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
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

- acquire
- harmonize
- share
- analyze
- organize
Test case: MONALISA project

The MONALISA (MONitoring key environmental parameters in the ALpine environment Involving Science, technology and Application) 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/](http://monalisasos.eurac.edu/sos/)
Data Acquisition: 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.
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.
In this particular scenario dedicated plug-ins, for example, implement:

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

- **acquire**
  - Collect data:
    - experiment
    - observe
    - measure
    - simulate
  - check data
  - validate data
  - clean data

- **share**
  - disseminate results
  - distribute data
  - control access
  - establish copyright
  - promote data

- **harmonize**
  - locate existing data
  - define data format
  - organize folders structure
  - manage
  - store data
  - create metadata

- **organize**
  - create metadata

- **analyze**
  - interpret data
  - describe data
  - derive data
  - produce research outputs
  - author publications
  - prepare data for preservation

---

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

Different sensors and stations
Different data organization with respect to the same sensor
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
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**

**Diagram:**
- Data provider/User
- Data manager
- Creation of the static file
- Standardization of the input data file
- XSLT code
- Insert sensor
- Insert Observation template
- SOS
- Code for the insertion of the sensor and observation

**Flow:**
1. Creation of the static file
2. Standardization of the input data file
3. Insert sensor
4. Insert Observation template
5. Code for the insertion of the sensor and observation

**Tools:**
- 52n
- PostgreSQL
• 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>

<observedProperty>
  <short_name>/short_name>
  <long_name>/long_name>
  <long_name_def>/long_name_def>
  <nom>/nom>
  <id_sensor_num>1</id_sensor_num>
</observedProperty>

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

```xml
<xml version="1.0" encoding="UTF-8"?
<static>
  <FOI_list>Tramin3der, Unterrain2FuchsangerEppan,
             Gries2NeufeldGreifensteinerweg, Girlan1Laum, Las6</FOI_list>
  <FOI_coordinates_list>46.3291770 11.2445070
                         46.016200
                         11.2564780, 46.5017570
                         11.2802330, 46.458950
                         11.2795330, 46.6165530
                         11.1517880</FOI_coordinates_list>

<observedProperty>
  <short_name>NDVI_UpRed_Avg</short_name>
  <long_name>Normalized_Differenced_Vegetation_Index_UpRed</long_name>
  <long_name_def>Normalized_Differenced_Vegetation_Index_UpRed_Avg</long_name_def>
  <nom>None</nom>
  <id_sensor_num>1</id_sensor_num>
</observedProperty>

<coordinate>
  <easting>11.2445070</easting>
  <northing>46.3291770</northing>
  <altitude>239</altitude>
</coordinate>
```
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 i\textsuperscript{th} static file
- Fill the *xml of the insert observation file inserting the values and post it to the SOS Database
XSLT definition and goals

Definition:

• XSL stands for: **Extensible Stylesheet Language**, and it is a stylesheet language for XML documents.
• XSLT stands for: **XSL Transformations**

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

Search and Discover sensors and their data information
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
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](https://github.com/52North/helgoland)

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

1. Access via 52north API stored in JSON Format by specifying desired datasets
2. Return the desired data to R

Generated with the General Functions and Download Scripts

SOS Database

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

Script based:

```
# Script based:
# sos4MONALISA_Executable.R

# Web Interface based (Shiny):

# sos4MONALISA_ShinyApp.R
```
Script based visualization:

```
# PERFORM THE DOWNLOAD
# Iterative Plotting of the Download

# Script based visualization

Select station
- GriessNeufeldGreifensteinweg

Select Property
- Air Temperature - 5 minute average

Available time period
- first: 2014-12-05
- last: 2015-12-31

Date Range
- 2014-12-05 to 2015-05-14
```

"Sos4MONALISA_Executable.R" Script
Shiny visualization:

- Basically a web interface
- Still in initial phase but already working
- Will be published online once 100% stable and tested
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*
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.
Thank you for your attention

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

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