ISTSOS, SENSOR OBSERVATION MANAGEMENT SYSTEM: A REAL CASE APPLICATION OF HYDRO-METEOROLOGICAL DATA FOR FLOOD PROTECTION.

M. Cannata [a] *, M. Antonovic [a], M. Molinari [a], M. Pozzoni [a]

[a] Institute of Earth Sciences, University of Applied Sciences and Arts of Southern Switzerland (SUPSI), Campus Trevano, CH - 6952 Canobbio, Switzerland- (massimiliano.cannata, milan.antonovic, monia.molinari, maurizio.pozzoni)@supsi.ch

WG I/2, WG I/6, ICWG I/V, ICWG V/3, ICWG V/5

KEY WORDS: SOS, sensors, observations, services, OGC, monitoring

ABSTRACT:

istSOS (Istituto scienze della Terra Sensor Observation Service) is an implementation of the Sensor Observation Service standard from Open Geospatial Consortium (OGC). The development of istSOS started in 2009 in order to provide a simple implementation of the Sensor Observation Service (SOS) standard for the management, provision and integration of hydro-meteorological data collected in Canton Ticino (Southern Switzerland). istSOS is entirely written in Python and is based on reliable open source software like PostgreSQL/PostGIS and Apache/mod_wsgi. The authors during this presentation want to illustrate the latest software enhancements together with a real case in a production environment. Latest software enhancement includes the development of a RESTful service and of a Web-based graphical user interface that allows hydrologists a better interaction with measurements. This includes the ability of new services creation, addition of new sensors and relative metadata, visualization and manipulation of stored observations, registration of new measures and setting of system properties like observable properties and data quality codes. The study will show a real case application of the system for the provision of data to interregional partners and to a hydrological model for lake level forecasting and flooding hazard assessment. The hydrological model uses a combination of WPS (Web Processing Service) and SOS for the generation of model input data. This system is linked with a dedicated geo-portal used by the civil protection for the management, alert and protection of population and assets of the Locarno area (Verbano Lake flooding). Practical considerations and technical issues will be presented and discussed.

1. INTRODUCTION

1.1 General context

In the recent years there has been a tangible effort in the political, scientific and technological domains aiming at establishing data interoperability. This on the basis of the commonly accepted assumption that interoperability allows to integrate information coming from different sources and to reuse data collected with different purposes (Pirotti et al., 2011). It also leads to more accurate and comprehensive information, eliminating duplication and harmonization costs, thus allows better understanding and ultimately wiser decision-making.

In the geospatial sector, the above mentioned effort resulted in the establishment of a series of legislative instruments at different levels that push/force data producers to follow a series of standards. For example the INSPIRE Directive (Directive 2007/2/EC) at European level or the Swiss Federal Act on Geoinformation (Geoinformation Act, GeoIG) (RS 510.62) at national level can be cited. Following these frameworks, the scientific world also started to develop a large number of research projects that studied new solutions, tested proposed standards (Bleier et al., 2009; ORCHESTRA Consortium, 2010; TRIDEC Consortium, 2012) and developed new software such as Geonetwork, Geoserver, pyCSW and ZOO.

Up to now, most of the these actions/researches have largely targeted typical Geographical Information System (GIS) layers, such as land-use, water protection zones, risk mapping, elevation models, hydrological network, etc. As a result, several services offering these types of data according to defined standards (WMS, WFS, WCS, CWS) are currently implemented and used in productive environment.

In a context of risk analysis and early warning system not only the base and contextual data are extremely important, but also historical and real-time records of environmental variables are crucial to correctly model, estimate and forecast impending hazards and expected impacts. Therefore, the accessibility to monitoring networks data without the need of tedious, error-prone and costly operations of data integration is certainly a goal to be pursued.

In this context, a series of initiatives like the intergovernmental Group on Earth Observations (GEO), the European programme Copernicus and the Open Geospatial Consortium Sensor Web Enablement Working Group (OGC-SWE) have been launched. Among others, some of the results of these initiatives are: case studies, IT architecture best practices, Web services and the definition of a series of standards.

1.2 The risk management system for the Verbano Lake flooding

The Institute of Earth Sciences (Istituto scienze della Terra, IST) has managed the hydro-meteorological monitoring network of the Canton Ticino since the end of the ‘70s. Its hydrometric stations are a complement of the federal network on major rivers, while its rain gauge stations are an integration of the MeteoSwiss meteorological monitoring network.

The data quality is assured by an automatic and manual check and by a frequent inspection of the sensors. Since 1979, IST has

This contribution has been peer-reviewed.
doi:10.5194/isprsarchives-XL-5-W3-111-2013

published the Hydrological Yearbook of Canton Ticino, with statistical analysis of data and some further insights.
In addition to scientific analyses and general land management purposes, the collected data are used for supporting a system for the management of flooding risk in the Locarno area. In fact, the watersides of the Verbano Lake are particularly prone to flood hazard due to the large catchment size (6599 km^2) compared to the small lake surface (212 km^2) and the limited outlet capacity.
The occurred flood events were generally concentrated in time but severe in magnitudo, especially in the fall season. For this reason in the 1997, the IST, MeteoSwiss, the Locarno e Vallemaggia civil protection agency, the Canton Ticino Land Department and other cantonal authorities established a specific working group for risk reduction. In this framework, the IST, between the 1999 and the 2002 set up an Early Warning System for the Verbano Lake that is still running today.

1.3 Technological change

In recent years, due to the evolution of technology, the network underwent a complete renewal, with new instruments and new data transmission protocols. Sensors moved, from being a passive device, capable of providing data only on demand (Analogic modems), to being an active instrument (GPRS modems), always connected and capable of pushing information in real-time.
This hardware adaptation led to the need for renewing also the information technology part of the system, including the data storage, the quality assessment and data validation, and the model input elaboration processes.
Considering the general context that pursues the data interoperability and the need for system upgrade, in 2009 the IST started to look with interest at the Sensor Observation Service standard (SOS) (OGC, 2007). Unfortunately, the requirements identified by experienced IST's hydrologists such as: supporting of irregular time series, robust quality control mechanism, data aggregation capability and observation correction feature were not available on existing SOS products (Cannata and Antonovic, 2010). Therefore, the IST decided to develop its own software implementation of the standard: istSOS. In 2010, the software became operational. In the subsequent years, a series of further needs were identified and istSOS evolved from a geo-service to a management system.
This paper focus the attention on the newly developed *"sensor observation management system"* based on the SOS. In the next section the authors describe the software components and architecture while in the third section its application in a real case is presented and evaluated. Finally, some conclusions are presented.

2. ISTSOS: A NEW SENSOR OBSERVATION MANAGEMENT SYSTEM

istSOS is a sensor observation data management system that allows to manage and dispatch observations collected from monitoring sensors. The system follows an interoperable approach, thanks to the support of the SOS version 1.0 specifications.
The SOS standard, as illustrated in Figure 1, is based on five key elements. The *observations* are the data that have been registered: they are related to a single *procedure* that is the sensor, device or process that performed the observations (i.e. a rain gauge). *Observations* are referred to one or more *observedProperty*, which identify the observed phenomenon (i.e. the rainfall), and to a *featureOfInterest*, which indicates the observed target (i.e. a medium like the Verbano Lake with or without geospatial representation). Connected triplets of *procedure-observedProperties-featureOfInterest* are then associated with one or more *offering*, which indicate a logical group of sensors (i.e. Ticino meteorological network).

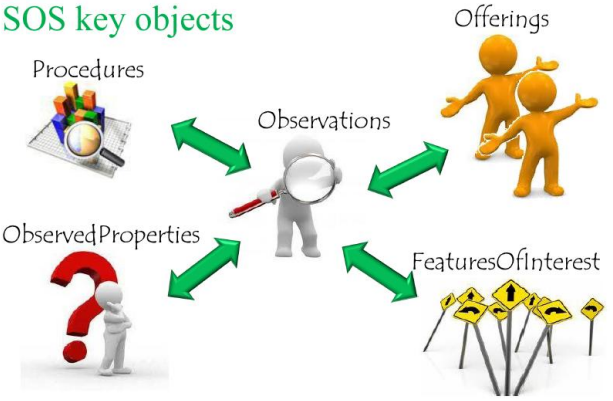

Figure 1. Sensor Observation Service key objects

The istSOS software is composed by two different services and a Web-based interface: *istsos* service exposes information according to the SOS standard, *wa* service exposes management features on the Web and *wainterface* permits the easy interaction with the two services.
The server side of the application (*istsos* and *wa* services) is entirely written in *Python* (Van Rossum, 2003) and relies on the *Apache HTTP server* (The Apache Software Foundation, 2013) enhanced by the module *mod_wsgi* (Dumpleton, 2013), which allows for hosting any Python application which supports the Python WSGI interface (Phillip, 2012). Moreover, it uses *PostgreSQL* (Worsley and Drake, 2002) with the *PostGIS* (Obe and Hsu, 2011) spatial extension as database for data storage, access and elaboration. Other Python packages from which istSOS depends on are *isodate* (http://pypi.python.org/pypi/isodate/0.4.0) for ISO8601 standard conversions, *psycopg2* (http://packages.python.org/psycopg2) for database connections, *pytz* (http://pytz.sourceforge.net) for time zone management, and *requests* (http://docs.python-requests.org) for HTTP interaction with other services.
The client side of the application is completely written in *Javascript* and takes advantage of the *ExtJS* (Sencha, 2013) library, the *dygraph* (Vanderkam, 2008) library and *CodeMirror 2* (http://codemirror.net).
This system has a nested architecture where *istsoslib* is the kernel of the system that enables SOS provision, *walib* is the pulp that implements the classes and functions required by the *wa* service and, finally, *wainterface* is the peel that wraps the other components in a user-friendly interface.
The next sections will detail the technologies, architectures and features for each component.

2.1 istsoslib

istsoslib is the core part of the system that enables the SOS standard provision through the *istsos* service. It has been developed following an abstract factory pattern approach that allows the instantiation of specialized classes or methods specifically built to execute a particular required task. The library implements three module types: *filter*, *responder* and *renderer*. When a request is submitted to the service, the

This contribution has been peer-reviewed.
doi:10.5194/isprsarchives-XL-5-W3-111-2013

modules are sequentially instantiated in this order: *filter*, *responder* and *renderer*.
Each module, when instantiated, calls a factory function that, depending on the current request, initializes the specialized object able to execute the desired task. The *filter* is responsible for verifying and converting the HTTP request (GET or POST) in Python object, the *responder* takes in charge the processing operations – information gathering and elaboration or transactional operations execution – and finally the *renderer* is devoted to convert the request results in a valid SOS response.
In addition to standard functionalities, as defined in the SOS documentation, *istsoslib* implements a series of extending features (not defined in the specifications) designed to support IST data management needs. These enhancements are:

1. Virtual procedures: capability to offer data that are not actually measured but are the result of an elaboration of other data belonging to other procedures. Typical example is the provision of river discharges that are computed from water depth observations using a rating curve.
2. Irregular time series: event-driven observation can provide empty measurements that correspond to no-event; this should be distinguished from data unavailability (missing observation on event occurrence).
3. On the fly aggregation: user can request aggregated data of provided observed properties by specifying an aggregation time resolution, an aggregation mode (maximum, minimum, average or sum) and a *no data* value for representing interval with no observations.
4. Different output formats: when observations are requested, the user can specify the desired output format; in addition to the XML format required by specifications, *istsos* supports application/json and plain/text formats.
5. Insert many at once: the specifications don't clearly define if more than one observation can be registered with a single request named *insertObservation* (note the singular); *istsos* support the registration of a series of observations at once.
6. Observation overriding: sometimes the data manager has the need to delete or replace existing observations. A typical example is the cancellation of test data recorded during instrument verification (e.g. empty a bucket of water in a rain gauge to verify the correctness of the registered water volume). Another typical example is the updating of observations with new records: when data are collected from other sources, they are often provided at regular intervals with overlapping periods. If some communication problems happen the following provision integrates erroneous or missing data.
7. Maximum data period per request: the system allows to set a maximum time period for a single *getObservation* request; this prevents server overload and service unavailability.
8. LIKE wise support in observed properties filtering: observed properties are generally in a hierarchic relation each other's according to a well-defined ontology. For example, temperature and humidity are both meteorological properties of the air so they are likely registered as *urn:ogc:def:property:x-istsos:1.0:meteorological:air:temperature* and *urn:ogc:def:property:x-istsos:1.0:meteorological:air:humidity.* User can request for *air* observed property and both are returned.
9. Data quality index: istsos supports natively the data quality index as an indispensable property of the data. In fact, data has no, or very poor, value if no information regarding their quality is provided. Each observation and, if mobile, each position of the instrument, is associated with a data quality index. User has the ability to define a series of quality index levels, set a default value (generally associated with raw data), update the indexes, and finally request data with an explicated quality index.

For further details on this component, interested readers can refer to Cannata and Antonovic (2010) and to istSOS documentation.

2.2 walib

walib is the pulp of the system that primarily enables the *wa* service and, through it, the setting and configuration of istSOS using a "REpresentational State Transfer" (RESTful) (Richardson and Ruby, 2007) approach. This newly developed component allows accessing the elements by using the HTTP methods: GET to retrieve, POST to add, PUT to update and DELETE to remove objects. In accordance with the RESTful paradigm, Figure 2 shows the exposed resources identified in a directory-structure-like URIs with information on the supported methods.
walib, supports the *json* format (http://www.json.org) for the representation of the content sent to the server, and the response provided to the client. In general, the GET method just requires the element specified in the URI; the POST requires a number of parameters, that are the same of the result of a GET operation on the same element, but without need of specifying the object identifier; the PUT requires the same parameters of the POST with the object identifier explicitly provided; and finally the DELETE operation requires the element identifier specified in the URI.
istsoslib allows for defining multiple *istsos* service instances, each one with its specific configuration that extends a system default configuration. Each service is identified by a specific unique name, which is also the name of the database schema where data are stored.
A generic *wa* response includes the following members: "success" indicates if the request was successfully executed, "message" defines a returned textual note (particularly useful if errors occurs), "total" identifies the number of returned elements and "data" lists the elements with their attributes and structure. Note that "total" and "data" are returned only if data are actually requested.
To better clarify how to interact with the *wa* service, in the following paragraphs a series of examples of communications with a service registered at the URL *http://localhost/istsos/wa/* with reference to the data quality element are presented. Each request will report the used HTTP method, the URI for the request, the Request Body and the Response:

Gather information about the quality index with code 200 for the service *demo*:

Method: GET
Address:*http://localhost/istsos/wa/istsos/services/demo/dataqualities/200*
Request Body: -
Response: *{"success": true , "message": "Data qualities of service <demo> successfully retrieved", "total": 1, "data": [{"code": 200, "name": "reasonable", "description": "the value is in a reasonable range"}]}*

This contribution has been peer-reviewed.
doi:10.5194/isprsarchives-XL-5-W3-111-2013

Update the quality index with code 200:

Method: *PUT*
Address:*http://localhost/istsos/wa/istsos/services/demo/dataqualities/200*
Request Body: *{"name": "reasonable", "description": "the value is in a statistical range"}*
Response: *{"message": "Update successful", "success": true}*

Insert a new quality index:

Method: *POST*
Address:*http://localhost/istsos/wa/istsos/services/demo/dataqualities*
Request Body: *{"code": 201, "name": "correct", "description": "the value is correct and manually checked"}*
Response: *{"message": "New element added", "success": true}*

Delete the quality index with code 201:

Method: *DELETE*
Address:*http://localhost/istsos/wa/istsos/services/demo/dataqualities/201*
Request Body: *-*
Response: *{"message": "Element successfully deleted", "success": true}*

In addition to common actions aiming at manipulating elements, the *walibs* offers features to perform general operations like enquiring for service status or testing the database connection.

Figure 2. Elements and relative HTTP method supported in the *walib*

2.3 wainterface

The *wainterface* has been primarily designed to provide a user-friendly environment for setting up and configuring *istsos* services. The interface has a menu allowing to access different sections and a tool bar with section-specific buttons activating different frames.
The currently available sections are: server, services and data editor. When the server section is selected the interface shows a toolbar with buttons for accessing to the following features:

1. about: provides information about the installed version of istSOS, available updates, news and general information about istSOS with relevant links to documentation, mailing list and code repository.
2. status: here the administrator has an overview of instantiated services including the number of registered *featureOfInterest*, *offerings*, *procedures* and *observedProperties*. Moreover, it provides the availability of the services, the corresponding database and the basic accessible requests (*GetCapabilities*, *DescribeSensor*, *GetObservation*, *GetFeatureOfInterest*, *InsertObservation* and *RegisterSensor*).
3. database: form for setting and testing the default PostgreSQL/PostGIS database connection.
4. service provider: form for setting the default service provider information, like contacts and address.
5. Service identification: form for setting the default service information regarding keywords, access fee, and authority.
6. Coordinate systems: form for setting the default coordinate system used for storing geospatial geometries, other supported reference systems and *urn* definition of easting, northing and elevation.
7. GetObservation configuration: form for setting the maximum supported extent in hours in a data request period, quality indexes and values to be used in case of no data in aggregation requests and by default.
8. Proxy configuration: this is the link to the service available on the Web.
9. New service: here the administrator enters in a wizard that guides him to the creation of new services.
10. Delete service: from this frame the administrator can delete an existing service; note that all the related information like procedures *sensorML* and *observations* are permanently cancelled.

The other interface section refers to services, here you can select an existing service and update the information entered during the creation. A first set of features reflects those available for the server section, namely database, service provider, service identification, coordinate system and GetObservation configuration. A second set of buttons are:

1. Offerings: here registered procedures can be associated with offerings according to the administrator preferences. Note that offerings, being a mandatory parameter in getObservation request, can be used to permit and restrict access to observations.
2. Procedures: lists registered procedures and provides links for modifying related metadata (sensorML) or deleting those selected.
3. New procedure: offers a form for registering a new procedure. Using it, *wa* generates the *RegisterSensor* request that is then forwarded to the *istsos* service. Here a check on mandatory parameters is performed.

This contribution has been peer-reviewed.
doi:10.5194/isprsarchives-XL-5-W3-111-2013

4. Observed Properties: allows for registering *observedProperties* definitions to be used in creating new procedures.
5. Unit of measures: permits for registering unit of measure definitions to be used in creating new procedures.
6. Data quality: allows for registering *qualityIndex* definitions to be used in creating new procedures.

The third section of the interface is the data editor (see Figure 3) panel that provides access to observations display and editing. Once the desired time series (service, offering and procedure) are selected, the administrator has to define a time interval and an *observedProperties*: at this point observations are plotted in a chart and displayed in a table. If desired, an editing session can be started. Within the editing session the administrator can select observations, either interactively on the plot or on the table, and adjust values either editing the selected rows or applying formulas using a calculator where values from other series can be used. Modifications are immediately visualized, and committed only once the save button is pressed.

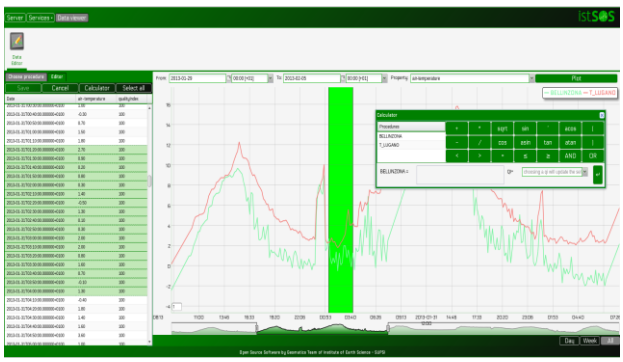

Figure 3. *wainterface*: data editor snapshot

A different application, which is part of the *wainterface* but not integrated in it, is the data viewer. It is a stand-alone Web page that can be used to display observations, according to administration set rules and permissions. Here with respect to the data editor the editing part is missing.

3. REAL CASE APPLICATION

Whenever structural interventions are not feasible, the damage reduction due to flood events passes through a solid early warning system. In fact, timely warning is crucial for efficient emergency response and contingency action planning.
For this reason, as stated in the introduction, a system for Early Warning on the Verbano Lake was set up by the IST between 1999 and 2002 and today is operational with improved capabilities that are mainly due to its renewal in order to fully take advantage of technological advances.
The system is composed by three main blocks: the data acquisition and management, the lake level forecast model and the intervention support system. These blocks are described in the following sections.

3.1 Hydro-meteorological data acquisition and management

In order to provide data for the forecasting model of the Verbano Lake level, the system has to collect information observed in the field (field true) all over its catchment and meteorological forecasts. The catchment in Switzerland is monitored by different agencies for different purposes: MeteoSwiss for meteorological scope, FOEN (Federal Office of ENvironment) for river conservation and TI (Canton Ticino) for land use planning; moreover, being a trans-national basin, the Italian part is monitored by Piemonte regional environmental agency ARPA (Agenzia Regionale per la Protezione dell'Ambiente) and CGL (Consorzio Grandi Laghi). As a result, field observations of hydro-meteorological data have to be collected in near-real time from several agencies and homogenized in order to provide consistent inputs for the forecasting modelling. Moreover, due to international conventions, data should be exchanged between the partners.
In the past, field instruments of TI and FOEN were equipped with GSM modem, so at defined time intervals (1 day or 2 hours in case of meteorological alert), the IST executed batches that called the sensor and downloaded the data. Then, MeteoSwiss, ARPA and CGL provided ASCII files through File Transfer Protocol (FTP) with updated observations. Transmissions occurred at given intervals and each dataset had overlapping periods between consecutives dispatches but with updated observations.
It is clear that the system is redundant, inconsistent and error prone. Redundant because each partner has to manage the data belonging, collected and already managed by others agencies. Inconsistent because late observation collection due to temporary instrument unavailability, or late corrected measures due to instruments deficiencies may lead to not unique datasets. Error prone, because FTP protocol does not provide transmission errors management and thus errors cannot be automatically caught and appropriate patching action cannot be automatically triggered.
For all the above expressed reasons it would be advised that all the partners move towards a service-oriented architecture using a standard protocol. Following this vision, the IST decided to test, validate and promote the adoption of one available standard for serving observation.
Today, using istSOS, the IST offers the TI monitoring network data with the SOS standard. Observations from 136 sensors deployed on the field pushes into the system, thanks to the GPRS, real-time measurements of rainfall, air temperature, humidity, atmospheric pressure and solar radiation, river heights, temperature and discharges, lake level and temperature. More than 30 million of measurements since the 1978 are currently accessible on demand. Moreover, the IST integrated in this system also about 3.5 million measurements a year coming from 111 sensors of partners. Overall, the system registers a monthly data flux of about 5 GB. Requesting 1 day of temperature data at 10 minutes resolution from one station takes approximately 250 milliseconds, while with hourly average aggregation, the same request requires about 450 milliseconds.
In addition, high resolution (about 7x7 Km) meteorological forecasts are received from MeteoSwiss in *grib* format. These are archived in a hybrid system that uses a metadata database together with *geoTIFF* files for data serving and retrieval.

3.2 Hydrological model

The hydrological model is a conceptually based model, where the main parameter (Curve Number) is calibrated at the beginning of each event, while other less important parameters are calibrated on historical events.
The results of the model are the water discharges in the main tributaries of the Verbano Lake and the lake level.
The forecasts have a 3-day horizon and are transmitted through fax, SMS, ftp and e-mail to all the partners.

This contribution has been peer-reviewed.
doi:10.5194/isprsarchives-XL-5-W3-111-2013

The model has been operative for more than ten years and has proven several times his consistency, in particular during the 2000 and 2002 flood events in the Verbano Lake.
In the last 3 years, due to a complete change in the database infrastructure, the model underwent various improvements.
The model input is now coming from a Web Processing Service process that:

- Collects data from the SOS.
- Dynamically computes basins total observed rainfall as a result of data interpolation and areal statistics.
- Collects data from weather forecasts.
- Dynamically computes basins total forecasted rainfall as a result of areal statistics.
- Opportunely formats and generates model inputs files.

The model core remains the same, but the interface improves with a better visualization of graphics.
The model outputs are transmitted in various formats to the partners named above and to the authorities which have the competence of civil protection. Besides, the model output is the crucial input for the Sitgap system described below.

3.3 Sitgap

SITGAP is the Italian acronym for: *Geographical Information System for the Management, Alarm and Planning of Verbano Lake flooding interventions*. The system is basically a Web-mapping interface for accessing to a series of information and tools supporting the local Civil Protection in mitigation and evacuation action planning. The application allows to select four different operational levels. Normal level – no floods are expected – allows to administer and retrieve all the emergency institution information, including contact points. Alarm level – floods are likely expected – permits to query and visualize forecasted water levels and corresponding flooded areas, exposed elements and relative information. Action level – an intervention has been activated – allows for planning and controlling field intervention and dislocated resources. Evacuation level – severe floods in place – allows the registration of evacuated persons with destination and the search by person or address.
The Web portal is currently based on Neapoljs for ESRI ArcIMS and ESRI ArcSDE for Oracle, but a new improved version is under development and will be likely released in the 2013.

4. CONCLUSIONS

The IST-SUPSI has studied the Sensor Observation Service in a real case application; from this experiment the authors have identified a series of requirements not addressed by the standard. Thus they engineered their own software solution that, thanks to specific extending features, fills the identified gaps.
Like other OGC services, SOS is designed to be machine readable. In fact, it is difficult for humans to directly interact with this type of services: think of what WMS would be without visualizers like OpenLayers. Similarly, SOS, in order to be really consumed by scientists and interested users, requires graphical interfaces that simplify the interaction. For this reason, the authors implemented a RESTful service capable of exposing system setting and SOS requests generation. A graphical user interface was then designed to ease the access to these features. As a result, the user is now able to easily manage the system, create new services, register new sensors and insert observations without any XML exposure.
The system in place for the Verbano Lake flooding provided an ideal real case to verify the implemented solution robustness. Thanks to istSOS, the IST's scientists have now access to a centralized point that aggregates information despite the sensor manufacturer, observation type or measured property. Moreover, hydrologists have now access to a more flexible and reliable system that, for example, takes automatically advantage of new available observations for the lake level forecasting or provides alerts whenever inactive or corrupted sensors are identified.
As a final comment, the authors proved that SOS has definitely an enormous potential, but, to be fully operational some gaps have to be filled and interfaces are required.

5. REFERENCES

Bleier, T., Bozic, B., Bumerl-Lexa, R., Da Costa, A., Costes, S., Iosifescu, I., Martin, O., Frysinger, S., Havlik, D., Hilbring, D., Jacques, P., Klopfer, M., Kunz, S., Kutschera, P., Lidstone, M., Middleton, S., Roberts, Z., Sabeur, Z., Schabauer, J., Schlobinski, S., Shu, T., Simonis, I., Stevenot, B., Uslander, T., Watson, K. and Wittamore, K., 2009. *SANY - An Open Service Architecture for Sensor Networks*. SANY Consortium.

Cannata, M., and Antonovic, M., 2010. istSOS: investigation of the sensor observation service. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Como, Italy, Vol. XXXVIII-4/W13.

Dumpleton, G., 2013. Mod_wsgi documentation http://code.google.com/p/modwsgi (24 Jan. 2013).

Obe, R. and Hsu, L., 2011. PostGIS in Action. Manning Publications, Stamford, 520 p.

OGC, 2007. *Open Geospatial Consortium, OpenGIS Sensor Observation Service. OpenGIS® Implementation Standard* http://www. ogcnetwork.net/SOS (7 Feb. 2013).

ORCHESTRA consortium, 2010. Project portal http://www.eu-orchestra.org (7 Feb. 2013).

Phillip, J.E., 2011. Python Web Server Gateway Interface v1.0, http://www.python.org/dev/peps/pep-0333 (24 Jan. 2013).

Pirotti, F., Guarnieri, A., Vettore, A., 2011. Collaborative Web-GIS design: a case study for road risk analysis and monitoring. *Transactions in GIS*, 15(2), pp. 213-226

Richardson, L., and Ruby, S., 2007. *RESTful Web Services*. O'Reilly Media, Sebastopol, 419 p.

Sencha, 2013. Ext JS Sencha Docs http://docs.sencha.com (7 Feb. 2013).

The Apache Software Foundation, 2013. Apache HTTP Server Version 2.2 Documentation http://httpd.apa-che.org/docs/2.2 (24 Jan. 2013).

TRIDEC consortium, 2012. Project portal http://www.tridec-online.eu (7 Feb. 2013).

Van Rossum, G., 2003. The Python Language Reference Manual. Network Theory Ltd, Bristol, 144 p.

This contribution has been peer-reviewed.
doi:10.5194/isprsarchives-XL-5-W3-111-2013

Vanderkam, D., 2008. dygraphs JavaScript Visualization Library http://dygraphs.com (7 Feb. 2013).

Worsley, J.C., and Drake, J.D., 2002. *Practical PostgreSQL: a hardened, robust, open source database*. O'Reilly Media, Sebastopol, 640 p.

6. AKNOWLEDGEMENTS

The authors thank all the partners of the Verbano Lake working group for flood risk reduction, and specially the Canton Ticino, (Ufficio dei corsi d'acqua), for supporting this project. Special thanks to Dr. Fleming and CSIR for contributing to the software development and Francesco Massa for testing and documentation.