An OGC/SOS Conformant Client to Manage Geospatial Data on the GRID

CROSS-Fire Project

António Esteves, Marco Caldas,
António Pina, Alberto Proença

University of Minho

FCT
Fundação para a Ciência e a Tecnologia

IBERGRID'2010

Braga :: 26th May 2010

AGENDA

- Cross-Fire project and Firestation application
- Identification of the present task goals
- SWE standards (SOS in more detail)
- Weather station used to test the SOS
- Application that populates the SOS database
- Case study: a WPS algorithm (SOS client) that accesses geo-referenced spatial data
- Conclusions and Future work

Cross-Fire Project

CROSS-Fire : Collaborative Resources Online to Support Simulations on Forest Fires

Support:

- Portuguese NGI (INGRID):
  - FCT grant GRID/GRI/81795/2006
- JRU Portugal:
  - EELA2: **E**-science grid facility for **E**urope and **L**atin **A**merica. FP 7, INFRA-2007-1.2.3

Project Goals

- To scale from the desktop towards a service-oriented information system.
- To benefit from Grid infrastructure.
- To provide decision-makers with a persistent set of independent high-level services.
- To share geospatial information.

- Main case study: **forest fires**.
- Fire spread simulation application: **Firestation**.

esteves@di.uminho.pt

CROSS-Fire Platform

- A core WPS layer based on 52North implementation
  - Business Logic
    - to handle the Firestation algorithms
  - Grid Services
    - to interface with the GRID infra-structure
  - Geospatial Services
    - to interface between clients and the SDIs → a collection of Web services that run as WPS algorithms
- Two external infrastructures:
  - SDI platform
  - GRID

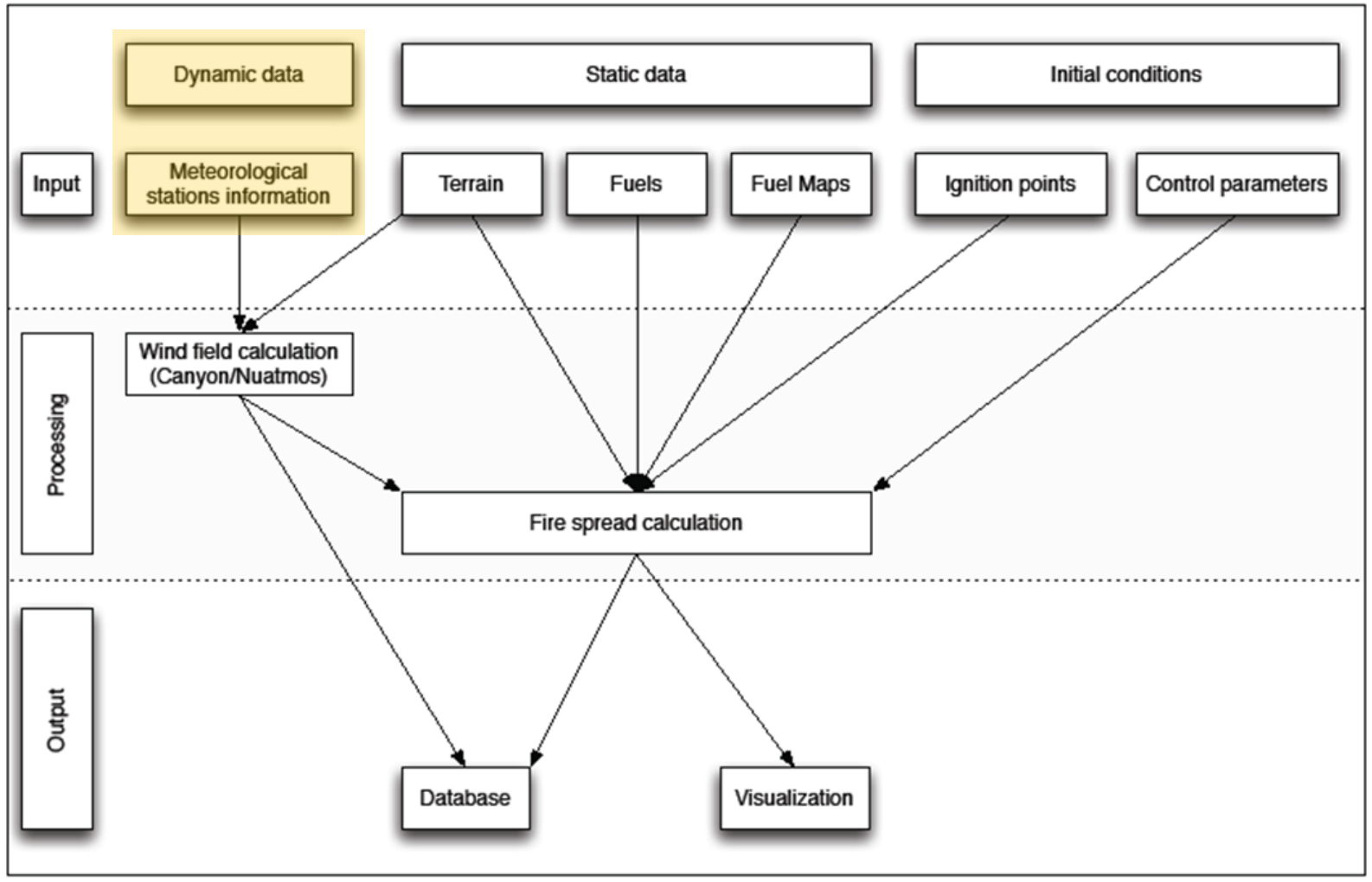
esteves@di.uminho.pt

Firestation Application

- An integrated system
  - Three modules to compute:
    - Wind field: Canyon and Nuatmos models
      - Navier-Stokes solver and analytic solution
    - Fire Weather Index of the Canadian system
    - Fire propagation over a complex topography:
      - topography (fuel and altitude), wind conditions, control parameters
- Is being ported to operate on the Grid: **G-Firestation**
  - GeoServer-based **SDI layer** to exploit geospatial services for data access and processing.
  - 52North-based **OGC/SWE conformant layer** to access CP relevant data sources.
  - A **graphical user interface** to access the platform facilities.

esteves@di.uminho.pt

Firestation Application



esteves@di.uminho.pt

Present Task Goals

- To integrate dynamic geospatial data from meteorological (in-situ and satellite) sensors on the CROSS-Fire grid-based risk management decision support system.
- To use standard services and data encodings:
  - From OGC **Sensor Web Enablement** (**SWE**), **WCS**, ...
    - SensorML, TransducerML, O&M.
    - SOS, SPS, WNS, SAS/SES.
  - Based on 52North implementation.
  - To describe, register, and discover sensors.
  - To store and access sensor observations.

esteves@di.uminho.pt

SWE Standards

- **SensorML** → a functional model of the sensor system, rather than a detailed description of its hardware.
- **Observation & Measurements** (**O&M**) → a framework, a conceptual model and an encoding formalized as an application scheme.
- **Sensor Planning Service** (**SPS**) → an interface to task a sensor system.
- **Sensor Alert Service** (**SAS**) → an interface for a web service to publish and subscribe alerts from sensors.
- **Web Notification Service** (**WNS**) → an interface for a service to interchange asynchronous messages with other service(s).
- **Sensor Observation Service** (**SOS**)

Sensor Observation Service

- Defines a standard web interface to request, filter, and get observations and metadata from sensors.
- Operations:
  - **getCapabilities** → a mandatory operation used to request metadata about the potentialities of the SOS service.
  - **describeSensor** → a mandatory operation used to access information about a sensor, returning a SensorML or TML document. A SensorML document specifies (at least) the location of the sensor and the phenomena it monitors.
  - **getObservation** → a mandatory operation used to request observations, encoded in O&M. A request, may include the offering we are interested in, a time filter, the procedure, the observed property, the feature of interest, the result, the result model, and the type of answer.

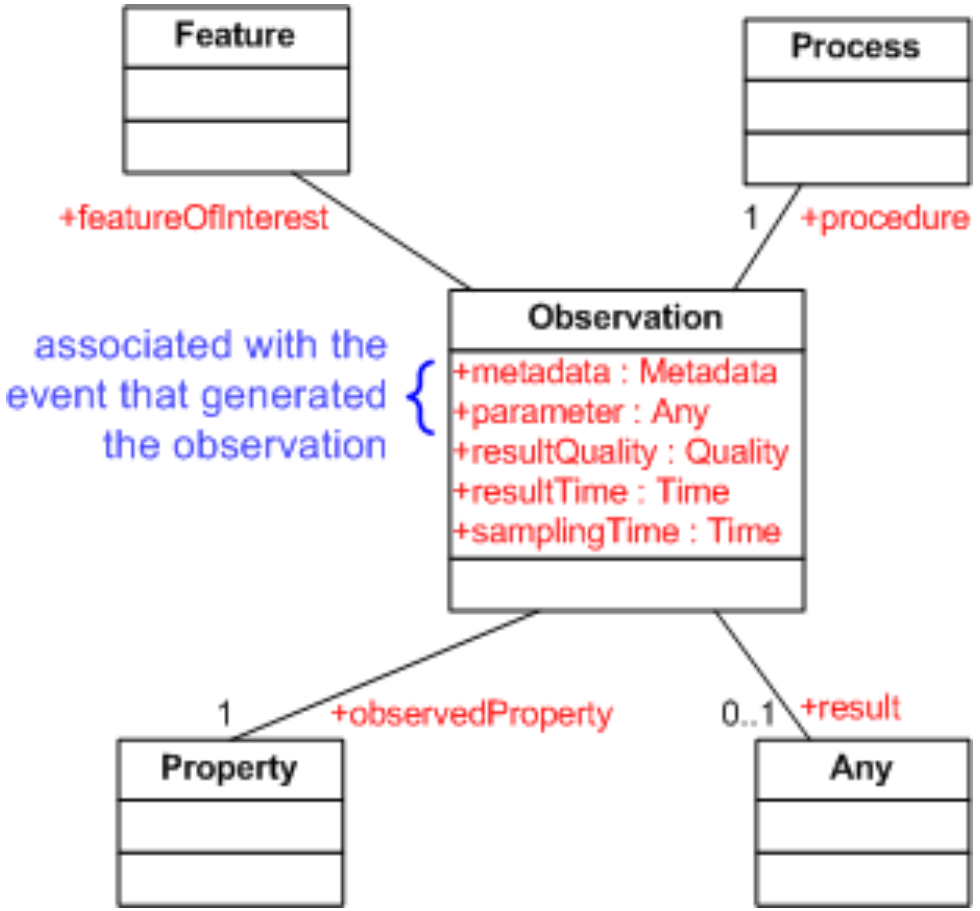
esteves@di.uminho.pt

Sensor Observation Service

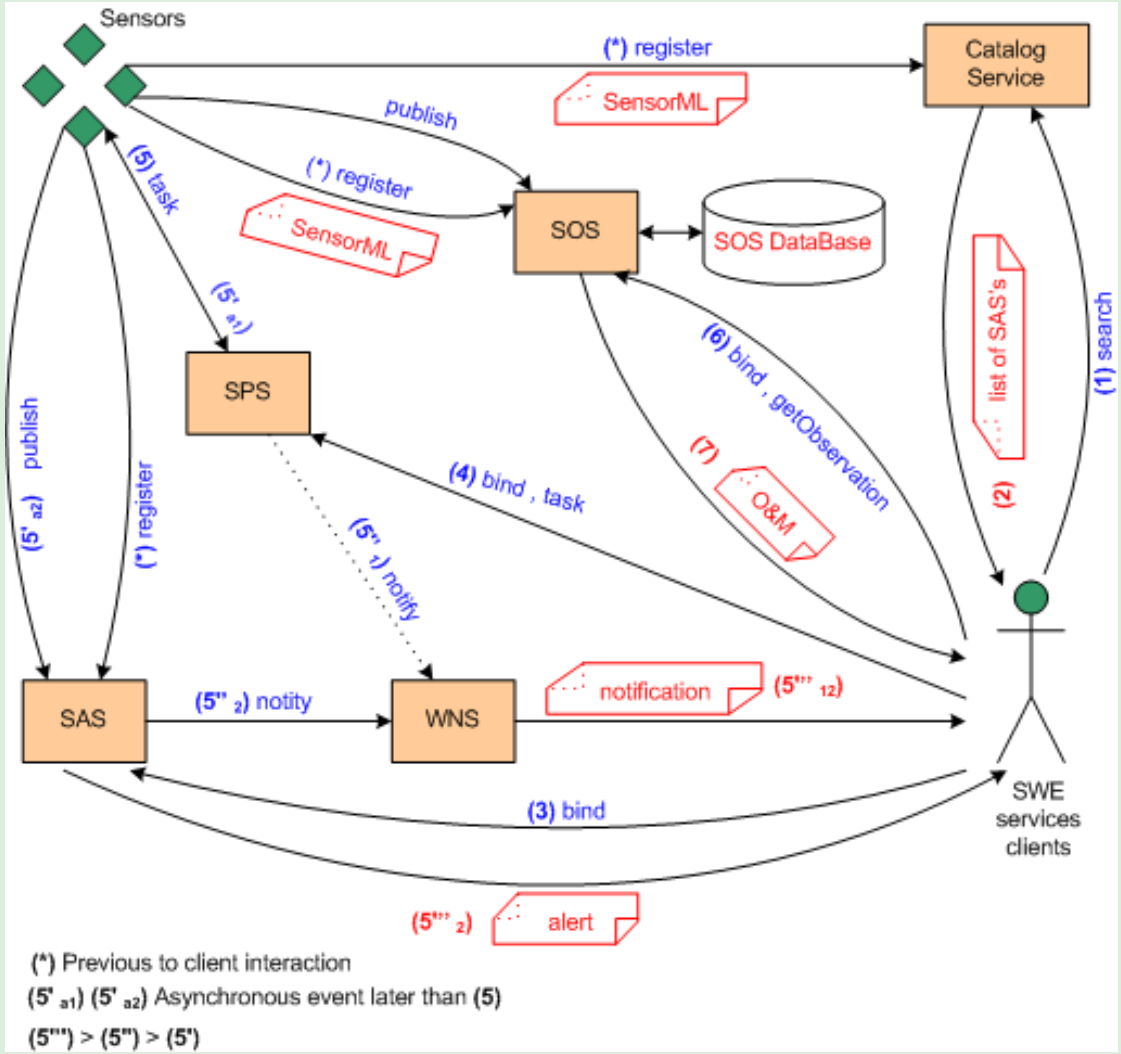
- Operations:
  - **registerSensor** → an optional operation for a client to register a sensor on SOS. The client can only insert observations belonging to a sensor already registered with the SOS.
  - **insertObservation** → operation for a client to insert observations in the system associated with a sensor.

Sensor Observation Service

- Simplified structure of an **O&M** observation:



SWE Standards Interaction Scenario



Weather Station (WS) Used to Test SOS

- Davis VantagePro2
  - A base station
  - A sensor suite:
    - rain collector
    - temperature sensor
    - humidity sensor
    - anemometer (wind speed and direction)
    - barometer
    - …
  - These are the relevant sensors for Firestation.

Application that Populates the SOS DB

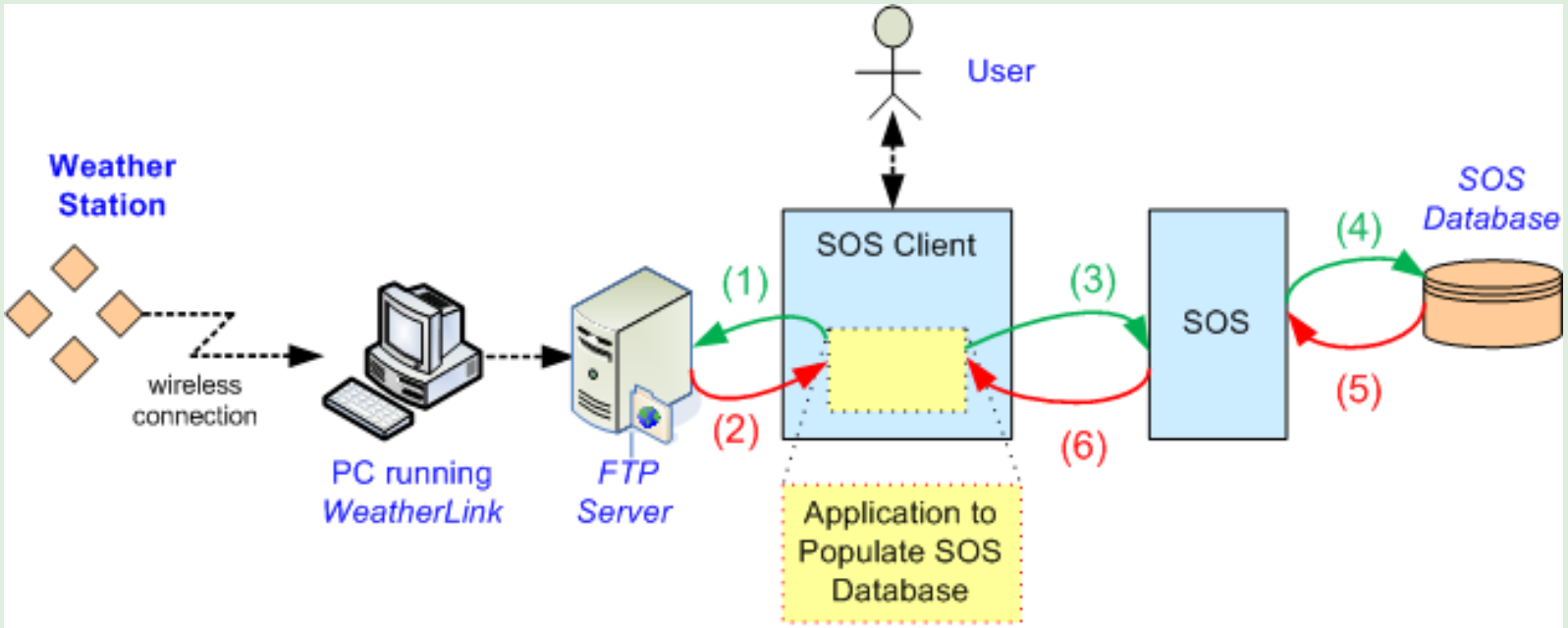
- The remote Vantis WS is connected to an FTP server.
- To retrieve the meteorological data from that WS it was developed a specific application.
- The retrieved data is processed and inserted in the SOS database.
  - It was necessary to implement a client version of the SOS **insertObservation** operation, and
  - To have registered previously the sensors on SOS.
- It was used a PostgreSQL database.
- The application was implemented in JAVA.

esteves@di.uminho.pt

Application that Populates the SOS DB

- To be self-configurable, the application reads a configuration file at start up. The configuration defines values for the following parameters:
  - The FTP server: IP, username and password.
  - The files to download.
  - The application mode of operation: frequency of access to the FTP server, if backup downloaded file or not, the files' storage place.
  - A section with the list of tags to be processed and inserted into the SOS database.
- The application consists of 3 parsers:
  - For the configuration file.
  - For the file generated by the weather station.
  - For the SOS answer.

Application that Populates the SOS DB



Dataflow of application operation:

(1) FTP get command
(2) file with latest observation
(3) insertObservation request
(4) execute insertObservation in DB
(5) result (success, failure)
(6) result (success, failure)

esteves@di.uminho.pt

Case Study: a WPS Algorithm that Accesses Geo-referenced Spatial Data

- This is an **SOS client** implementation to mediate the interaction between WPS and SOS.
- **WPS** is a OGC standard used to make calculations in a standard way through the Internet.
- WPS mandatory operations are:
  - **getCapabilites**
  - **DescribeProccess**
  - **Execute**
- The facilities necessary for the WPS algorithm are implemented as a **GetObservation** class.

esteves@di.uminho.pt

Case Study: a WPS Algorithm that Accesses Geo-referenced Spatial Data

- To implement the **GetObservation** class it was necessary to develop other classes:
  - **CreateDoc** → It is responsible for creating the XML document to be sent as a request to the SOS.
  - **GetHTTP** → It allows us to send requests to the SOS and receive the answer.
  - **ObservationOffering** → It is a support data class to store the data that will be returned by the getStations method from GetObservation class.
- The **GetObservation** class provides 2 methods:
  - **getStations** → lets WPS to know which weather stations are located in a given region.
  - **getData** → allows WPS to consult the SOS database.

esteves@di.uminho.pt

Case Study: a WPS Algorithm that Accesses Geo-referenced Spatial Data

- **getStations** flowchart:

```
getStations ()
    {
    Send a getCapabilities() request to the SOS;
    Process the Capabilities doc returned by the SOS;
    Apply a spatial filter to the processed data;
    Store data about stations on ObservationOffering
      class;
    Return ObservationOffering to WPS;
    }
```


esteves@di.uminho.pt

Case Study: a WPS Algorithm that Accesses Geo-referenced Spatial Data

- **getData** flowchart:

```
getData ()
    {
    createDoc();
    Fill the XML doc with the request data;
    The XML doc is sent to the SOS by GetHTTP;
    The request result is parsed and placed on HashMap;
    The HashMap is returned to the WPS;
    }
```

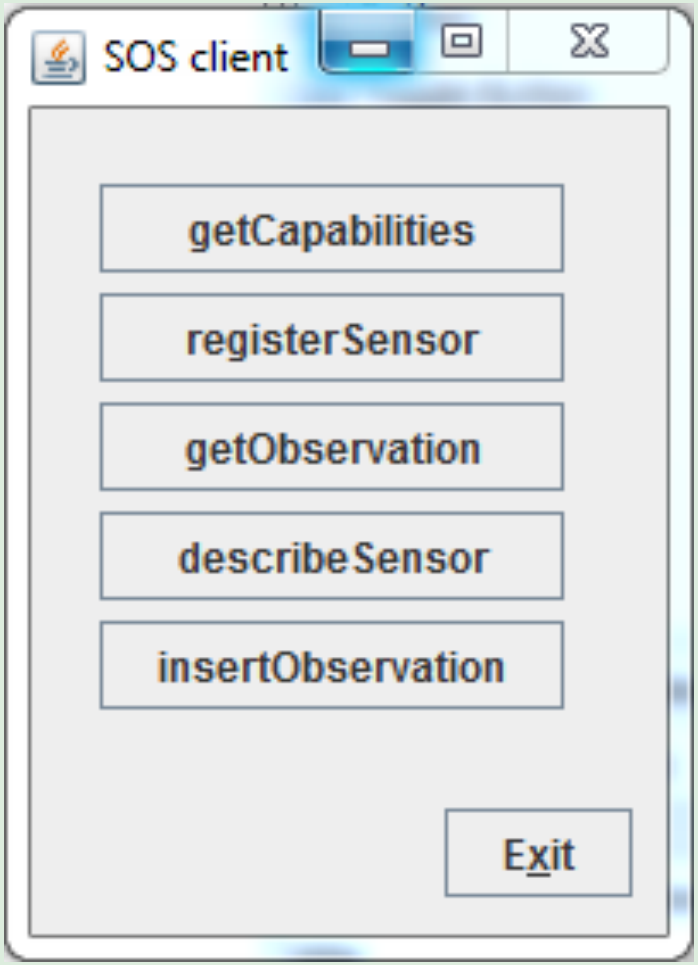

esteves@di.uminho.pt

Case Study: a WPS Algorithm that Accesses Geo-referenced Spatial Data

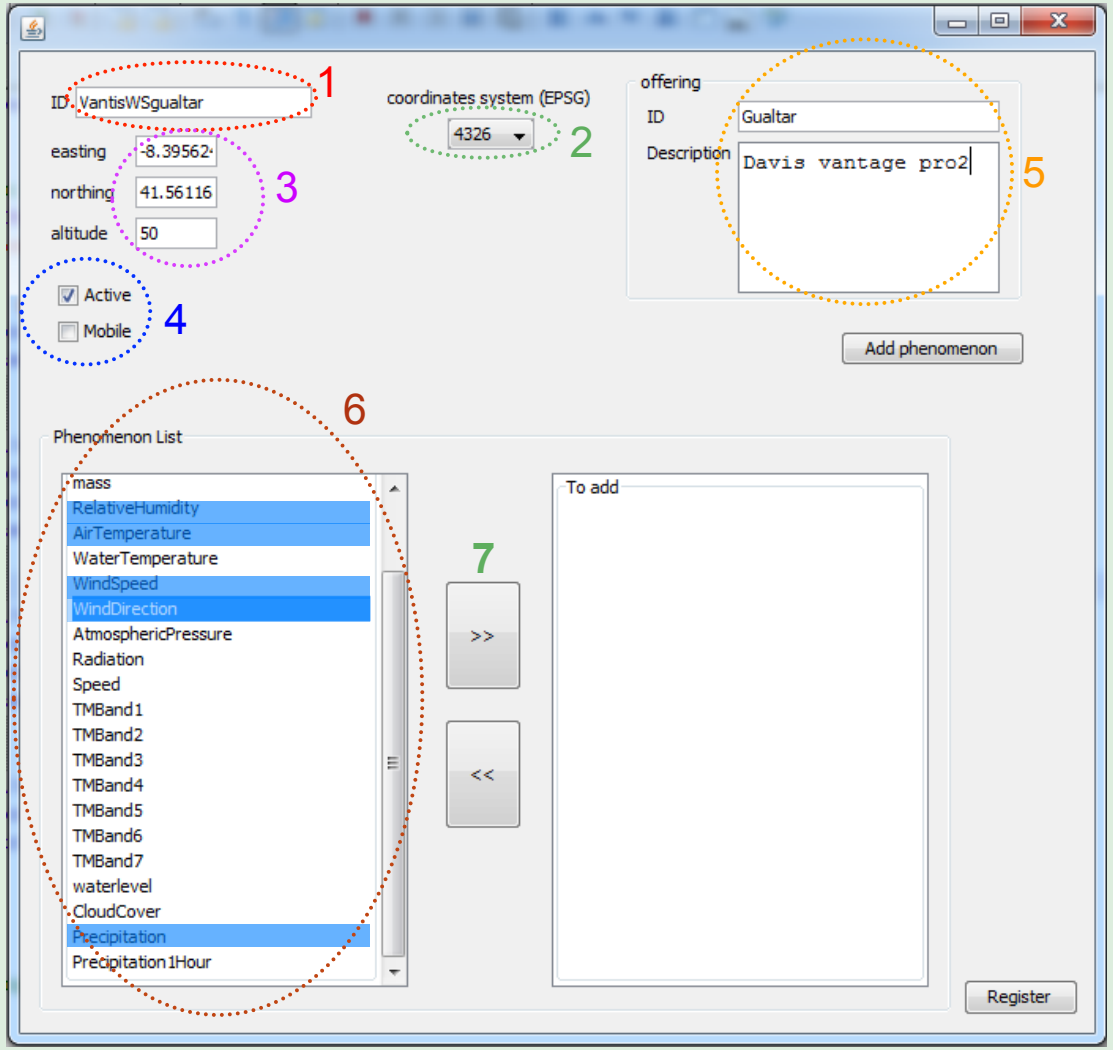
- For validation purposes, we decided to implement a graphical user interface (GUI) to the WPS algorithm (SOS client).

esteves@di.uminho.pt

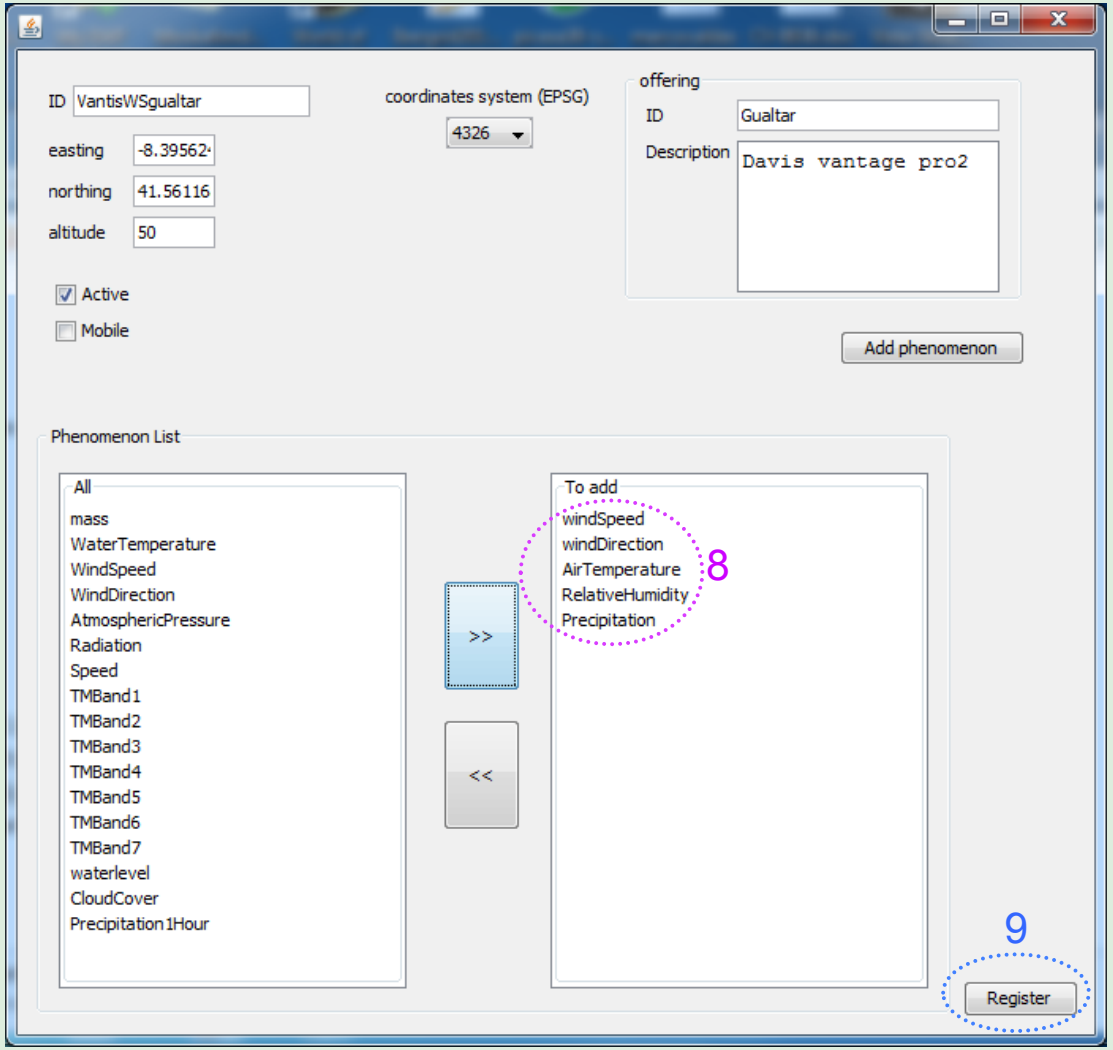
SOS client: Operations



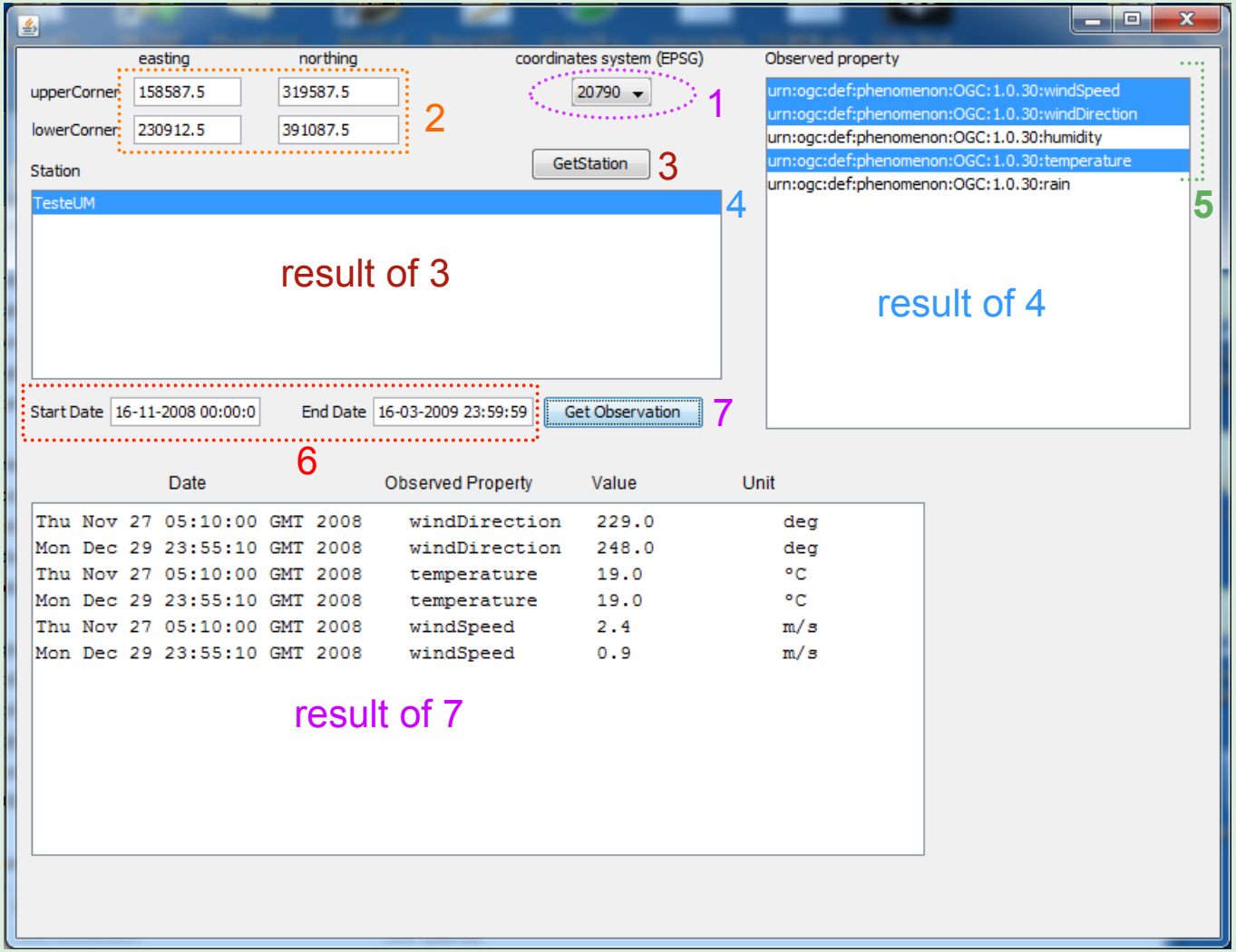
SOS client: registerSensor (1)



SOS client: registerSensor (2)

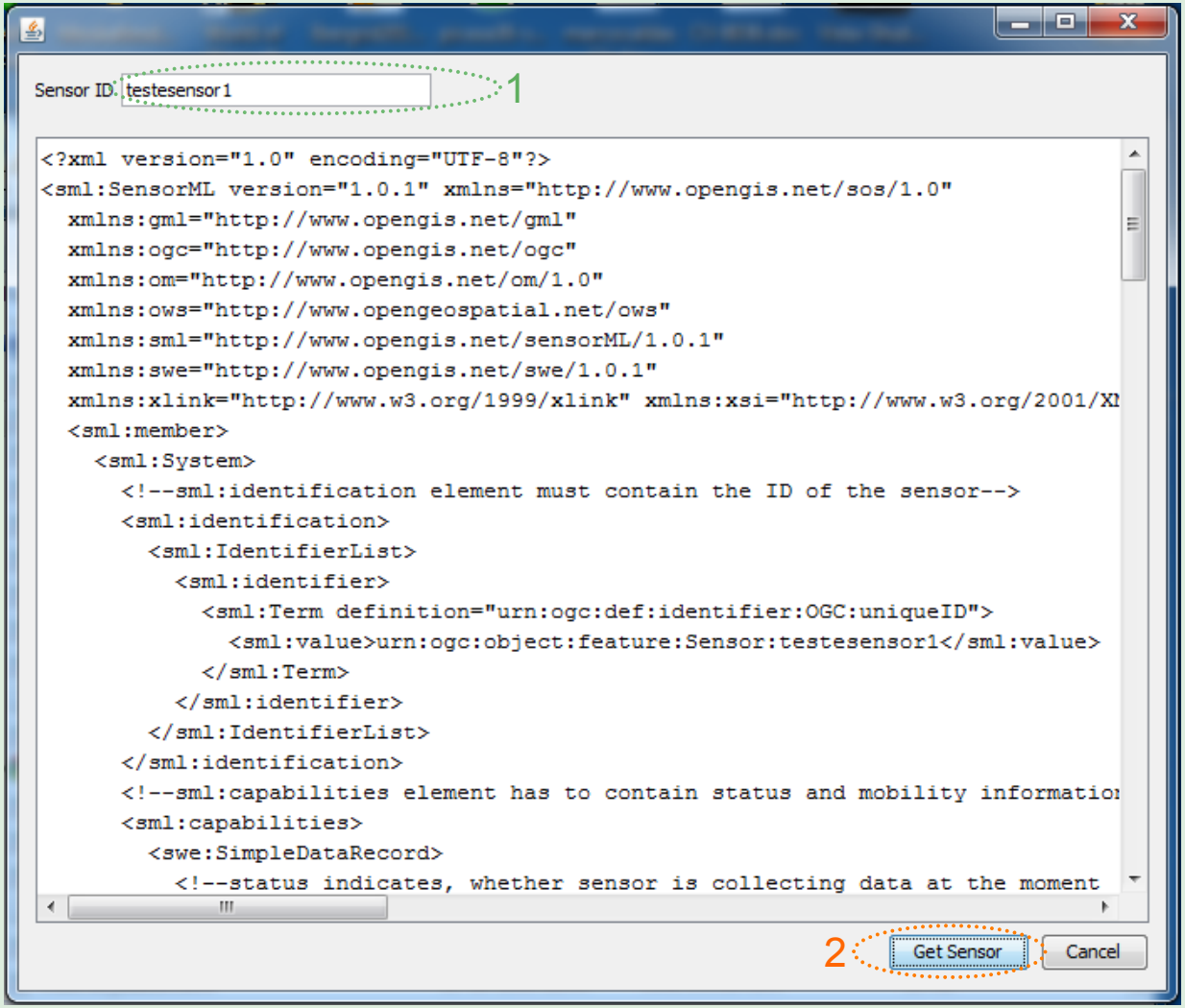


SOS client: getObservation



SOS client: describeSensor

3 SensorML description of ***testesensor1***



Conclusions

- The main reason to implement a new SOS client, was the necessity of interacting with WPS.
- The developed SOS client is functional and fully compliant with SOS, SensorML, and O&M standards from OGC.
- With a slight modification of the application that populates the SOS database, the implemented client works with any weather station.
- The client supports spatial and temporal filters.

Future Work

- We are working on the integration of other types of spatial data (such as satellite images) on CROSS-Fire.
- Several projects, satellites, and instruments were evaluated.
- We consider the utilization of MODIS instrument.
- The data we are interested in is:
  - land coverage (vegetation)
  - burned areas.
- The information will be provided as **coverages** by a **WCPS** service.

Thank You

for

your attention

QUESTIONS?