is an optional feature enabling to access what was the state of data on a specific <time instant>

DB2, Oracle somehow implement it

- **System time** (ACID time):
  - system maintained (versioned)
  - only past time
  - default is now

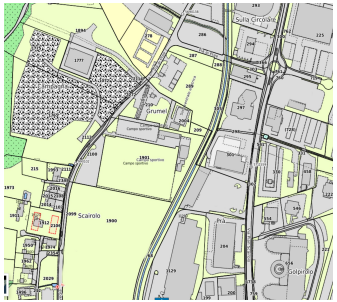
how was the data in a given period

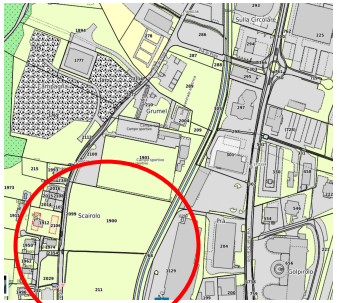
- **Business time** (validity time, application time):
  - user maintained
  - future dates may have sense
  - time resolution user defined

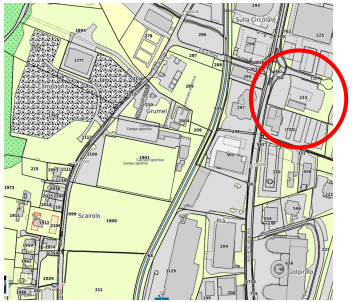
for which period the data is meant to be used

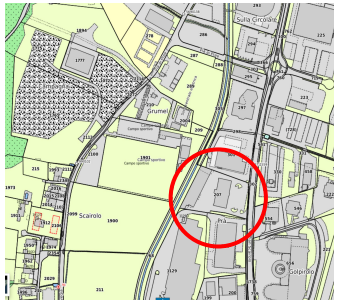
Vanroose, P. (2015). Temporal Data & Time Travel in PostgreSQL.

Cadastral traveltime









2019-03-24T14:55:03.223Z

2020-07-12T09:07:55.841Z

2020-11-19T11:28:37.047Z

PATCH: parcel 1900 changed geometry

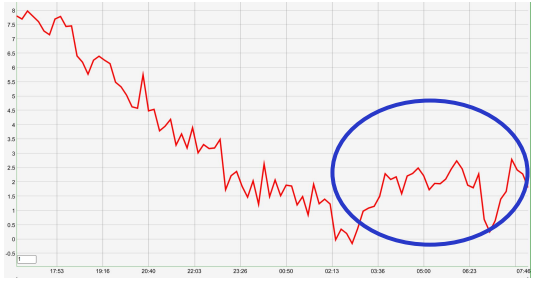
POST: new parcel 211 registered

POST: new building in parcel 233

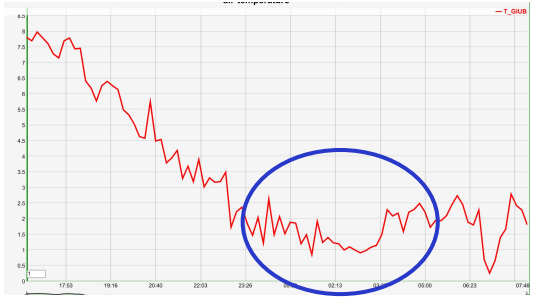
POST: new building in parcel 207

Observation traveltime

data as of
2022-12-29T14:55:03.223Z



data as of
023-01-06T09:07:55.841Z,



data as of
2023-02-02T11:28:37.047Z

- gap filling using spline interpolation techniques
- manual correction of erroneous data

	id	phenomenonTime	resultTime	result	resultQuali			system_time_validity
1	1	2023-03-25 15:30:00.000 +0100	2023-03-25 15:30:00.000 +0100	23.5	[NULL]	1	1	["2023-04-23 15:56:02.702888+02",infinity)
2	2	2023-04-23 15:56:05.306 +0200	2023-04-23 15:56:05.306 +0200	99	[NULL]	1	1	["2023-04-23 15:56:05.306006+02","2023-04-23 15:56:09.712914+02")
3	2	2023-04-23 15:56:05.306 +0200	2023-04-23 15:56:05.306 +0200	100	{"quality": 100}	1	1	["2023-04-23 15:56:09.712914+02","2023-04-23 15:56:15.312782+02")
4	2	2023-04-23 15:56:05.306 +0200	2023-04-23 15:56:05.306 +0200	200	{"quality": 200}	1	1	["2023-04-23 15:56:15.312782+02","2023-04-23 15:56:18.964311+02")
5	2	2023-04-23 15:56:05.306 +0200	2023-04-23 15:56:05.306 +0200	300	{"quality": 300}	1	1	["2023-04-23 15:56:18.964311+02",infinity)

127.0.0.1:8018/v1.1/Observations

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

```
0:
    id:                 1
    phenomenonTime:     "2023-03-25T14:30:00+00:00"
    resultTime:         "2023-03-25T14:30:00+00:00"
    result:             23.5
    resultQuality:      null
    validTime:          null
    parameters:         null
    datastream_id:      1
    feature_of_interest_id: 1
1:
    id:                 2
    phenomenonTime:     "2023-04-23T13:56:05.306006+00:00"
    resultTime:         "2023-04-23T13:56:05.306006+00:00"
    result:             300
    resultQuality:      '{"quality": 300}'
    validTime:          null
    parameters:         null
    datastream_id:      1
    feature_of_interest_id: 1
```

NOW

127.0.0.1:8018/v1.1/Observations?as_of_system_time=2023-04-23T15:56:16.123000%2B02:00

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

```
0:
    id:                 1
    phenomenonTime:     "2023-03-25T14:30:00+00:00"
    resultTime:         "2023-03-25T14:30:00+00:00"
    result:             23.5
    resultQuality:      null
    validTime:          null
    parameters:         null
    datastream_id:      1
    feature_of_interest_id: 1
1:
    id:                 2
    phenomenonTime:     "2023-04-23T13:56:05.306006+00:00"
    resultTime:         "2023-04-23T13:56:05.306006+00:00"
    result:             200
    resultQuality:      '{"quality": 200}'
    validTime:          null
    parameters:         null
    datastream_id:      1
    feature_of_interest_id: 1
```

AS OF

Preliminary proof of concept

Integrating **as_of_system_time** in OGC standards (?)

1. GET data should include the optional **AS_OF_SYSTEM_TIME** parameter, which can optionally be a time instant or a time period.

2. historic values should be **immutable**, to guarantee persistence

Additionally

3. Data returned from a GET request should always refer the **SYSTEM_TIME** to which data refer to.

4. Transactional operations should include commit metadata (git-like):
   a. **commit message** to clarify the operation and reason for data changes
   b. **commiter name** to keep track of author of the change

Thanks

massimiliano.cannata@supsi.ch