eurac research

Data Exchange Server (DES) application: data handling from acquisition to sensor observation services

Costa Armin, Frisinghelli Daniel, Marsh-Hunn Daniel,
Monsorno Roberto, Rossi Mattia, Ventura Bartolomeo, Vianello Andrea

Institute for Earth Observation – eurac research

Geospatial Sensor Webs - from data to information in the Sensor Web
August 28 – 30, 2017

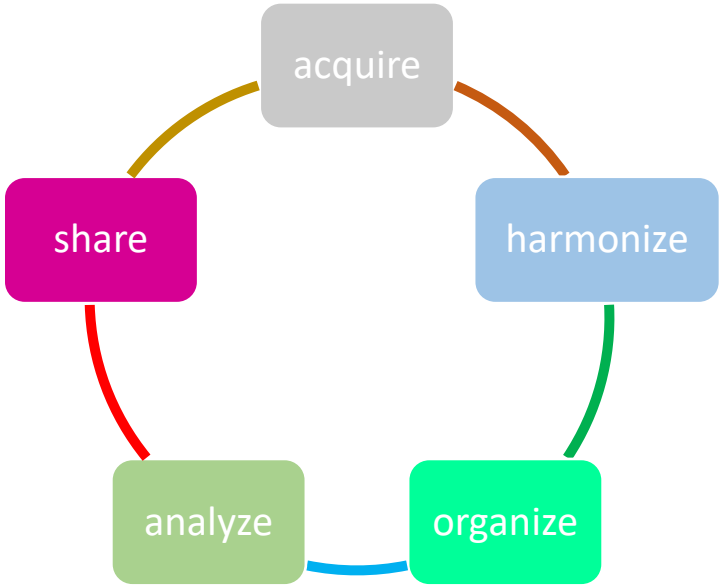
Researcher's question

how to handle multiple data?

Especially if they belong
to different sensors at different
scale
with different characteristics

Common Technology Platform

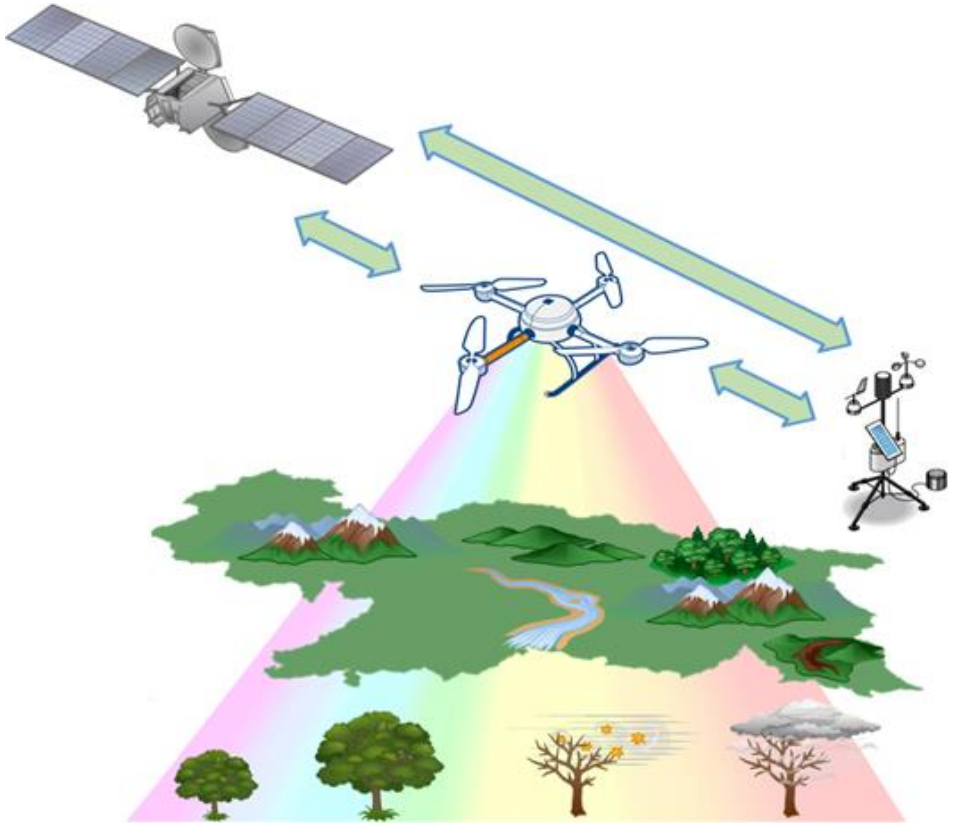
Automation of the Data Life Cycle



Test case: MONALISA project

The MONALISA (*MON*itoring key environmental parameters in the *AL*pine environment *I*nvolving *S*cience, technology and *A*pplication) project's main goal is the ***development of multi-scale monitoring approaches for key environmental parameters*** and production processes using innovative monitoring technologies and non-destructive methods in the ***application field of agriculture***.

biophysical parameters and all information coming from different sources:
Remote Sensing, UAV, ground stations, laboratory

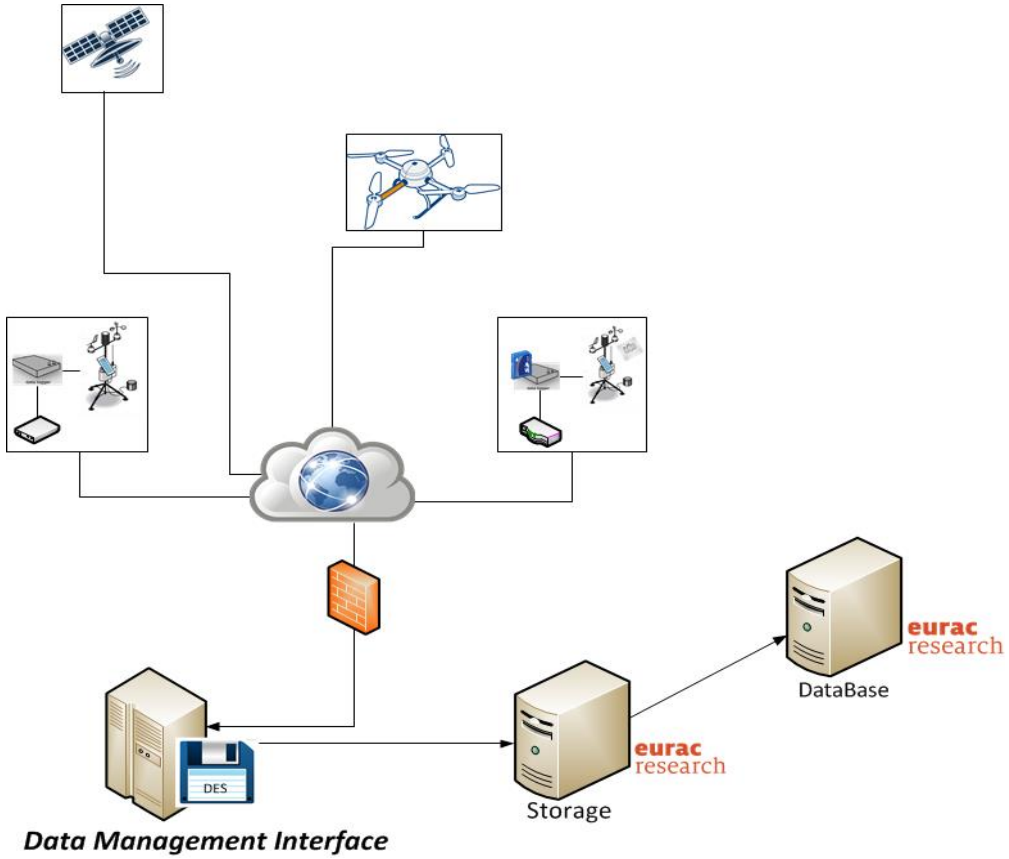


For further information please visit: http://monalisasos.eurac.edu/sos/

Data Acquistion: DES (Data Exchange Server)

DES (**Data Exchange Server**) is a standalone application, developed in-house, to enable the execution of tasks and data transfers from and to different systems by means of a set of XML processor configuration files.

It implements a concurrent, multi-threaded mechanism to transfer different kinds of data in PUSH and GET mode by using standard protocols like SFTP, FTP, SSH.



DMI (**Data Management Interface**):
data are collected and distributed on an online server available to the public at eomountdata.eurac.edu via HTTP and FTP protocols.

eurac research

DES and data life cycle

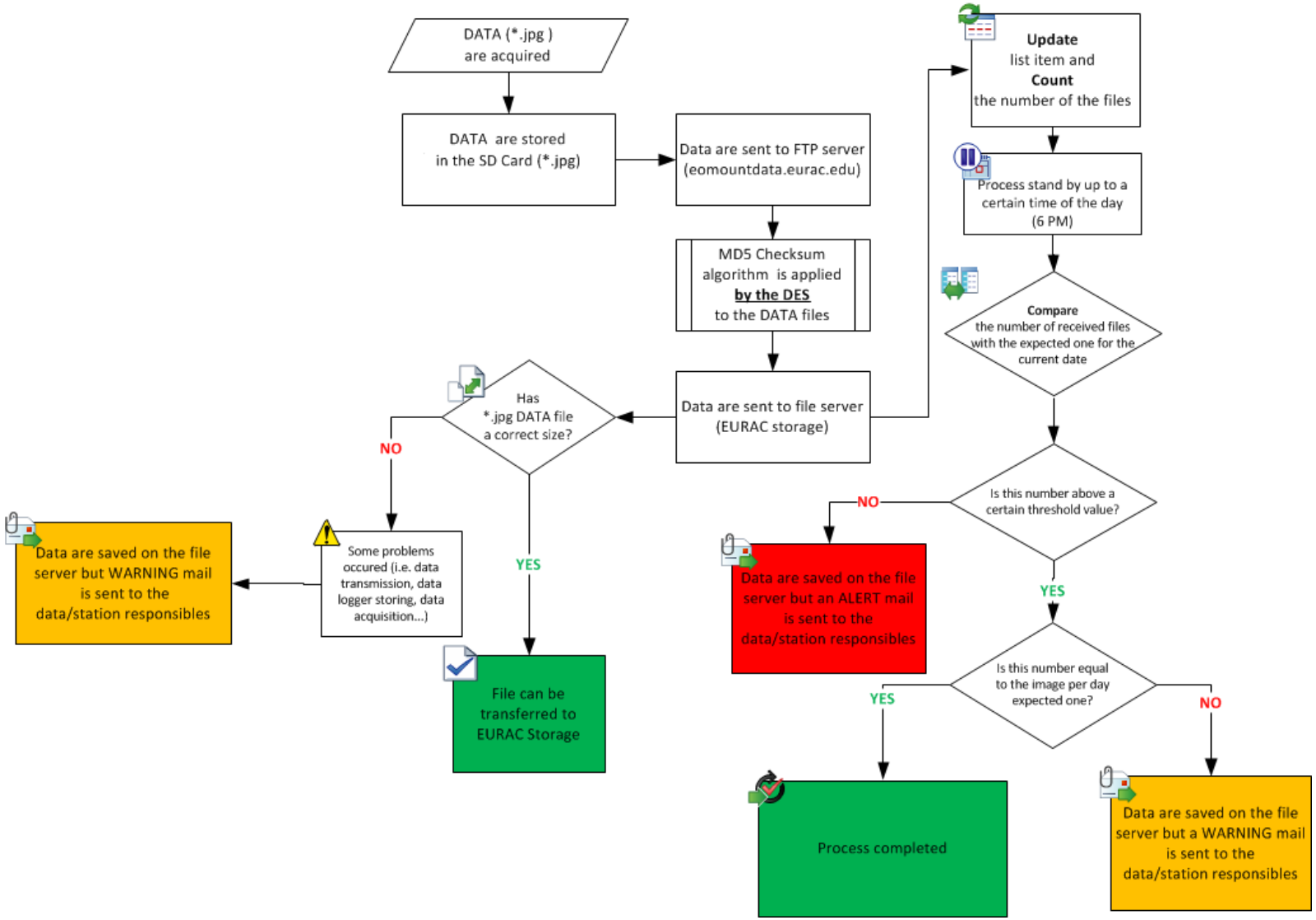
(example 1)

To summarize

DES is a multi-tasking application that can execute any kind of configurable tasks or jobs implemented as dedicated plug-ins.

In this particular scenario dedicated plug-ins, for example, implement:

- ✓ data collection,
- ✓ data validation,
- ✓ data check,
- ✓ data warning,
- ✓ data move to final location.

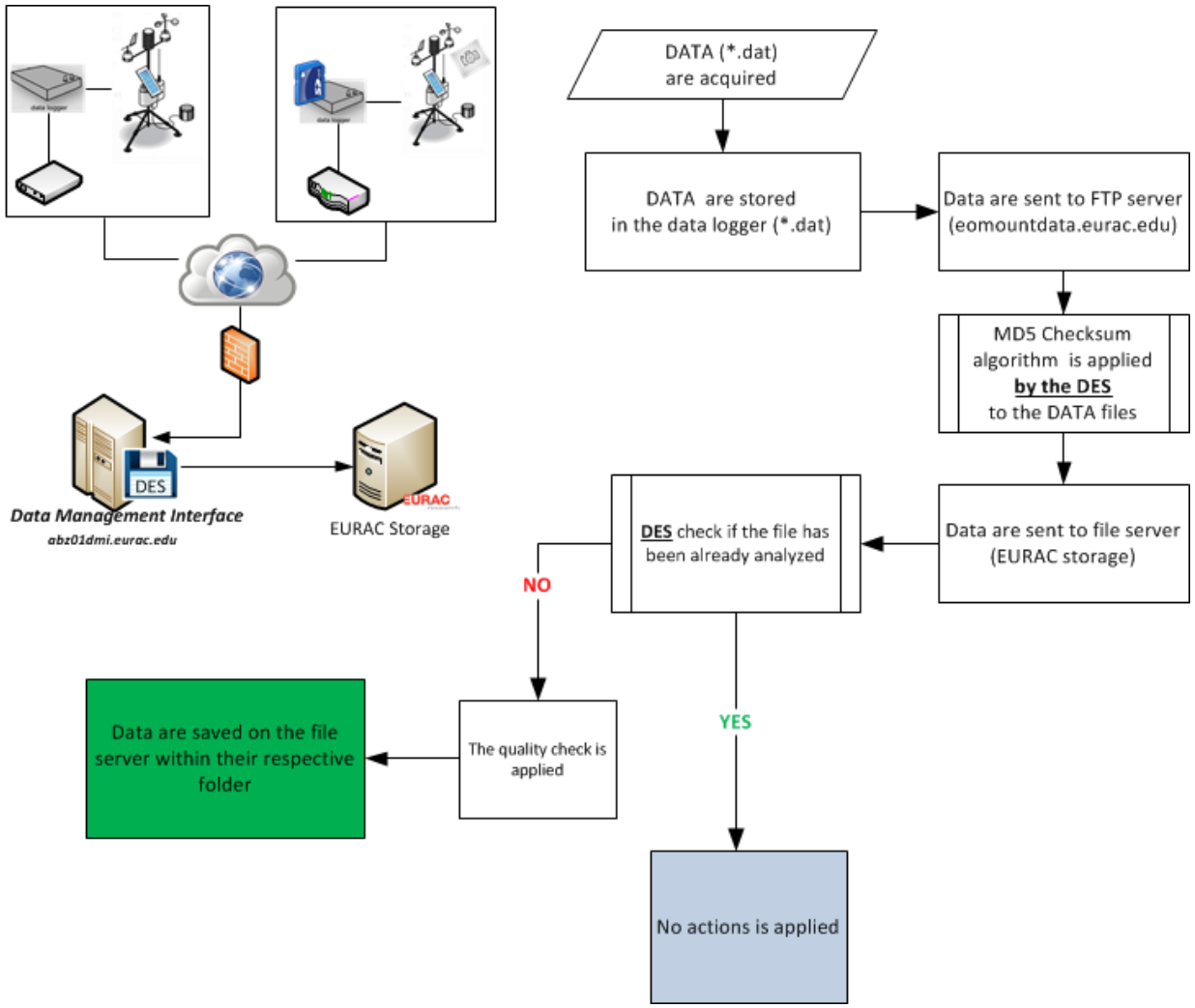


DES and data life cycle

(example 2)

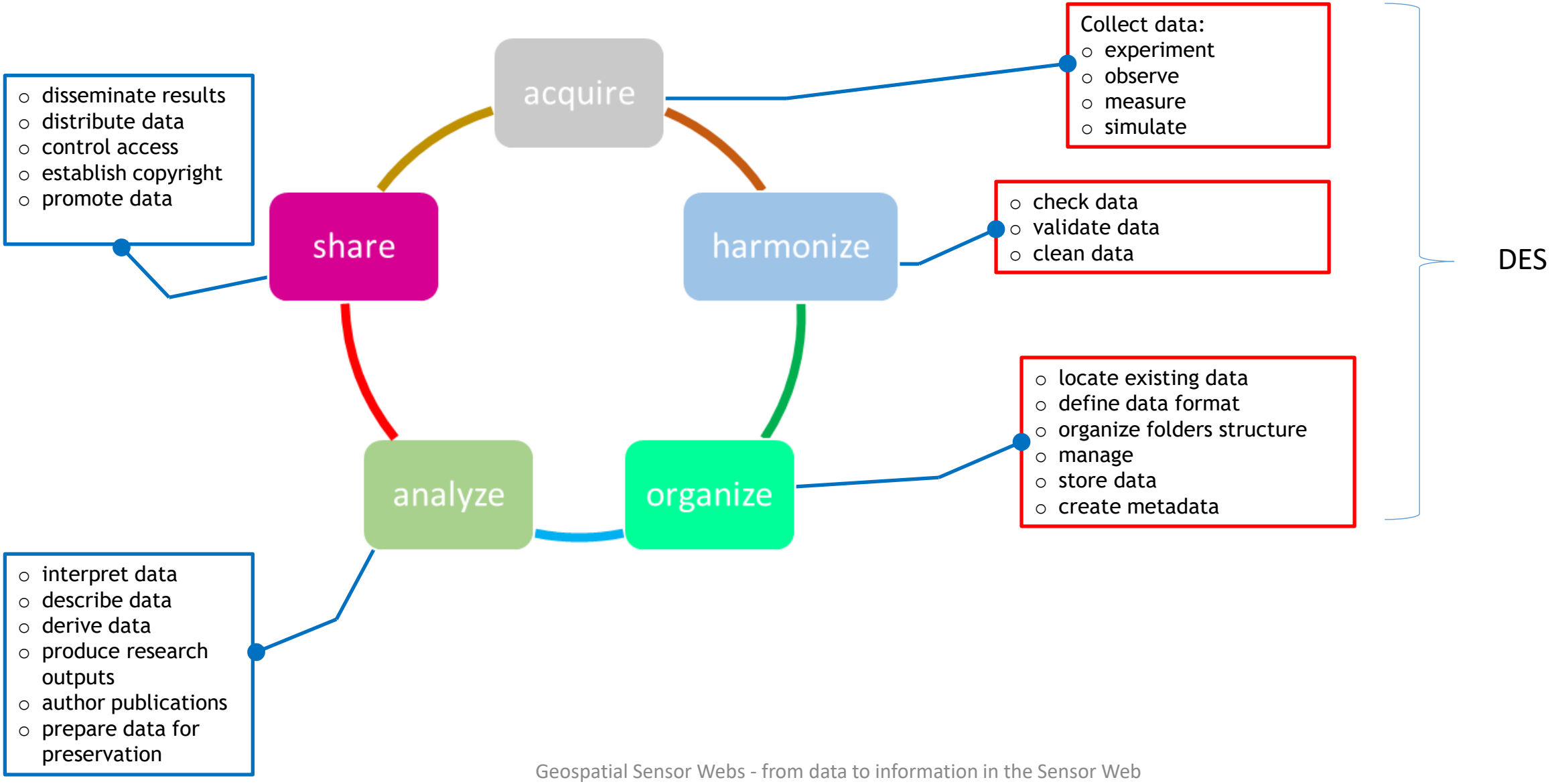
In this particular scenario dedicated plug-ins, for example, implement:

- ✓ data collection,
- ✓ data check,
- ✓ data quality check,
- ✓ data move to final location.



eurac research

data lifecycle



Analyze: Automatic data insertion

Different data organization with respect to the same sensor

Different sensors and stations

Different name of input files

Different data organization

Absence of a unique structure of the input data

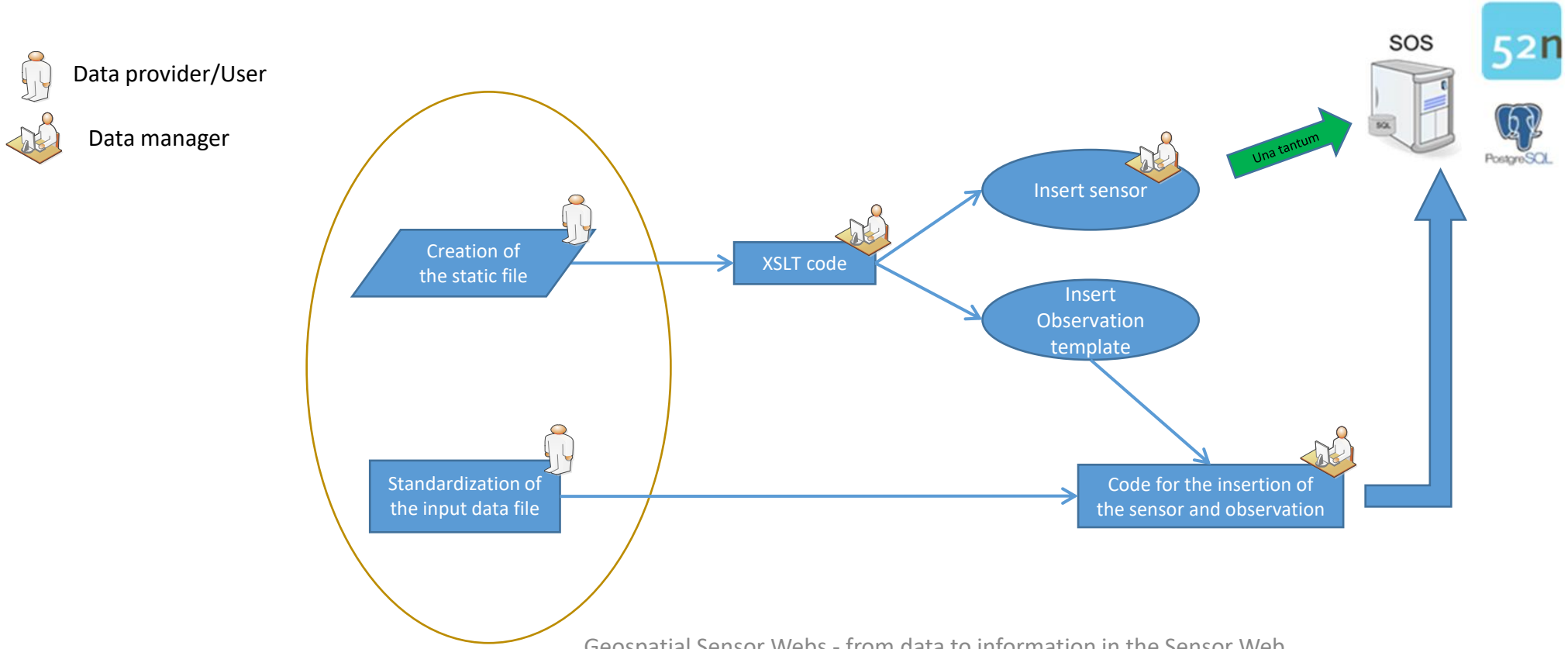
Define rules for the input data format

1. *Input data naming convention*
2. *Date and time format of the measurement*
3. *Structure of the input file*
4. *Presence of the header in each input file*

Automatic data insertion

From the data provider/user point of view only two actions are requested:

1. *Creation of the static file*
2. *Standardization of the input data file following the previous 4 rules*



Automatic data insertion

- **Definition of a static file (XML format)**

The idea of creating a static file is essentially twofold:

- *it will contain all the information useful to create those information for filling the database (Insert Sensor and Insert Observation XML file)*;
- *it can be seen as a raw version of the **metadata** for the considered monitoring station (Feature Of Interest)*.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<static>
   <FOI_list></FOI_list>
   <FOI_coordinates_list></FOI_coordinates_list>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<static>
   <FOI_list>Tramin13er,Unterrain2FuchsangerEppan,
      Gries2NeufeldGreifensteinerweg,Girlan1Lamm,Lana6</FOI_list>
   <FOI_coordinates_list>46.3291770 11.2445070,46.5016200
      11.2504780,46.5017570 11.2802830,46.4558950 11.2795330,46.6165530
       11.1517880</FOI_coordinates_list>
```

```xml
<observedProperty>
   <short_name></short_name>
   <long_name></long_name>
   <long_name_def></long_name_def>
   <uom></uom>
   <id_sensor_num>1</id_sensor_num>
</observedProperty>
```

```xml
<observedProperty>
   <short_name>NDVI_UpRed_Avg</short_name>
   <long_name>Normalized Differenced Vegetation Index Up Red</long name>
   <long_name_def>
      Normalized_Differenced_Vegetation_Index_UpRed_Avg</long_name_def>
   <uom>None</uom>
   <id_sensor_num>1</id_sensor_num>
</observedProperty>
```

```xml
<coordinate>
   <easting></easting>
   <northing></northing>
   <altitude></altitude>
</coordinate>
```

```xml
<coordinate>
   <easting>11.2445070</easting>
   <northing>46.3291770</northing>
   <altitude>239</altitude>
</coordinate>
```

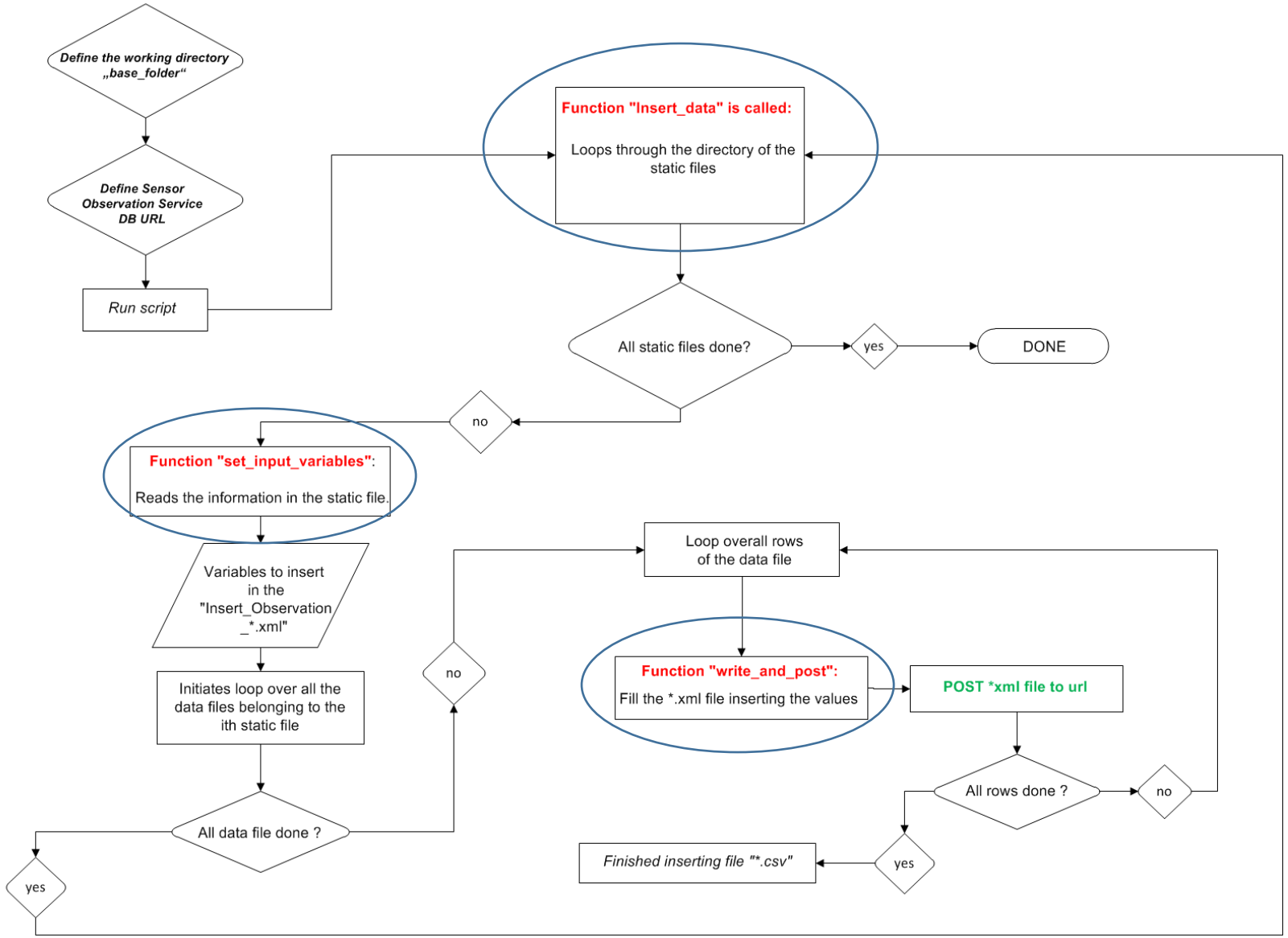
Insert of the observations

A python script has been developed.
The necessary information for this script are:

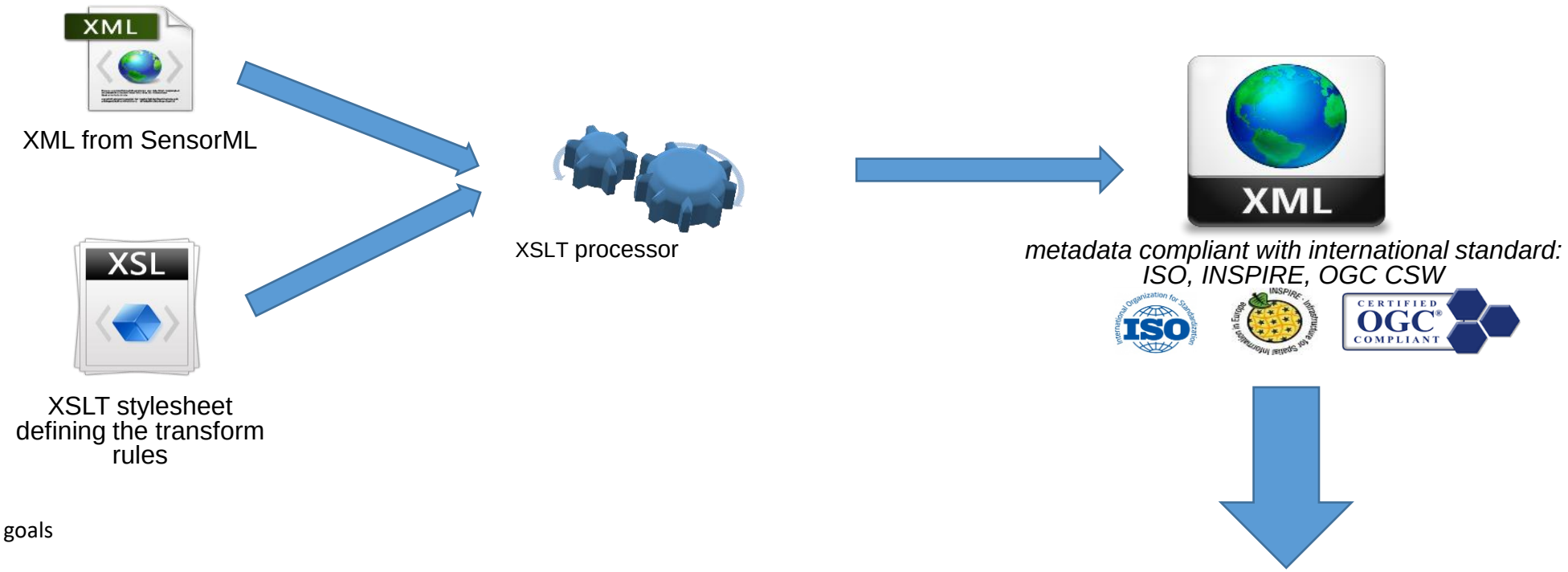
1. Working directory base folder
2. SOS Database URL

The main phases of the script are:

- ✓ The insertion of the data which loops through the directory of the static files
- ✓ Set the input variables extracted from the ith static file
- ✓ Fill the *.xml of the insert observation file inserting the values and post it to the SOS Database



Metadata: data about the data



Search and Discover sensors and their data information

XSLT definition and goals

Definition:

- XSL stands for: ***Ex**tensible **S**tylesheet **L**anguage*, and it is a style sheet language for XML documents.
- XSLT stands for: ***XSL T**ransformations*

Goals:

- XSLT is a transformation language for XML.
- XSLT is a W3C XML language (the usual XML well-formedness criteria apply)
- XSLT can translate XML into almost anything, e.g.
  - ✓ Well-formed HTML (closed tags)
  - ✓ Any XML, e.g. yours or other XML languages like SVG, X3D
  - ✓ Non XML, e.g. PDF, ZIP, … etc

Environmental Database

The MONALISA Database has been implemented using 52North SOS project following the OGC Sensor Web Enablement standards

- 1 unique database
- + 30.000.000 record collected for scientific analysis
- 75 environmental parameters + 10 laboratory measurements
- + 80 installed sensors
- 31 stations of measurements dislocated in the province of Bolzano

- 2 different but complementary ways to access to the Database:
  - Web Application (Helgoland)
  - Plugin SOS_R-Access

Helgoland Web Application

http://monalisasos.eurac.edu/sos/

Visual Exploration and Analysis of Sensor Web Data

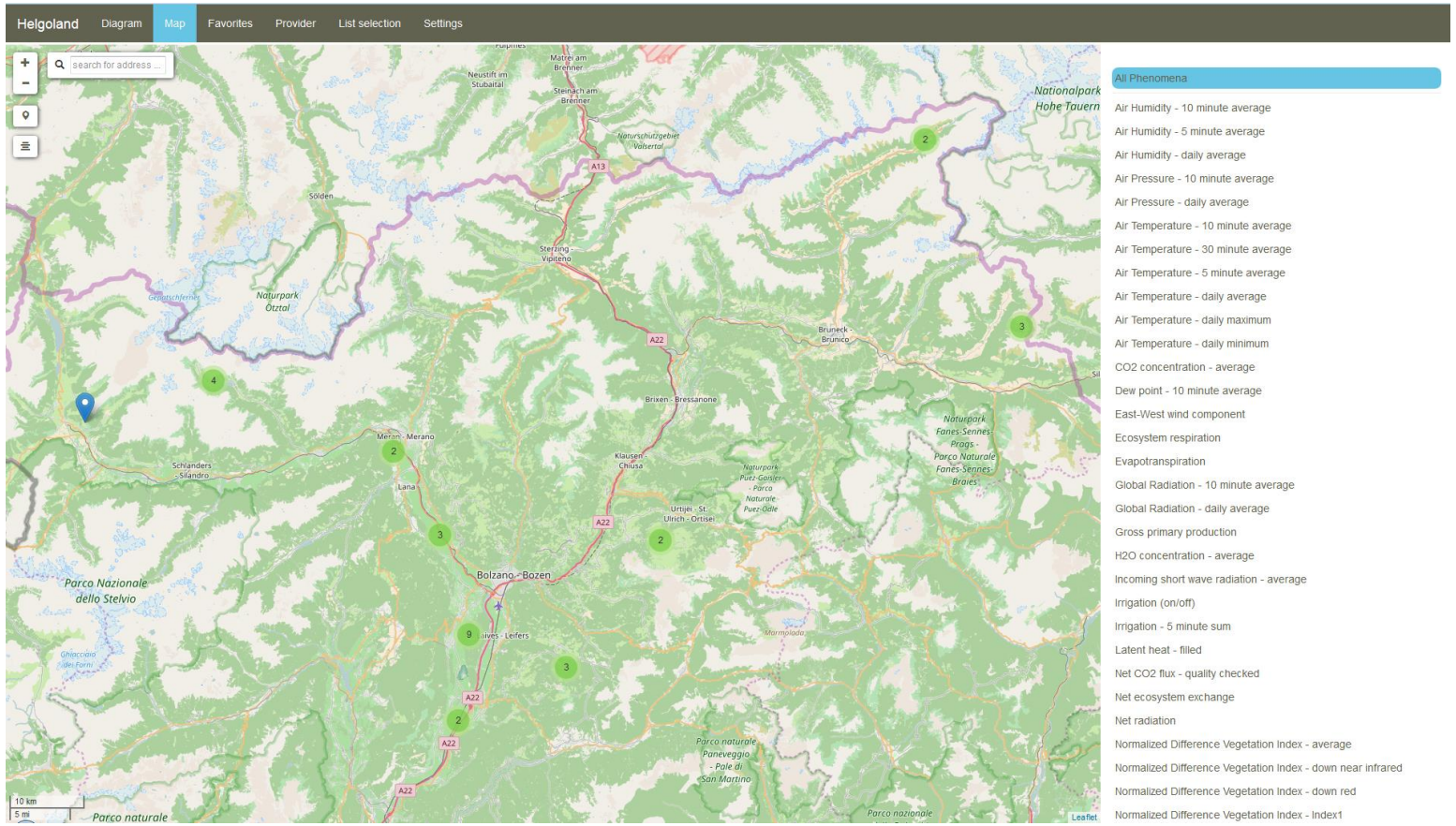
Helgoland is a lightweight web application to explore, analyze and visualize a broad range of sensor data. You can:

- ✓ explore stations or mobile sensor platforms in a map,
- ✓ select time series by a list selection,
- ✓ visualize time series data,
- ✓ or create favorites of selected time series.

The application is based on HTML, JavaScript and CSS and can connect to different Sensor Web endpoints (REST-APIs). These Sensor Web REST-APIs provide a thin access layer to sensor data via RESTful Web binding with different output formats.

Features:

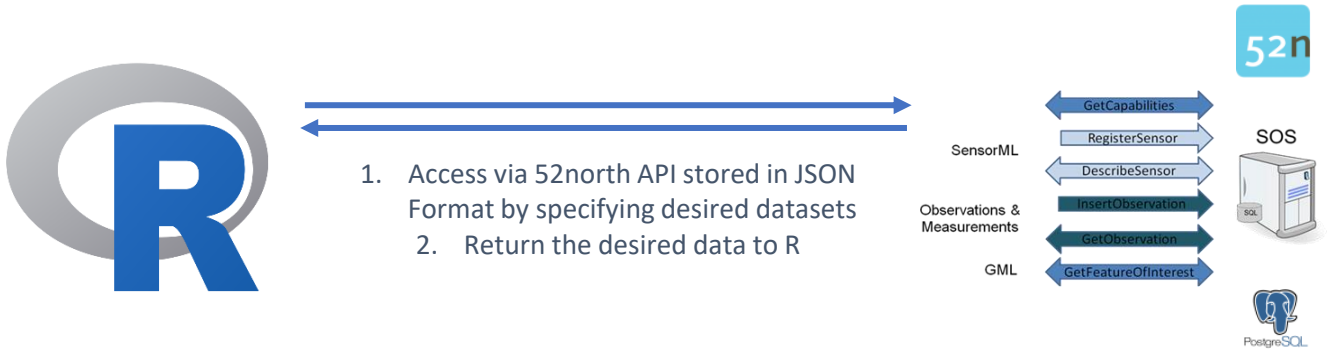
- ❑ access to SOS instances (supports OGC SOS spec...)
- ❑ diagram view of multiple time series, temporal zooming & panning...
- ❑ data export (pdf, Excel, CSV)
- ❑ Combination w/ R...
- ❑ Architectural basis: HTML, JavaScript, CSS



For more information, please visit https://github.com/52North/helgoland

MONALISA SOS_R-Access

- **Provides automated access to the MONALISA SOS Database via R**
  - *Accessing and exploring the structure of the whole SOS Database via .JSON featuring API*
  - *(Bulk) Download and storage of the data as '.csv' tables and/or '.RData' listed files*
  - *Create different plots of months and/or years selected (Work in progress)*
  - *Reproducible for other APIs featuring the JSON Format*
  - *Multiple R-Scripts stored and documented in a Git repository*
    - *Soon a central MONALISA repository will be set up*



SOS Database

Generated with the General Functions and Download Scripts

MONALISA SOS_R-Access

Access and Download MONALISA API via R

Script based:

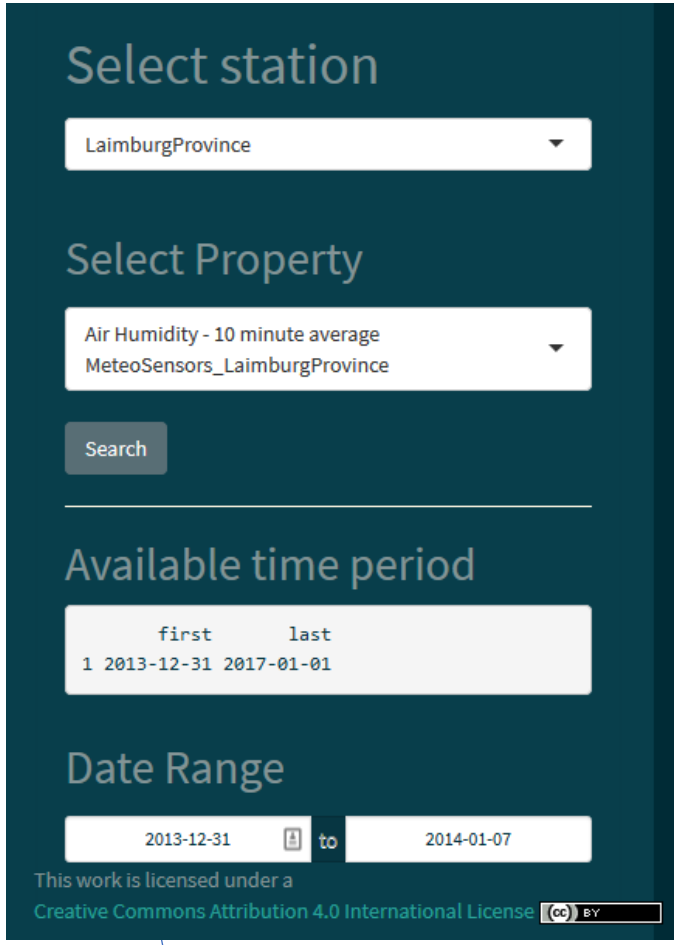
sos4Monalisa_Executable.R* | app.R | sos4Monalisa_jsonDownload.R | Old_Functions.R

Source on Save

```r
##############################################
### Program for the automatic SOS Download ####
### By Daniel Frisinghelli & Mattia Rossi  ####
### EURAC Research                         ####
##############################################
# 1 ####################################
######## INITIALIZATION ################
########################################

source("R/DependingFunctions.R")
source("R/sos4Monalisa_jsonDownload.R")

# 2 ####################################
###### MANDATORY INPUT #################
########################################

starturl = "http://monalisasos.eurac.edu/sos/api/v1/timeseries/"
datestart = "2015-12-31 00:00"
dateend = "2016-02-28 00:00"
# csv = TRUE
path = "Y:/Workspaces/FrD/2017/SOS/SOS4R"

# 3 ####################################
####### PERFORM THE DOWNLOAD ###########
########################################

SOS = SOS_download_data_json(starturl, datestart, dateend, path)
```

„sos4MONALISA_Executable.R“ Script

Web Interface based (Shiny):



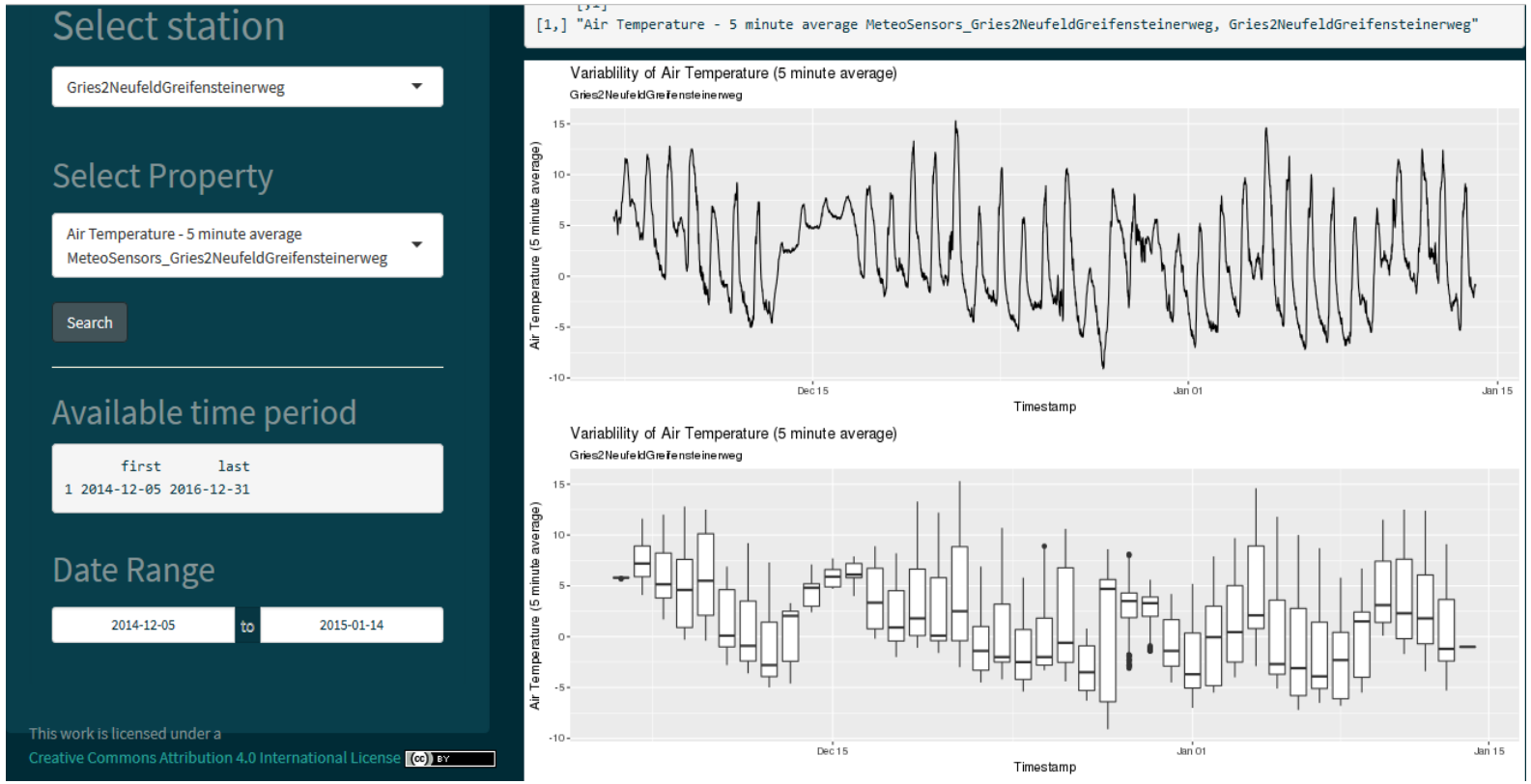
„sos4MONALISA_ShinyApp.R“ Script

MONALISA SOS_R-Access

Script based visualization:

```
# 3 ##################################
####### PERFORM THE DOWNLOAD ###########
########################################

SOS = SOS_download_data_json(starturl= starturl,
                             datestart=datestart,
                             dateend=dateend,
                             path=path,csv=F)

# 3 ##########################################
####### Iterative Plotting of the Download ###
##############################################

outdir<-"Y:/Workspaces/RosM/MONALISA/Images/"
dircheckup(outdir)

for(i in 1:length(SOS)){

  x<-SOS[[i]]
  gout<-arrange_SOS_plots(SOS_get_lineplot(DAT[[1]]),SOS_get_boxplot(DAT[[1]]))

  props<-names(x)[2] %>% str_split(.," \\(") %>% .[[1]] %>% .[1]
  fois <-x$FOI %>% unique
  ggsave(gout,filename = paste0(outdir,"ArrangePlot_",fois,"_",props,".png"), width = 10, height = 7 )
}
```

„sos4MONALISA_Executable.R“ Script



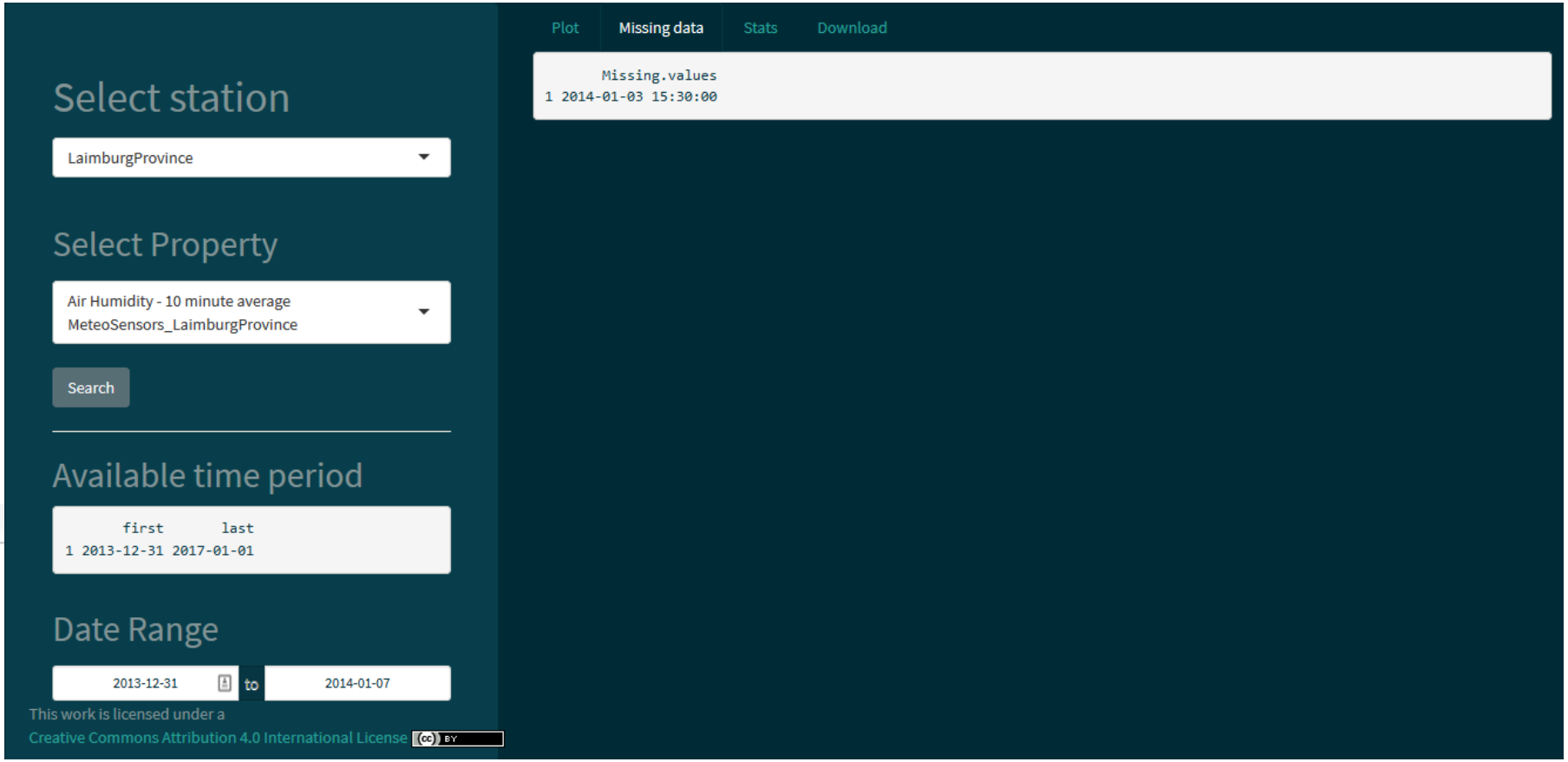
MONALISA SOS_R-Access

Shiny visualization:

- *Basically a web interface*
- *Still in initial phase but already working*
- *Will be published online once 100% stable and tested*

```r
source("R/DependingFunctions.R")
source("R/Shiny/global.R")
source("R/Shiny/ui.R")
source("R/Shiny/server.R")

shinyApp(ui = ui, server = server)
```

„sos4MONALISA_ShinyApp.R“ Script



Conclusions

An automatic system to handle the entire lifecycle of heterogeneous data has been developed combining

Data Exchange Server *52°North SOS* web-service

Considering the generic approach of this system, it can be used for any other kinds of data:

Environmental, Farming activities, renewable energy, wireless sensor network, …

Two different ways to query, use and analyze the data stored in the database have been introduced:

MONALISA SOS_R-Access *Helgoland*

Work in progress

- A GIT Repository will be soon available where the python script and the SOS_R-Access plugin will be available together with a user manual and data to be used for testing the system
- A user-friendly web interface will be developed to help the data provider/user to create the static file
- The exploitation about how to combine Shiny with the entire system will be further implemented and investigated

eurac research

Thank you for your attention

http://monalisasos.eurac.edu/sos/

bartolomeo.ventura@eurac.edu
support.rs@eurac.edu