

**GVSOS:
A NEW CLIENT FOR OGC® SENSOR
OBSERVATION SERVICE INTERFACE
STANDARD**

Alain Tamayo, Joaquín Huerta, Carlos Granell, Laura Díaz, Ricardo Quirós

Department of Information Systems
Universitat Jaume I, Spain.

alain.tamayo@gmail.com, {huerta, carlos.granell, laura.diaz, quiros}@lsi.uji.es

Keywords: Sensor systems, SOS, Sensor Web Enablement, gvSIG

Abstract

A key problem with sensor networks is achieving interoperability between different networks potentially built using different software and hardware platforms. Services interfaced by the Open Geospatial Consortium (OGC) specifications allow GIS clients to access geospatial data without knowing the details about how this data is gathered or stored. Currently, OGC is working on a set of interoperable interfaces and metadata encodings known as Sensor Web Enablement (SWE) that enables the integration of heterogeneous sensor systems and measurements into geospatial information infrastructures. In this article we present the implementation of gvSOS, a new module for gvSIG to connect to Sensor Observation Services (SOS). The gvSOS client module allows gvSIG users to interact with SOS servers, displaying the information gathered by sensors as a layer composed by features. We present the software engineering development process followed to build the module. For each step of the process we specify the main obstacles found during the development such as, restrictions of the gvSIG architecture, inaccuracies in the OGC specifications, and a set of common problems found in current SOS server implementations available on the Internet. For most of the problems found we propose a solution, or at least we present a path that might lead to it.

1. Introduction

Sensor networks, computer networks that provide access to spatially distributed sensors that monitor physical phenomena like temperature, sound and pressure (Percivall and Reed, 2006)(Chen et al., 2009), are gaining popularity in the development of software applications. Reasons for the expansion of sensor technology are basically twofold. Recent improvements in sensor hardware and communications ease the creation of smaller and inexpensive sensors thanks, for instance, to the use of low-cost processors and integrated radio transmitters (Kanoun and Tränkler, 2004). In addition, sensors are also revolutionising our daily life because they can be used in a wide variety of applications. Examples include from scientific contexts like environmental monitoring (Martinez et al., 2004), habitat monitoring (Mainwaring et al., 2002)(Szewczyk et al., 2004), structural health monitoring (Paek et al.,2005)(Chintalapudi et al., 2006), seismic detection (Werner-Allen et al., 2005) (Werner-Allen et al., 2006) to focused applications like augmented reality (Kealy and Scott-Young, 2006) and serious games (LaViola, 2008) applications.

Apart from dealing with the inherent complexity of individual networks, a current problem already pointed out in the literature (Douglas et al., 2008) (Jabeur et al., 2009) is achieving interoperability between different sensor networks, since accessing observational datasets is often limited by incompatibilities between systems and protocols. In the field of environmental modelling it is common to integrate heterogonous data sources to run and calibrate hydrological models for a given watershed (Díaz et al., 2008). In such application scenarios flood gauges, air pollution monitors, meteorological sensor stations are combined with satellite imagery and other spatial data in order to provide the needed input data for the model. In practice, however, each remote data source may use distinct encodings, formats and even

communication protocols (Chen and Helal, 2008) leading to a lack of standardization and interoperability problems when multiple sensor networks are combined.

Over the last few years, a lot of work has been devoted to standardise the components and interfaces. The mechanisms exposed by Spatial Data Infrastructures (SDI) (Masser, 2005) have been widely adopted to permit interoperable access to geospatial data through service interfaces like Web Map Service Implementation Specification (WMS) (OGC, 2006), Web Coverage Service Interface Implementation Specification (WCS) (OGC, 2006a) and Web Feature Service Implementation Specification (WFS) (OGC, 2005). Using these service specifications, GIS clients can access geospatial data without knowing details about how this data is gathered or stored. In this way, data can be shared and reused, users can choose the best tool for the task, and people with less training can benefit from using geospatial data in more applications (OGC, 2005a).

In the sensor context, standardization also simplifies interoperability issues between components inside specific networks and the interoperability of different clients with different networks. The Sensor Web offers a solution to the problem of interconnecting sensor networks. The Sensor Web (van Zyl et al., 2009) (Liang et al., 2005) (Jabeur et al., 2009) defines an infrastructure and open standards that provide open access to sensors, sensor networks, and their observational datasets. The sensor layer in SDIs is being encompassed by a recent initiative called Sensor Web Enablement (SWE), for “specifying interoperability interfaces and metadata encodings that enable real time integration of heterogeneous sensor webs into the information infrastructure” (OGC, 2008). This framework of open standards for exploiting sensor of all types should allow sensor nodes and their corresponding sensor networks to be monitored and controlled through web interfaces using GIS clients (Percival and Reed, 2006).

The current systems for the Sensor Web described in the Related Work section still suffer some problems that limit Sensor Web and SDI integration:

- Although observational datasets have been identified as key element in spatial data infrastructures, they are still isolated in the sense that they are not usually incorporated in SDI contexts to the same degree as other geospatial data.
- Observational datasets are still disconnected from processing capabilities. This feature is critical, for instance, in various domains concerned with analysis tasks in risk management situations where processing raw sensor data to produce meaningful information is extremely important.

This article deals with the above issues by providing a new module for gvSIG (gvSIG, 2009) to access observational datasets in the realm of SWE standards and SDI ideals. gvSIG is an open source SDI thick-client designed for managing geographic information, providing support for common data formats, including vector and raster spatial data, remote spatial databases, and standard OGC web services. The Sensor Observation Services (SOS) client module allows gvSIG users to connect to and interact with SOS servers offering observations and measurements gathered by sensors. In addition, by providing easy access to sensor data on a client-side processing tool like gvSIG, we are enabling data processing and data fusion capabilities to fully exploit the potential of the Sensor Web, letting users and decision-makers process and use real-time sensor data from heterogeneous distributed sensor systems and networks.

We present the software engineering development process followed to build the module. For each step of the process we specify the main obstacles found during the development, such as restrictions of the gvSIG architecture, inaccuracies in the OGC's specifications, and a set of common problems found in current SOS servers implementations available on the Internet. The rest of the document is structured as follows: Section 2 presents an introduction to OGC

standards related with SWE. Section 3 introduces gvSIG. In Section 4, we present the analysis and design phases of the SOS client. Further details on the implementation of the main use cases are presented in Section 5. In Section 6 we discuss some tests and preliminary results. Section 7 presents related work in the sensor web context. Finally, in Section 8 we present some conclusions and briefly outline future work.

2. Sensor Web Enablement

Despite of the large number of existing sensor networks deployment, most of them remained traditionally close to certain sensor communities offering limited mechanisms for interoperability (OGC, 2008a). The OGC Sensor Web Enablement (SWE) initiative aims to solve this situation by providing a set of interoperable, standardised interfaces, data encodings and metadata to make sensor resources openly accessible and available through Internet (OGC, 2008)(Zyl et al., 2009). In essence, interoperable and scalable service-oriented networks wrapping heterogeneous sensor systems can be implemented based on the models, encodings, and services of the SWE architecture (OGC, 2007). SWE acts like a middleware between the physical sensor networks and the software clients operated by final users. In this section we briefly introduce the specifications related to our implementation.

SWE includes several implementation specifications defining services and encodings.

Implementation specifications for encodings are:

- *Observation & Measurement Schema (O&M)*: It defines standard models and XML Schema for encoding observations and measurements from a sensor (OGC, 2007a).
- *Sensor Model Language (SensorML)*: It defines standard models and XML Schema for describing sensor systems and processes. (OGC, 2007b).

- *Transducer Markup Language (TransducerML or TML)*: It defines the conceptual model and XML Schema for describing transducers and supporting real-time streaming of data to and from sensor systems (OGC, 2007c).

Implementation specifications for services are:

- *Sensor Observations Service (SOS)*: Standard web service interface for requesting, filtering, and retrieving observations and sensor system information. (OGC, 2007d).
- *Sensor Planning Service (SPS)*: Standard web service interface for requesting user-driven acquisitions and observations. (OGC, 2007e).

2.1. Encodings

As mentioned before, O&M defines standard models and XML Schema for encoding observations and measurements from sensors. Observations are related with (OGC, 2007a):

- a *feature of interest*, which is a feature representing the real world object which is the observation target;
- an *observed property* identifying or describing the phenomenon for which a value is measured or estimated;
- a *procedure* describing the process used to produce the result; and a *result* containing the value produced by the procedure.

On the other hand, SensorML defines standard models and XML Schema for describing sensor systems and processes. One of the main concepts in SensorML is *process*, which defines inputs, outputs, parameters, and a method for that process, as well as a collection of metadata useful for discovery and user assistance. Sensor systems are modelled as a collection of physical and non-physical processes. The former is concerned with detectors, actuators and sensor

systems, all of which have some relationship with space and time. The latter includes processes that can be modelled by mathematical operations (OGC, 2007b).

2.2 Services

The SOS specification provide access to observations from sensors and sensor systems in a standard way that is consistent for all sensor systems including remote, in-situ, fixed and mobile sensors (OGC, 2007d). The information exchanged between SOS clients and servers follows the O&M specification for observations and the SensorML specification for sensors or system of sensors descriptions. Observations are grouped into *observation offerings*. An observation offering is a set of related observations that follow some criteria. Unfortunately, the specification does not provide a more precise definition, or any clear guidance on doing this grouping.

The SOS implementation specification defines three operation profiles: core profile (mandatory), transactional profile (optional) and enhanced profile (optional). The core profile contains three operations: *GetCapabilities*, *DescribeSensor*, and *GetObservation*. These are basic operations needed for any data consumer to access sensor observations stored in a SOS server. The *GetCapabilities* operation common for all OGC services allows clients to retrieve service metadata from the server. The *DescribeSensor* operation allows SOS clients to retrieve SensorML or TML descriptions of a given sensor. The *GetObservation* operation is used to retrieve sensor observations from the server. The other two profiles offer operations to support data producers (*RegisterSensor*, *InsertObservation*) and operations to provide clients with a richer interface for interacting with the server (*GetResult*, *GetFeatureOfInterest*, *GetFeatureOfInterestTime*, *DescribeFeatureOfInterest*, *DescribeObservationType*, and *DescribeResultModel*).

3. gvSIG

gvSIG¹ is a full-fledged GIS open source tool that provides support for common data formats, including vector and raster spatial data, remote spatial databases, and standard OGC web services. It provides the most common GIS tools such as map navigation, query map information, distance measurement, thematic cartography, legend edition, labelling, feature selection, data tables with statistics, geoprocessing tools, CAD and raster processing (Anguix and Díaz, 2008). gvSIG allows for the combination in a single view of geospatial data with different formats retrieved from different sources such as local files or an SDI. This software is developed by the Regional Council for Infrastructures and Transportation of Valencia, Spain, using Java under the GNU General Public License (GPL). Its name is an abbreviation of "*Generalitat Valenciana, Sistema d'Informació Geogràfica*" (gvSIG, 2009)

gvSIG is built using a plug-in model where functionality can be added to a generic framework called *Andami* that permits building multiple documents interface applications. Andami is an extensible framework and can be customized using plug-ins to build different kinds of applications. The GIS-specific behaviour is added to the application by its three main subsystems (gvSIG, 2009a):

- *gvSIG*: The gvSIG subsystem is the equivalent of the Presentation layer in an enterprise layered architecture. It handles the interaction between the system and the user. It provides the user with the graphic representation of geographical data and the tools to interact with this data.

¹ All the information presented here corresponds to gvSIG 1.1.2

- *FMAP*: This library includes all that is needed to handle GIS objects. It includes classes that can draw layers, assign legends, execute queries, make spatial analysis, etc.
- *SubDriver*: Contains classes to access the different data formats. It handles the communication with real data sources isolating other layers from the specific details of the interaction

A gvSIG plug-in is a module containing a set of extensions to add new functionality to the system. An extension is a Java class implementing the *IExtension* interface that bridges new functionality with the existing one (gvSIG, 2009a). Every installed plug-in is located in a single directory called gvSIG\extensions and loaded by Andami at start-up.

gvSIG allows users to create projects containing views, tables and layouts. Views contain the graphical representation of geospatial information and its corresponding legend. Tables contain the representation of alphanumeric data related with the geospatial information contained in the views. Layouts are representations of views in a printer friendly format.

4. Analysis and Design

In this section we present the analysis and design phases of the SOS client plug-in gvSOS. We expose the plug-in's use case model and high-level architecture followed by a more detailed view of the main gvSOS layers. gvSOS implements the Core profile of the SOS 1.0.0 specification and other supporting specifications such as SensorML 1.0.1 and O&M 1.0.0.

4.1 Use case model

The use case model for gvSOS is very simple, containing only four use cases and two actors.

The use cases are:

- *Initialize Extension*: This includes the initialization of the extension carried out at application start-up, when user interface elements related with the extension are registered (toolbar buttons, menu items, connection wizards, etc.)
- *Add SOS Layer*: In this use case the user adds a new SOS layer to a gvSIG view. This process includes the connection to the SOS server, and the selection of the information will be displayed in the layer.
- *Display Sensors Information*: Once displayed in a view, sensors can be selected by the user to ask for further details about them. This information will be displayed in a table in a different window.
- *Display Sensors Observations*: The sensors can also be selected to request their observations. This information can be filtered using temporal or spatial filters, depending on the server's capabilities.

The actors involved are *End Users* that fires use cases related with retrieving and displaying information about sensors and their observations; and *gvSIG*, that triggers the plug-in initialization at system start-up.

4.2 High Level Architecture.

The high-level architecture is similar to the gvSIG architecture. It is arranged as a variation of the layer pattern (Buschman et al., 1996), containing the same main gvSIG layers as can be seen in the package diagram in Figure 1:

- *gvSIG-SOS*: It contains the plug-in user interface, providing the wizard used by the user to connect to the SOS server and dialog boxes to request information about sensors and observations. This package extends the gvSIG subsystem.
- *FMap-SOS*: This package contains the code for integrating the information received from the server with the rest of the information in the system. It contains the code to create and draw layers, drivers for interacting with the remote clients, etc. It is an extension of the FMap subsystem.
- *RemoteServices-SOS*: It implements the low-level communication with the remote SOS server, isolating the rest of the system from the details of this interaction. This package is an extension of the SubDriver subsystem.

4.3 Geospatial Data Handling: Fmap layer

The internal functioning of the *Fmap* package is a rather complex one. Simplifying things we can say that *SOSFeature* are the features that are represented in a layer of type *FLyrSOS*, in this case they represent individual sensors or sensor systems. *FlyrSOS* uses an instance of *FMapSOSDriver* to recover all the data from the server. Once the data is retrieved, *SOSAdapter* processes this information. *FMapSOSDriver* inherits from *ConcreteMemoryDriver*, which allows data to be kept in memory for caching purposes. *FMapSOSDriver* recovers data from the SOS server using functionality provided by lower layers. A class diagram showing these relationships is presented in Figure 2.

At this point a major design challenge was found: the gvSIG architecture provides support for adding and caching features in *ConcreteMemoryDriver*, but only information about geometries and simple attributes values can be included in such features. There is no support for complex data structures like sensors containing information about observations, with the

added complexity that observations hold intrinsically a temporal component. To solve this problem the sensors descriptions are kept as features following the gvSIG model, but observations are stored in two nested hash tables. The first table uses the sensor identifier as key to link the sensor with an internal table containing the observations data. This data uses the observed property as key to access pair values including the time instant and the measured value for an observation.

4.4 Geospatial Data Access: SubDriver layer

In this layer, the analysis package RemoteClient-SOS is implemented. The RemoteServices-SOS package implements the communication with the SOS Server and it resembles most of the structure of the rest of the OWS extensions already included in gvSIG. The main classes in this package are *SOSClient* and *SOSProtocolHandler* (Figure 3). *SOSClient* represents the SOS client end-point, encapsulating the logic for connecting to and requesting operations from the SOS server. *SOSProtocolHandler* is in charge of sending all the requests from a SOS client to the proper server. Only the SOS specification *core profile* is supported so far. *SOSProtocolHandler* is a subclass of the abstract class *org.gvsig.remoteClient.OGCProtocolHandler*, which implements the common behaviour for all OGC services. *SOSProtocolHandler* is also abstract and it must be specialized for every version of the SOS specification.

5. Implementation

In this section we explain in further detail the main use cases. We show how end users can interact with gvSOS to display sensors and information gathered by them.

5.1 “Add SOS Layer” Use Case

We start with the “Add SOS Layer” that is started when, after creating a view, the user wants to add a new layer. By selecting the option *Layer\Add Layer* from the menu or pressing the “Add Layer” toolbar button a dialog box is shown. Within this dialog box we select the “SOS” tab to specify the parameters to connect to the SOS server and retrieve the data describing a given observation offering. The *SOSWizard* class handles the interaction with the user to establish the connection with the server and to select the observation offering to be displayed. After establishing the connection and setting the required parameters, *SOSWizard* must return to the gvSIG subsystem the layer to be displayed. This layer contains all of the sensors within the offering represented as points. The gvSIG subsystem uses this layer to create the view that shows the information to the user. *SOSWizard* is composed of several panels and uses some helper classes to fill them with the necessary data. The most important helper class is *FMapSOSDriver*, already mentioned in previous sections.

An example of the final result of these operations is shown in Figure 4. In the view four layers are included; the first one starting from the bottom contains an ECW image of the Iberian Peninsula. The second and third layers contain boundaries from Spain and Spanish provinces respectively. The layer at the top contains sensor systems located in harbours of four Spanish cities retrieved from a local SOS server used for testing purposes. Sensors systems are represented as yellow points.

5.2 Show sensors and observations information use cases

Once the layer is displayed the last two use cases can be started by using the mouse to select the target sensors and then clicking the “See table attributes” button, to get the sensors information,

or the “Get Observations” button to get the observations. Information about sensors is displayed in a table showing the main sensor attributes such as location, input and outputs, etc. Interaction with the remote server is not necessary to implement this behaviour because metadata about sensors was already read when the layer was loaded. The *Fmap* subsystem implements this behaviour in class *FlyrVect*, the direct ancestor of *FlyrSOS*. *FlyrVect* keeps an internal table model which is updated every time the layer content changes. This table model is used by gvSIG to show a table with the stored data at user request.

To display information about observations users must select the sensors first and then press the *GetObservation* toolbar button. This action displays a dialog box where the observed properties to be shown and a set of filters can be specified. The filters included in this version are only temporal filters, which allow the specification of time intervals or time instants to constrain the observations. Using the data entered in this dialog box, *GetObservation* requests for each sensor are generated and sent to the server. After the observations are read from the server a dialog box showing them in a table is displayed to the users (Figure 5). In the left side of the dialog box users have a list with the requested sensors. Depending on the selected sensor, a table containing observations values is displayed on the right side. This table includes the time in which every observation was made and the value of the observation. The example in Figure 5 shows the value of the *WaterLevel* property in a four days period, sampling the property every twelve hours.

The user can also display this information as a graph, being this in most of the cases a more convenient solution than just looking into the bare numbers (Figure 6).

6. Tests and Preliminary Result

Although the module implementation is not complete yet, tests have been executed to validate the functioning of some of its components. These tests have revealed some details in the SOS specification that complicates the implementation of clients and servers:

- In the server capabilities document there is no information about how procedures and observed properties of an offering are related. The only way to know which observed property is related with a given sensor is by executing a *DescribeSensor* operation. This problem make a user interface where users can select the information to be shown whether by observed properties or by sensors harder to implement and less responsive. Considering that users do not have information *a-priori* about how sensors and observed properties relate, they must wait for several requests to be issued to and answered by the server before they could select a given observed property measured by a given sensor.
- The time attribute for offerings permits specifying a temporal reference system through the frame attribute. Default ISO 8601 and the Gregorian calendar with UTC are used. Having only this information does not seem to be enough. ISO 8601 allows several variations in the dates and times representation with basic and extended formats. After running some tests with SOS servers available on the Internet, we have observed that they do not support all possible variations in the date and time formats. Usually only one or a few variations are supported. This situation can provoke that a client sending valid ISO 8601 time values to the server may receive “Invalid Time Format” responses. To solve this problem gvSOS remembers how servers represent time values and convert any time value sent to it to this format. This problem could be avoided if the capabilities

document included more detailed information about the time format supported by the server.

- Some examined server implementations do not use the parameter name *sensorId* to identify the sensor or sensor systems to be described using the *DescribeSensor* operation. Instead the parameter name *procedure* is used. This implies that client issuing a correct request to a server may receive an error message. This name difference seems to be motivated by the fact that the name *sensorId* does not necessarily refers to a sensor in this case. It refers to some of the procedures included in the server's capabilities document, which according to the SOS specification can be "...sensor systems, instruments, simulators, etc.".
- Practically no information about temporal filters is included in any of the OGC specifications. Maybe this is the reason why usually service implementations omit the *FilterCapabilities* section from the capabilities document. To complicate matters, filters specified in a *GetObservation* request are scattered all around since temporal filters are included in the *sos:eventTime* tag, Id and spatial filters are included in the *sos:featureOfInterest* tag, and scalar filters are included in the *sos:result* tag. A solution to this problem could be to include all of the filters used in a *GetObservation* request in a single section, much in the same way it is done in the capabilities document.
- The SOS specification states that zero or many temporal filters (*eventTime* tags) may be specified in a *GetObservation* request. There is no mechanism for a SOS client to know the number of filters supported for a specific server. As a result a client might send a correct request containing more filters than supported by the server, without any warranties about what the server will response. A first attempt to solve this issue could be to include an attribute to every filter category specifying how many filter are

supported for the category. This solution could be extended specifying this value for every individual filter type.

7. Related Work

The fact that SWE specifications are so recent is reflected in the relatively few known implementations of SWE-based services. OGC keeps a list of products compliant with its specifications or implementing them. In such list, only a handful of products are listed that implement the SWE specifications, in either the client or the server versions. On the server side, SOS implementations are now part of widely known GIS products such as MapServer (MapServer, 2009), Degree (Degree, 2009), or part of others product suites such as those produced by 52°North Initiative or the VisAnalysis Systems Technologies (VAST) team, from the University of Alabama in Huntsville (UAH). A comparison between some SOS servers can be found in (Chen et al., 2009).

On the client side, the trend so far has been the implementation of custom-made clients to connect to specific servers. This is due to the fact that building a client using web technologies such as Perl or PHP, combined with mapping services such as Google Maps is relatively easy. Another reason is that building a “general-purpose” client, able to interoperate with several server products, is very hard even if both follow the same specification (see Section 6). Even though, there are two products including SOS clients that in our opinion must be mentioned: the 52°North’s OX-Framework and the VAST’s Space Time Toolkit (STT). Both products allow the connection to several OGC Services, SOS included.

The OX-Framework provides developers with a client architecture where the information of several kinds of OGC Services can be accessed, visualized and integrated (52North, 2009). The framework can be used to construct different types of applications such as thick or mobile

clients. It can also be extended to support other OGC services by adding new “connectors”. Currently, different versions of the WMS, WCS, SOS and SAS specifications are supported.

The VAST team has developed multiple tools related with SWE specifications, most of them specialized in working with SensorML. The most relevant for this discussion is the Space Time Toolkit (STT), which provides capabilities for integrating spatially and temporally-disparate data in a highly interactive 3D display environment. Data retrieved from different OGC services, including SOS, can be handled and visualized using this tool (UAH VAST, 2009).

The combination of gvSOS and gvSIG produces a similar product, but with larger capabilities. gvSIG does not only provides access to OGC Services, it also provides access to an enormous number of vector and spatial formats such as: SHP, DGN, DXF, JPEG2000, DWG, etc.; and spatial databases such as PostGIS, Oracle and MySQL. Sensor data can be integrated, processed using different functions, and exported to different formats.

8. Conclusions

We presented details of the development process for a new module for gvSIG to connect to Sensor Observation Services. The SOS client module allows gvSIG users to interact with SOS servers, displaying information about sensors and observations in a set of layers composed by features. Sensors are shown as features in the map and their information can be inspected also in tabular form. Observations can be inspected through tables or graphs, being the latter a more convenient choice for users. We also mentioned some details found in the SOS specification making difficult the implementation of servers and clients. However, the OGC SWE specifications are a first attempt to make sensor resources broadly available to isolated communities to foster the integration of sensor data in Spatial Data Infrastructures.

Finally, future research should try to align with ongoing sensor experiments as in the context of GEOSS and the interoperability experiments within the OWS phase-6, which pursue among other open issues to create geospatial processing workflows combining sensor data with mainstream GI data. Currently, gvSOS is being extended with the Enhanced Profile of the SOS 1.0.0 specification.

Acknowledgements

This work has been partially supported by the European Commission, Erasmus Mundus Programme, M.Sc. in Geospatial Technologies, project no. 2007-0064 and by the CENIT *España Virtual* project through the *Instituto Geográfico Nacional* (IGN). We would also like to thank to the Prodevelop staff and to Fran Peñarrubia for their invaluable collaboration.

References

- 52° North 2009 Homepage. WWW document <http://52north.org/> (accessed Feb 14, 2009)
- Anguix A, Díaz L 2008 gvSIG: A GIS desktop solution for an open SDI. *Journal of Geography and Regional Planning* Vol. 1 (3), 41–48
- Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M 1996 *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*. Wiley
- Chen C and Helal S 2008 Sifting Through the Jungle of Sensor Standards. *Pervasive Computing*, vol 7, no 4: 84-88
- Chen N, Di L, Yu G, Min M 2009 A flexible geospatial sensor observation service for diverse sensor data based on Web service. *ISPRS Journal of Photogrammetry and Remote Sensing* 64 , 234-242.
- Chintalapudi K, Paek J, Gnawali O, Fu T S, Dantu K, Caffrey J, Govindan R, Johnson E, Masri S 2006 Structural damage detection and localization using NETSHM. In *Proceedings of the 5th international conference on Information processing in sensor Networks (IPSN) Tennessee, USA*
- Degree 2009 Homepage. WWW document <http://www.deegree.org/> (accessed Apr 14, 2009)
- Díaz L, Granell C and Gould M 2008 Case Study: Geospatial Processing Services for Web-based Hydrological Applications. *Geospatial Services and Applications for the Internet*. Springer Science+Business Media: New York, pp 31-47
- Douglas J, Usländer T, Schimak G, Esteban J F and Denzer R 2008 An Open Distributed Architecture for Sensor Networks for Risk Management. *Sensors*, vol. 8, no 3: 1755-1773.
- gvSIG 2009 Homepage. WWW document <http://www.gvsig.gva.es>. (accessed Feb 14, 2009)
- gvSIG, 2009a Guía de referencia para gvSIG 1.1. WWW document <http://www.gvsig.org/web/docdev/reference>. (accessed Feb 14, 2009)
- Jabeur N, McCarthy J D, Xing X and Graniero P A 2009 A knowledge-oriented meta-framework for integrating sensor network infrastructures. *Computers & Geosciences*, vol 35, no 4: 809-819

- Kanoun O and Tränkler H-R 2004 Sensor Technology Advances and Future Trends. *IEEE Transactions on Instrumentations and Measurements*, vol 53, no 6: 1497-1501
- Kealy A and Scott-Young S 2006 A Technology Fusion Approach for Augmented Reality Applications. *Transactions in GIS*, vol 10, no 2: 279-300.
- LaViola J J 2008. Bringing VR and Spatial 3D Interaction to the Masses through Video Games. *IEEE Computer Graphics and Applications*, vol 28, no 5: 10-15
- Liang S H L, Croitoru A and Vincent Tao C 2005 A distributed geospatial infrastructure for Sensor Web. *Computers & Geosciences*, vol 31, no 2: 221-231
- Mainwaring A, Culler D, Polastre J, Szewczyk R, Anderson J 2002 Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. Georgia, USA
- Mapserver 2009 Homepage. WWW document <http://mapserver.org/> (accessed Apr 14, 2009)
- Martinez K, Hart J K, Ong R 2004 Environmental sensor networks. *IEEE computer* Vol. 37, Issue 8, 50- 56
- Masser I 2005 *GIS Worlds: Creating Spatial Data Infrastructures*. ESRI Press: Redlands
- OGC 2005 *OpenGIS® Web Feature Service Implementation Specification*. Version 1.1.0. OGC Document Number 04-094
- OGC, 2005a "The Importance of Going Open". OGC Whitepaper, 2005
- OGC 2006 *OpenGIS® Web Map Server Implementation Specification*. Version. 1.3.0. OGC Document Number 06-042
- OGC 2006a *OpenGIS® Web Coverage Service (WCS) Implementation Specification*. Version 1.1.0. OGC Document Number 06-083r8
- OGC 2007 *OGC® Sensor Web Enablement: Overview And High Level Architecture*. OGC Whitepaper
- OGC 2007a *Observations and Measurements – Part 1 - Observation schema*. Version 1.0.0. OGC Document Number 07-022r1
- OGC 2007b *OpenGIS® Sensor Model Language (SensorML) Implementation Specification*. Version 1.0.0. OGC Document Number 07-000

OGC 2007c *OpenGIS® Transducer Markup Language (TML) Implementation Specification*. Version 1.0.0. OGC Document Number 06-010r6

OGC 2007d *Sensor Observation Service*. 1.0.0. OGC Document Number 06-009r6

OGC 2007e *OpenGIS® Sensor Planning Service Implementation Specification*. Version 1.0.0. OGC Document Number 07-014r3

OGC 2008 *Sensor Web Enablement WG*, WWW document <http://www.opengeospatial.org/projects/groups/sensorweb> (accessed Feb 14, 2009)

OGC 2008a *OGC® Sensor Web Enablement Architecture*. Version 0.4.0. OGC Document Number 06-021r4

Paek J, Chintalapudi K, Govindan R, Caffrey J, Masri S 2005 A wireless sensor network for structural health monitoring: performance and experience. In *Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors*, Sydney, Australia

Percival G and Reed C 2006 OGC sensor web enablement standards. *Sensors and Transducers Journal*, vol 71, no 9: 698-706

Szewczyk R, Mainwaring A, Polastre J, Anderson J, Culler D 2004 An analysis of a large scale habitat monitoring application. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems*, Maryland, USA

UAH VAST 2009 *Homepage*. WWW Document http://vast.uah.edu/index.php?option=com_content&view=frontpage&Itemid=1 (accessed Feb 14, 2009)

Werner-Allen G, Johnson J, Ruiz M, Lees J and Welsh M 2005 Monitoring volcanic eruptions with a wireless sensor network. In *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN'05)*, Istanbul, Turkey

Werner-Allen G, Lorincz K, Ruiz M, Marcillo O, Johnson J, Lees J and Welsh M 2006 Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, Vol. 10, No. 2, 18-25.

Zyl, T L van, Simonis, I and McFerren, G 2009 The Sensor Web: systems of sensor systems. *International Journal of Digital Earth*, vol 2, no 1, 16 - 30

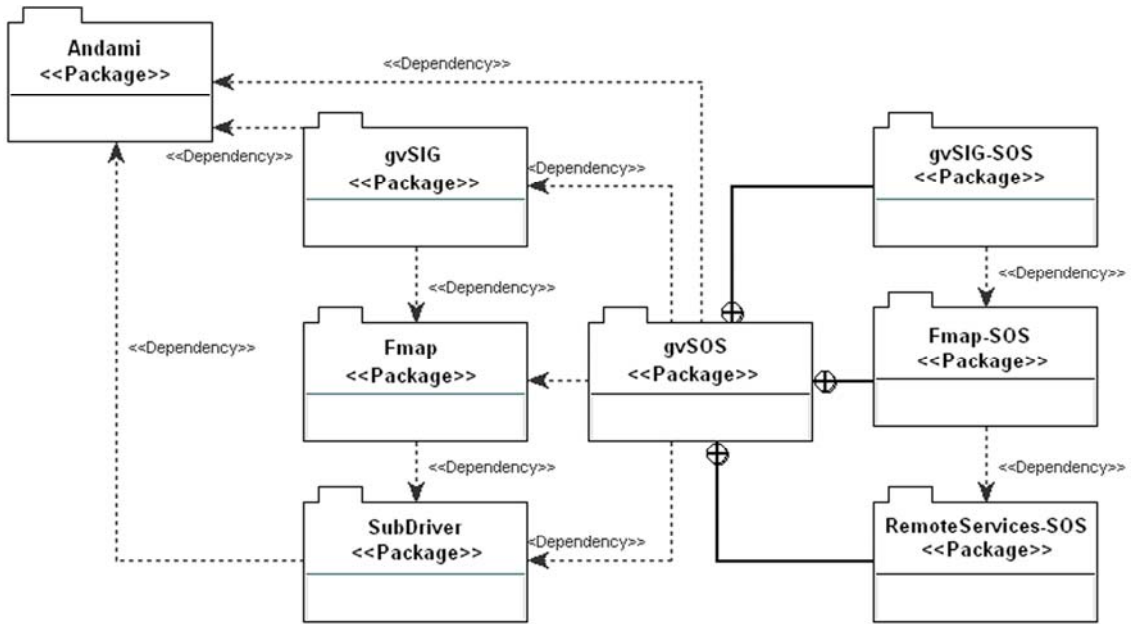


Figure 1. gvSOS architecture.

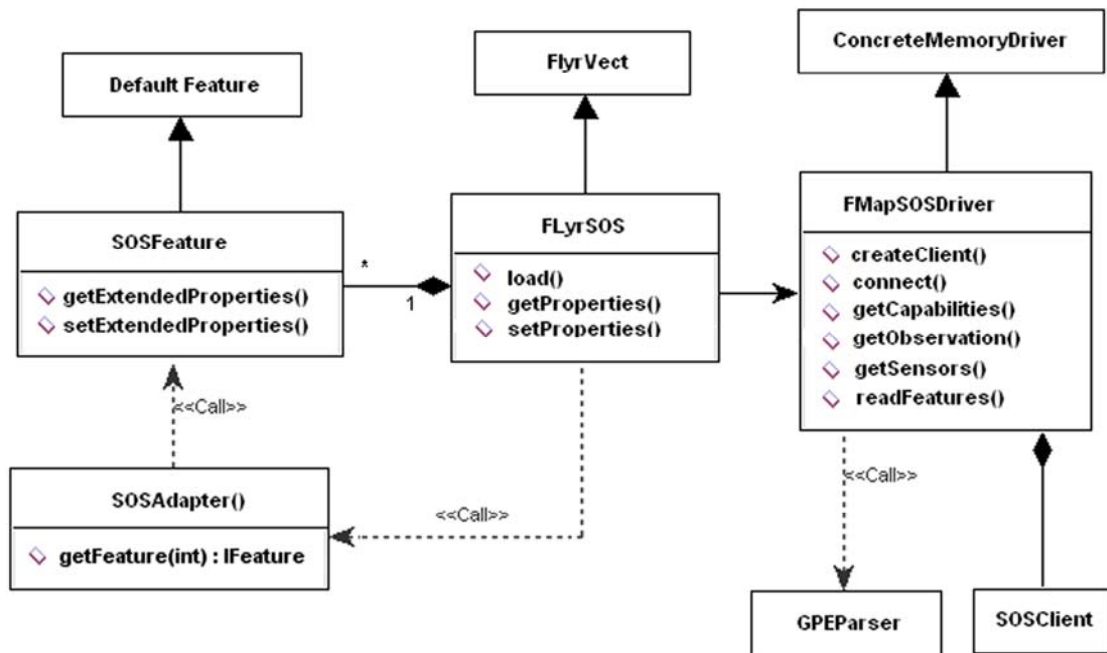


Figure 2: Fmap-SOS class diagram.

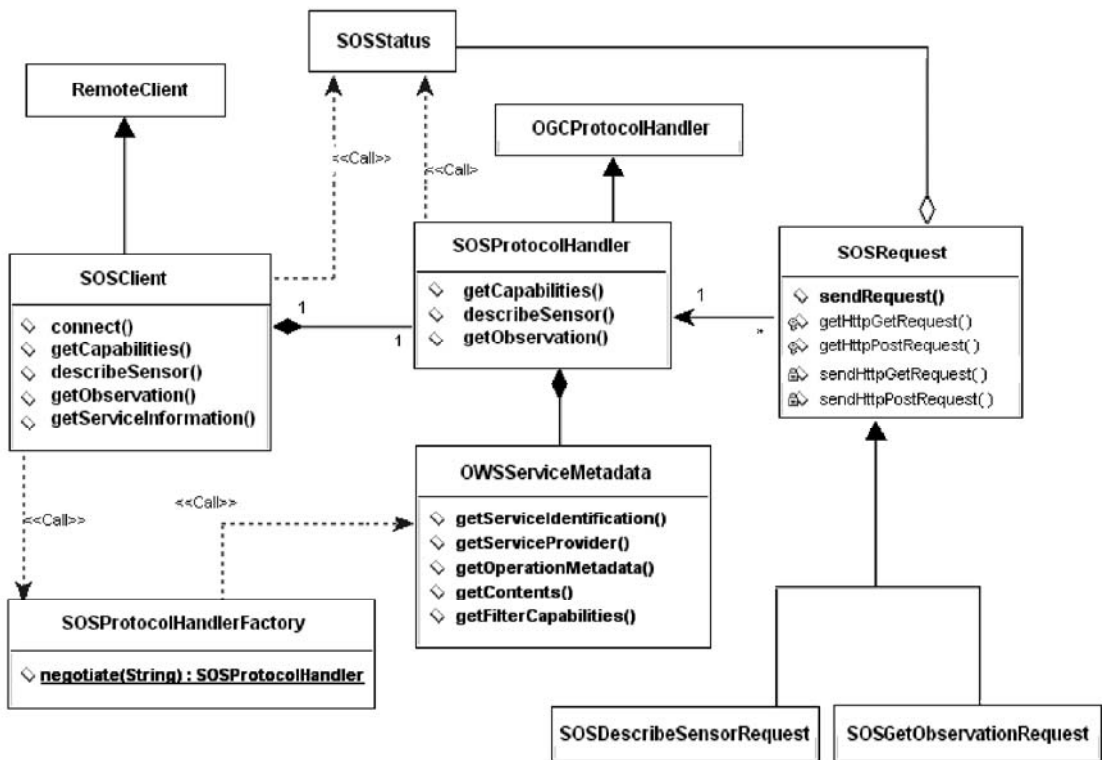
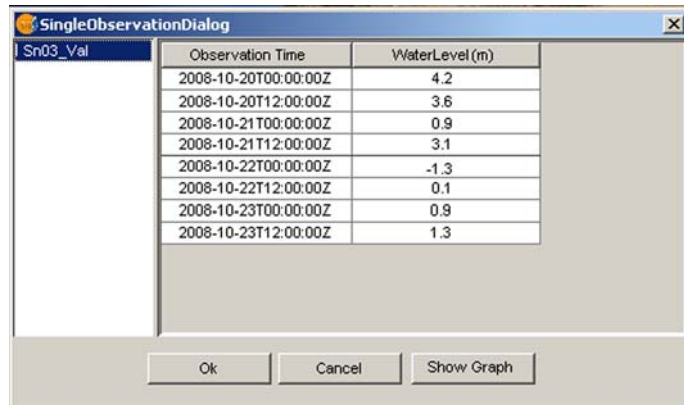


Figure 3: RemoteServices-SOS class diagram.



Figure 4: Sensor systems located in four Mediterranean harbours.



The image shows a software dialog box titled "SingleObservationDialog". It contains a table with two columns: "Observation Time" and "WaterLevel(m)". The table lists eight observations with their corresponding times and water levels. At the bottom of the dialog, there are three buttons: "Ok", "Cancel", and "Show Graph".

Observation Time	WaterLevel(m)
2008-10-20T00:00:00Z	4.2
2008-10-20T12:00:00Z	3.6
2008-10-21T00:00:00Z	0.9
2008-10-21T12:00:00Z	3.1
2008-10-22T00:00:00Z	-1.3
2008-10-22T12:00:00Z	0.1
2008-10-23T00:00:00Z	0.9
2008-10-23T12:00:00Z	1.3

Figure 5: Table showing observation times and values.

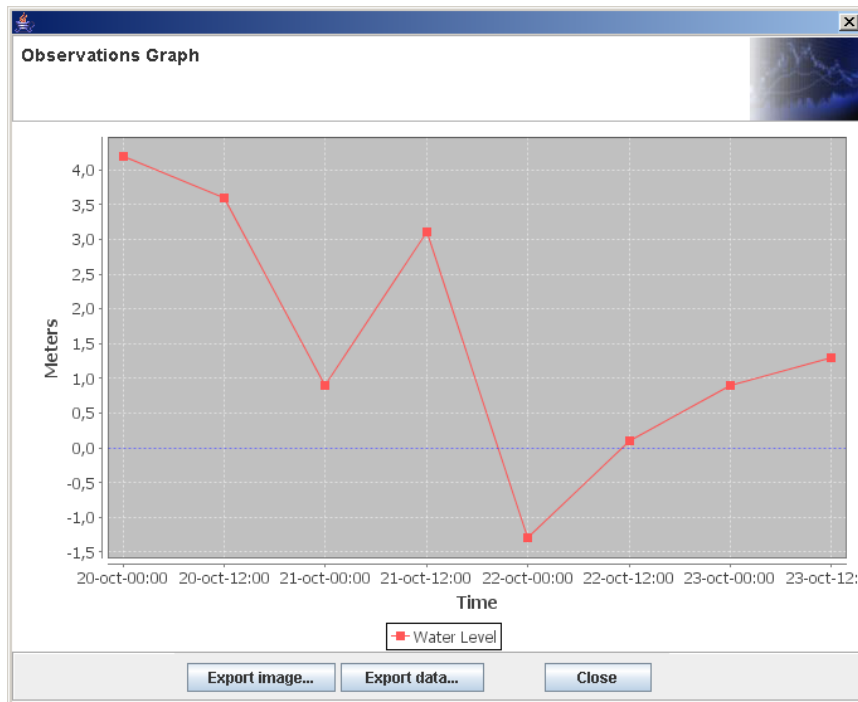


Figure 6. Displaying observations on a graph.

List of Figures

Figure 1. gvSOS architecture.

Figure 2. Fmap-SOS class diagram.

Figure 3. RemoteServices-SOS class diagram.

Figure 4. Sensor systems located in four Mediterranean harbours.

Figure 5. Table showing observation times and values.

Figure 6. Displaying observations on a graph.