Masters Program in Geospatial Technologies

CREATING A GWT WEB APPLICATION FOR THE SOS STANDARD ENHANCED PROFILE

Mauricia Benedito Bordonau

Dissertation submitted in partial fulfilment of the requirements
for the Degree of *Master of Science in Geospatial Technologies*

UNIVERSIDADE NOVA ISEGI
UNIVERSITAT JAUME·I
WESTFÄLISCHE WILHELMS-UNIVERSITÄT MÜNSTER

CREATING A GWT WEB APPLICATION FOR THE SOS STANDARD ENHANCED PROFILE

Thesis supervised by:
PhD Oscar Belmonte
PhD Joaquin Huerta
PhD Miguel Neto

March 2011

ACKNOWLEDGEMENTS

First of all I would like to specially thank Professor Joaquin Huerta for his support from the beginning to the end of this master. Thanks also to my supervisor Professor Oscar Belmonte and co-supervisor Professor Miguel Neto for accepting this responsibility. I also would like to express my gratitude to the researchers Laura Diaz, Jose Gil and Alain Tamayo for their help in all phases of this project. Finally thanks to all my friends of this program for making this experience so grateful.

ABSTRACT

Sensor Networks have become very popular in recent years. With the proliferation of the use of these networks for very different purposes, it has appeared also the necessity of developing one standard to unify all these types of networks and make them interoperable. This standard is Sensor Observation Service (SOS) and was developed by the Open Geospatial Consortium (OGC) in 2007 as part of the Sensor Web Enablement (SWE) activities.

The access to the information provided by sensor networks can be done using PC, laptops on mobile devices such as mobile phones, and that's why in this project, it has been developed a *thin client* in Google Web Toolkit (GWT), which follows the SOS standard to access to all information contained in the server without overloading the device.

KEYWORDS

Sensor Observation Service

Standard Protocol

Google Web Toolkit

Enhanced Profile Operations

Open Geospatial Consortium

Sensor Web Enablement

ACRONYMS

API – Application Program Interface

AJAX – Asynchronous JavaScript And XML

GIS – Geographic Information Science

GWT – Google Web Toolkit

HTML – Hyper Text Markup Language

HTTP – Hyper Text Transfer Protocol

IDE – Integrated Development Environment

INIT – Institut Universitari de Noves Tecnologies de la Imatge

KML – Keyhole Markup Language

OGC – Open Geospatial Consortium

OWS – OGC Web Services

RPC – Remote Procedure Call

SAS – Sensor Alert Service

SDI – Spatial Data Infrastructure

SII – Spatial Information Infrastructure

SPS – Sensor Planning Service

SWE – Sensor Web Enablement

UJI – Universitat Jaume I

URL – Uniform Resource Locator

XML – eXtensible Markup Language

INDEX

INDEX OF FIGURES

CHAPTER 1
INTRODUCTION

In the last decade private and public organizations have done a big effort in order to create Spatial Data Infrastructures (SDI) to manage and monitor relevant data related with different phenomena that has associated geographic information. With this purpose, diverse implementations of sensor networks have emerged.

Sensors have the ability to measure the data and send it to a server. In a sensor network, different sensors are sending its observations to a central sever where they are stored. These sensor networks can be deployed for working over distinct objectives; thereby, nowadays exist sensor networks for managing fields as diverse as environmental monitoring, health care, emergency management, traffic control or military actions. Originally, the communication between sensors and servers was different in each network, that is, each network had the ability to operate using its own protocol. That lack of unification made very difficult the access to stored data from other external services.

In that context arises the idea of creating a communication standard: The Open Geospatial Consortium (OGC) in 2007 defined the first version of Sensor Observation Service (SOS) standard as one of the family of standards and specifications that make up the OGC Sensor Web Enablement (SWE) activities (OGC, 2007). This standard is focused in sensor network management providing them a simple way to control the data and making all systems interoperable. But in all literature consulted about SOS, there is a strong emphasis on how to model phenomena and how to save, into a database, the data entered by the sensors, in other words, the most important part is focused in how to enrich the server part. Nevertheless, it is also very important to have a good platform for distributing the data collected allowing the user to deal with the data in a simple and intuitive way.

In real situations, when asked the server for any data, it is returned an XML file document to the user. XML files are text documents that represent the information i a very structured and reusable way. These files are composed of tags that mark well-defined parts, and these parts can be composed in turn of more parts. For users who

already know the XML format and also for users who are not familiarized with it, in many situations could be interesting to show the information contained in these documents in a more friendly way, for example: using maps, text boxes or graphics. Thus, it is necessary to process the XML document, corresponding to the server response, to automatically extract the useful information to use it in the user applications.

Some extensions, like *gvSOS* presented by (Tamayo, 2009) for gvSIG, or the one for arcGIS: *arcGIS SOS extension* developed by 52North (52 North, 2010), automate the parsing process of the server responses making very simple and comfortable the process of attaching data coming from remote servers to the desktop GIS application.

In this context it becomes essential the implementation of a thin-client able to connect with a server, meeting the defined requirements for the SOS standard providing it the ability to graphically display the georreferenced data stored in the server.

In the document (OGC, 2007), the OGC defined a SOS standard as a set of operations divided into 3 profiles depending on the purpose of each one. Then, there are three profiles: Core, Transactional and Enhanced with 3, 2 and 7 operations respectively which are going to be described in Section 2.2 of this document.

The goal of this project is the implementation of a thin-client able to communicate with SOS service providers and capable to perform the operations defined in the standard. The operations programmed in this thesis are the corresponding ones to the enhanced profile.

During the implementation of this thin-client there has been used a combination of technologies mixing both, new code and third party applications. For the thin-client implementation it has been decided to use Google Web Toolkit (GWT) because it provides a simple way of programming and deploying new applications due to its integration with Eclipse. GWT is an open source and free framework for building and optimizing complex web applications. In a normal approach of programming an Ajax application, the programmer must have some knowledge about HTML, XML and JavaScript.

Using GWT it is possible to design an AJAX application only using just the Java programming language. Another important feature of GWT is that it is compatible with most web browsers and platforms, so any program or application can be executed without major problems.

This new client will make available to all people, the capacity of working with a sensor network. The intention in this thesis has been programming an intuitive application, easy to use, following the SOS standard which can query any SOS server and retrieve the results to show them on a map. This make easier the interpretation of the stored data in any sensor network and it helps the user in the analysis processes over any phenomena stored there.
This thesis is structured as follows:

First of all a state of the art is presented, in this section there are described the main concepts which constitute the basis for this project. Also, there is an overview of the actual framework for sensor networks and SWE applications presenting some real implementations in diverse areas that are already deployed. Next section describes the technologies used; there are presented the thin-client model and the GWT programming concept because they are the newest and more important concepts treated on this thesis. After that, it starts the analysis part; the whole application is described in this chapter. It includes the communication diagram of all the components that are part of this system and the explanation of how they interact to provide the required functionality for this program. In addition to this, it presents the system architecture with all the packages that are part of project, and an explanation of their purpose and functionalities. Finally, there has been designed a use case diagram and a real use case as an example of the functionality provided by the system. In the problems found chapter there are described and analyzed the three main problems appeared during the implementation of the application. This application is in the first version, so there are lots of improvements that can be done and they are described in the future work section. Conclusion part, extracts the important conclusions reached with this project.

CHAPTER 2
STATE OF THE ART

In this chapter it is going to be described what a Sensor Network is and also the concepts of Sensor Web Enablement which constitute the background of this project. The basis of the application is to provide communication between a client and a server based in Sensor Observation Service (SOS) standard, so it is also necessary to analyze this standard for having a better understanding of which are the operations that compose it, how they retrieve the data from the SOS database and how can this give a better comprehension of the system's communication model.

To complete this phase, it is also interesting to have a look over the real implementations and applications that are using SWE standards including SOS, to get an idea the large number of different fields in which these standards can be applied.

2.1 Sensor Network, Sensor Web and SWE

In this sub-section it is explained the key concepts of sensor networking, sensor web and SWE focusing on its advantages, features and components and the services they can offer to be applied in the real world.

2.1.1 Sensor Network

Sensor networks are composed of many spatially distributed sensors. These networks are especially well suited for environmental monitoring such as temperature, sound, vibration, pressure or air pollutants. Each node (or sensor) in the network communicates with its server using wireless technology and the access to the data is performed using standard protocols. The development of sensor networks requires technologies from three different research areas: sensing, communication, and computing (including hardware, software, and algorithms). (Chong, 2003).

A sensor network design is influenced by many factors, which include fault tolerance; scalability; production costs; operating environment; sensor network topology; hardware constraints; transmission media; and power consumption. (Azquiyadiz et al. 2002):

- **Fault Tolerance:** It refers to the ability of the net to continue with the overall task when a sensor node fails. It means that the sensor network can't be influenced by the failure of a sensor.

- **Scalability:** sensor networks can be formed by hundreds or thousands sensors. A good sensor network schema should be able to handle this amount of nodes.

- **Production Costs:** Sensor networks have many nodes therefore the cost of one node should be low to justify the creation of a network instead of deploying traditional sensors.

- **Operating environment:** It must be taken into account that the nodes of a sensor network could work under very hard conditions, for example: at the bottom of the ocean under high pressure, or at deserts or polar areas with intense hot and cold.

- **Sensor Network Topology:** deploying high number of nodes densely requires careful handling of topology maintenance.

- **Hardware Constraints:** Often sensors are located in areas where it is very difficult to access so they must consume extremely low power and they must be autonomous and able to operate unattended.

- **Transmission Media:** To enable the global operation of sensor networks, the chosen transmission medium must be available worldwide. The links in a wireless sensor network can be formed by radio, infrared or optical media.

- **Power Consumption**: Sensors can only been equipped with a limited power source, and sometimes is impossible to refill it, so it is very important to consume as less energy as possible.

Looking at the previous issues of sensor networks it is possible to guess which will be some of the advantages they have over traditional centralized approaches:

- Low energy consumption
- Scalability
- Cheaper than traditional wired network
- Ideal for non-reachable areas
- It can be accessed through a centralized monitor

Next figure shows a very basic schema of a wireless sensor network (Román R, Javier L. 2009):

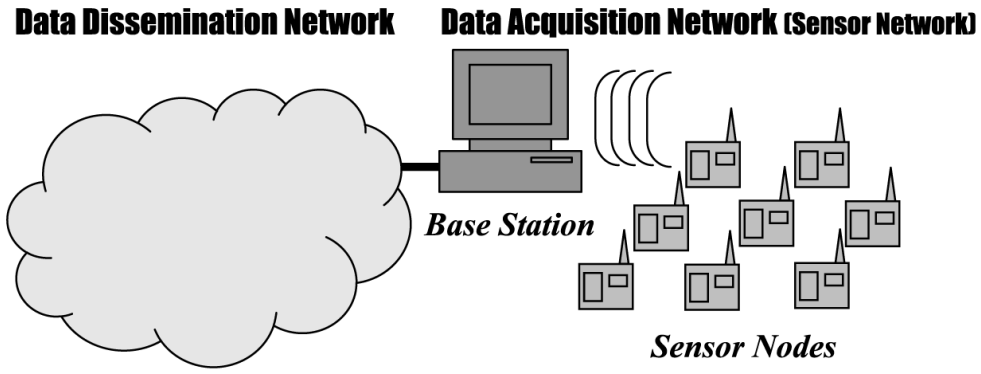

Figure 1: Wireless sensor network schema

2.1.2 Sensor Web

Sensor web is a kind of sensor network which is web accessible by Application Program Interfaces (APIs) and using standard protocols. In general, the Sensor Web is a macro-instrument comprising a number of sensor platforms (OGC, 2006). These platforms can be orbital or terrestrial, fixed or mobile: This system would be embedded into an environment to monitor and even control it. The purpose of a sensor web system is to extract knowledge from the data it collects and use this information to intelligently react and adapt to its surroundings. (Kevin A. 2011)

It links a remote end user's cognizance with the observed environment (Kevin A. 2011).

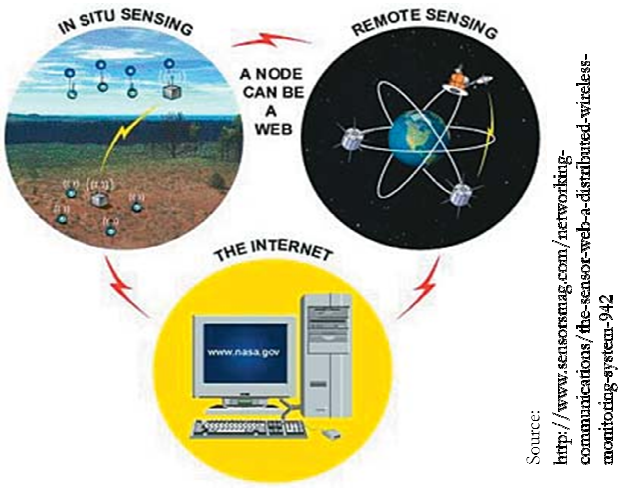

Figure 2: Sensor Web System communication schema

2.1.3 Sensor Web Enablement

The Open Geospatial Consortium‘s Sensor Web Enablement (SWE) activities, refers to web accessible sensor networks and also to a stored sensor data that can be discovered and accessed using standard protocols. These activities have been executed mainly by the OGC Web Services (OWS) initiatives, under the Interoperability Program which is setting the interfaces and protocols that will be enable sensor webs through which applications and services will be able to access sensors of all types over networks such as the Internet and with the same standard technologies and protocols that enable the Web. (OGC, 2007)

OGC defines the SWE family of components as the following one:

- **Observation & Measurements (O&M)**: General models and XML for encoding, (archived and real-time) sensor observations and measurements.

- **Sensor Alert Service (SAS):** A service by which a client can register for and receive sensor alert messages. This service supports alert publication, subscription and notification.

- **Sensor Model Language (SensorML):** General models and XML schema for describing sensors and processes associated with measurement. Provides information needed for discovery of sensors, location of sensor observations, low-level sensor observations processing and list properties.

- **Sensor Planning Service (SPS):** Service by which a client can determine collection feasibility for a desired set of collection requests for one or more sensors/platforms, or a client may submit collections requests directly to these sensors/platforms.

- **Transducer Markup Language (TML):** General characterizations of transducers (receivers and transmitters), their data, how that data is generated, the phenomenon being measured by of produced by transducers, transporting the data, and any and all support data (metadata) necessary for later processing and understanding of the transducer data.

- **Web Notification Service (WNS):** Service by which a client may conduct asynchronous dialogues with one or more other services. It supports SPS (Sensor Planning Service) operations.

- **Sensor Observation Service (SOS):** provides an API for managing deployed sensors and retrieving sensor data and specifically *observation* data. This standard is described in next section

In general, SWE is a suite of standard encodings and web services that enable:

- Sensors, processes and observations discovery
- Tasking of sensor or models
- Access to observations and observation streams
- Publish and subscribe capabilities for alerts
- Robust sensor systems and process descriptions

2.2 Sensor Observation Service

SOS is a standard defined by the OGC who provides an API for managing deployed sensors and retrieving sensor data and specifically *observation* data.The goal of SOS is to 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, 2007). SOS uses the O&M specification for encoding the observations and measurements from a sensor, and the SensorML specification for sensor description or sensor systems.

A SOS organizes collections of related sensor system observations into *Observation Offerings* that can be defined as a group of observations offered by a service that are related in some way (OGC, 2007). Each Observation Offering is constrained by these parameters (OGC, 2007):

- Specific sensor systems that report the observations
- Time period(s) for which observations may be requested
- Phenomena that are being sensed
- Geographical region that contains the sensors
- Geographical region that contains the features that are the subject of the sensor observations

According with the standard, the operations of SOS are classified in three profiles: Core, Transactional and Enhanced profile. Then it can be defined a *special* Entire profile encompassing all operations defined in the standard.

Each operation request consists in a XML file that the client sends to the server. Each operation needs specific parameters which are defined in the standard document (OGC, 2007). Some of these parameters are mandatory while others are optional. For having an well-formed XML request, a part of the corresponding headers, it is also necessary to include at least, the information corresponding to the mandatory parameters; otherwise, the server will return an exception report.

For a better understanding the next figure shows the structure of the operations defined in the SOS standard.

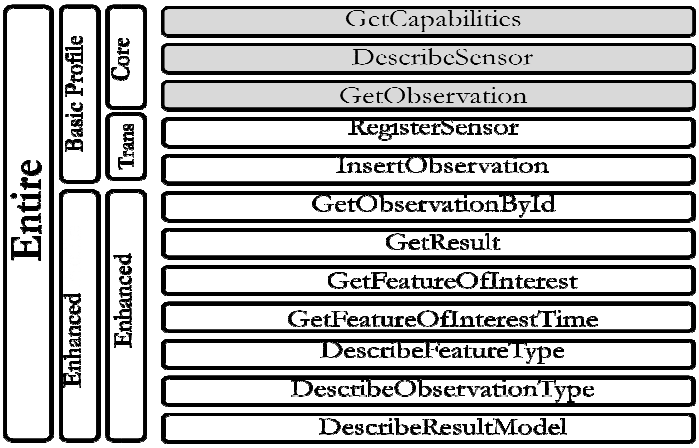

Figure 3: SOS standard's operations structure

These are all the operations specified in the standard document for retrieving data from sensors. And it is also important to indicate that some of these operations are mandatory and the other ones are optional. This means that a SOS service must have implemented at least all the mandatory operations, which correspond to the ones in the Core profile: GetCapabilities, DescribeSensor and GetObservation.

2.2.1 Core profile

- **GetCapabilities:** This operation allows clients to retrieve service metadata about a specific service instance.
- **GetObservation:** This operation is designed to query a service to retrieve observation data structured according to the O&M specification.
- **DescribeSensor:** Designed to request detailed sensor metadata encoded in either SensorML of TML. The sensor characteristics can include lists and definitions of observables supported by the sensor.

2.2.2 Transactional Profile

- **RegisterSensor:** This operation allows the client to register a new sensor system with the SOS as part of the transactional profile. This operation is mandatory for the transactional profile.

- **InsertObservation:** This operation allows the client to insert new observations for a sensor system.

2.2.3 Enhanced Profile

- **GetObservationById:** It is designed to return an observation based on an identifier field.
- **GetResult:** The purpose of the GetResult operation is to allow a client to repeatedly obtain sensor data from the same set of sensors without having to send and receive requests and responses that largely contain the same data except for a new timestamp. This operation uses much less bandwidth than would be necessary for a full GetObservation call.
- **GetFeatureOfInterest:** Returns a Feature of Interest that was advertised in one of the observation offerings of the SOS capabilities document.
- **GetFeatureOfInterestTime**: Returns the time periods for which the SOS will return data for a given advertised Feature of Interest.
- **DescribeFeatureType:** Returns the XML schema for the specified GML feature advertised on GetCapabilities.
- **DescribeObservationType:** This operation returns the XML schema that describes the Observation type that is returned for a particular phenomenon. This allows the SOS to list the set of Observation types that it can deliver.
- **DescribeResultModel**: Returns the schema for the result element what will be returned when the client asks for the given result model by the given ResultName.

2.3 Sensor Network Applications

The applications of sensor networks are numerous involving very different fields like health care, military matters, environmental monitoring or transport systems and the actions that are normally carried out by these sensor networks involve tasks like monitoring, tracking or controlling. Typically, Sensor Networks Applications consist in deploying a sensor network in the region where it is meant to collect data through its sensor nodes. In this section some of these applications are described to know how sensor networks work when applied to real cases in the real world.

During the last years, there has been a considerable evolution in the research of sensor networks due to the recent progress in computing and communication systems. Nowadays, it is possible to produce very small sensors with a very low cost, even equipped with inexpensive low-power processors that give them long periods of autonomy, allowing the development of wireless sensor networks and *ad-hoc* networks for numerous applications covering a very wide range of disciplines.

In the following sub sections there are shown different disciplines where sensor networks can be applied.

2.3.1 Military and defense applications

These applications were the first to appear and they have served as the basis for future development of sensor networks. At present, some of them still are used.

- In the Cold War *SOund SUrveillance System* (SOSUS), a system of acoustic sensors on the ocean bottom was deployed at strategic locations to detect and track quiet Soviet submarines. SOSUS is now used by the National Oceanographic and Atmospheric Administration (NOAA) for monitoring events in the ocean, like seismic and animal activity. (Chong, 2003)

- Also during the Cold War, networks of air defense radars were developed and deployed to defend the continental United States and Canada. This air defense system has evolved over the years to include aerostats as sensors and Airborne

Warning and Control System (AWACS) planes, and is also used for drug interdiction. (Chong, 2003)

- Modern research on sensor networks started around 1980 with the Distributed Sensor Networks (DSN) program at the Defense Advanced Research Projects Agency (DARPA). (Chong, 2003).

Now sensor networks (mainly wireless sensor networks) are an integral part of military command, control, communications, computing, intelligence, surveillance, reconnaissance and targeting systems.

2.3.2 Environmental Monitoring

In this type of applications, usually the variables to be monitored are intrinsically distributed in the space, one example of this, can be the temperature or the wind speed. This fact makes sensor network something very suitable for this type of monitoring.

Examples of these networks are:

- **Glacsweb:** developed by the Southampton University is a technology to monitor glacier behaviour to understand climate change and its effect on sea level rise using sensor networks (Glacsweb, 2011).

- **Center for Embedded Network Sensing (CENS):** Is a multidisciplinary centre, where people are participating for a wide range of fields including, among others, computer science, environmental engineering, urban planning or biology. They are involved in projects for studying vegetation response to climatic changes and diseases and also one of their activities is the development and deployment of new measurement tools and techniques to identify the sources and fates of chemical and biological pollutants in natural, urban, agricultural watersheds and coastal zones (CENS, 2011).

- **WFS Technologies:** WFS's wireless technology enhances monitoring systems in the energy and environmental sector by enabling data transmission through-

water and through-ground in "real time" (WFS, 2011). They apply their methods in different fields like Oceanography and environmental management and their activities in those areas are reservoir monitoring, hydrometry, flood and water quality monitoring.

- **FLOODNET** is a warning system to make flood predictions based of readings of water level collected by a set of sensors nodes. One of the objectives of this project is to improve warning times to minimize flood damages (Zhou J, De Roure D. 2006).

2.3.3 Habitat monitoring

Researchers in the life sciences are becoming increasingly concerned about the potential impacts of human presence in monitoring plants and animals in field conditions (Mainwaring A. Et al. 2002). Sensor networks, especially wireless sensor networks, are a good alternative to the invasive methods.

This kind of networks can be incorporated on the field without any distortion of the results due to the human presence, so, sensor networks represent a decisive advance over traditional and invasive methods, and also the cost for long-term studies is significantly reduced because, as said before, there are less people involved in taking measurements and observations directly on the field.

One example of these kind of application is the deployment of three wireless sensor networks for habitat monitoring on Great Duck Island, about 25 km of the Coast of Maine, to monitor the Leach's Storm Petrel. The networks monitored data of temperature, humidity, occupancy and pressure to correlate nesting patterns with microclimates (Szewczyk R et al. 2004).

2.4 Sensor Web Enablement Applications

In this section it is going to be shown some real implementations following the SWE model. As SWE is a family of standards, they can be used in many situations and for many purposes because it allows the user to access, discover and use data from any kind

of sensors via web. This fact makes the SWE applications very useful and convenient in a very wide range of areas. Here, there are presented some of these applications to have a general overview of where they are implemented and what are they used for.

2.4.1 EO2HEAVEN

One of the projects using SWE standards is EO2HEAVEN (Earth Observation and ENVironmental modelling for the mitigation of HEAlth risks), which its objective is to contribute to a better understanding of the complex relationships between environmental changes and their impact on the human health. The project will monitor changes induced by human activities, with emphasis on atmospheric, river, lake and coastal marine pollution and integrates both remote and in-situ environmental measurements. (EO2HEAVEN, 2011).

People coming from multiple disciplines have designed a GIS Application based in an open and standard Spatial Information Infrastructure (SII) to assist in the early detection of potential health endangerments. The study cases of this project are located in Uganda: for investigating the impact of climatic variables on the outbreak of cholera, Durban (South Africa): for dealing with the pollution and respiratory diseases and Dresden (Germany): addressing the environment effects on allergies and cardiovascular diseases (EO2HEAVEN, 2011).

2.4.2 Osiris

Osiris is the acronym for Open architecture for Smart and Interoperable networks in Risk management based on In-situ Sensors.

OSIRIS provides a Service Oriented Architecture based on standards and delivering functions ranging from in-situ earth observation to user services. The programme is structured around four key areas of major environmental risk: forest fires, industrial risks, unexpected fresh water pollution and air pollution in urban areas, and their experiments take into account these four factors: environmental concerns, time constraints, type of sensors and data produced by the sensors.

Addressing smart deployment, use and reconfiguration of in-situ sensor systems, the OSIRIS proposed architecture is scalable to allow for easy integration of new sensor data to improve the quality of service (Osiris, 2011).

One of the implemented OSIRIS applications is the:

Home Weather Station System

For the development of this application there has been necessary to include the following components of the SWE Framework:

- SOS standard: which allows requesting data measured by sensors
- O&M standard: used for returning the sensor data in a standard way
- SensorML standard: used to retrieve information about the sensors of how the measurement process was performed

The objective of this application is to publish and share the data corresponding to the measurements obtained by a home weather station.

The next figure represents an example using this application:

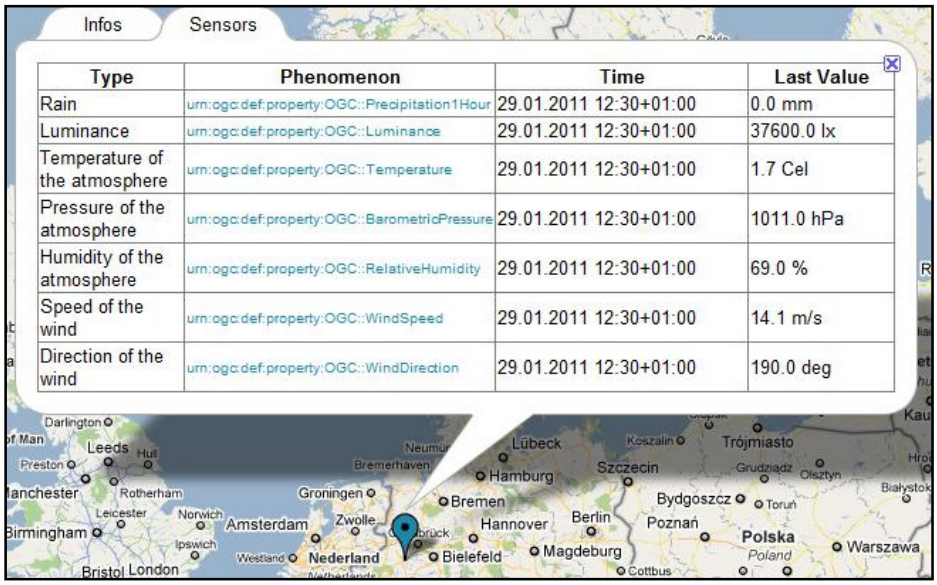

Figure 4: Sensor information displayed when clicking over a weather station

2.4.3 NASA

The United States National and Space Administration (NASA) are working in some projects to improve sensor web technology in satellites which are using SensorML standard for geolocalization and other purposes. Some of these projects are base in the use of OGC's SWE suite of standards and they are using Earth Observing 1 (EO-1) together with some other satellites to create sensor web applications (Bacharach S. 2011).

Nowadays, NASA is using SWE standards to standardize and simplify the process of sending commands to satellites. According to Dan Mandl of Goddard Space Flight Center (GSFC): "Software services based on open architectures have opened the door to standardized tasking of Earth imaging satellites". With SWE standards, ongoing experiments indicate that NASA can make better use of scarce resources by enabling users to focus on the data products they need rather than on complex satellite control procedures.

2.4.4 OOSTethys

OOSTethys is a group of software developers who are developing together with marine scientists open source tools to integrate ocean observing systems, in order to reduce the time needed to install, adopt and update standards-compliant web services. They work with standard organizations, and its aim is to develop free software based on standards to work with over 1000 platforms with real-time data they have. With this methodology, they provide an interoperable network that can be accessed via Internet (OOSTethys 2011).

In the next figure there can be seen the OOStethys architecture which has 6 modules, and the relationships among these modules:

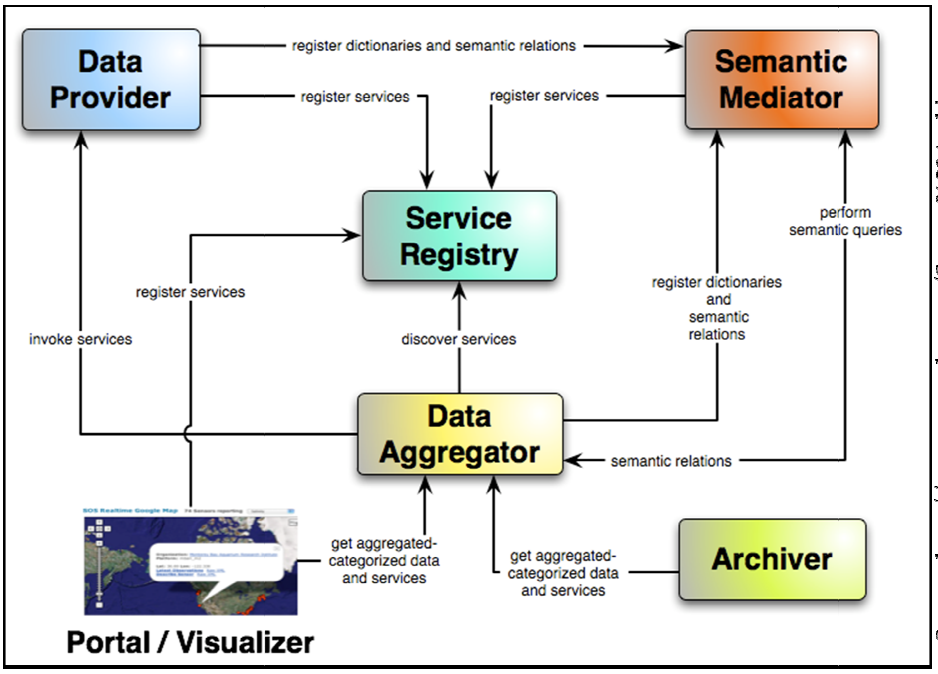

Figure 5: OOSTethys service architecture

- **Data provider:** Refers to the elements that provide the direct observations to be inserted in the database system. In this group it can be included in-situ sensors, mobile sensors or samplers. These elements also will insert the data into the database in a standard way. Up to this moment OOSTethys works with the SOS standard from OGC, and the operations currently supported are: GetCapabilities, DescribeSensor and GetObservation.
- **Semantic Mediator:** this module works for recording new vocabulary or mapping some expressions to a predefined one.
- **Service Registry:** Lets the registration in the OOSTethys service to other service providers, and allows also accessing the data.
- **Data Aggregator:** Can temporary store data coming from the data provider. It also can manage the data to transform it in useful information for the user so, it can be embedded in a visualization portal.
- **Data Archiver:** Is a data storage site that gets the data from data providers and data aggregator.
- **Visualization Portal:** Is the web portal from where the user can interact with the system and perform operations. In this portal there is information about the data providers and developers and also, it is an embedded map based in Google Maps showing all the platforms which are sending data (using the SOS standard) to the OOSTethys system.

OOSTethys provides lots of possibilities for filtering the data stored in its systems. Next figure represents an example of a possible output when clicking over a data center.

Figure 6: Information displayed when clicking over a data center

As seen, there is a big amount of data stored in OOSTethys platforms following the SOS standard, and many analyses can be performed. Also OOSTethys involves lots of people and organizations around the world in an interoperable system promoting and encouraging the use of these technologies.

2.5 Summary

In this section it has been presented the concepts of sensor network, sensor web and SWE which are the most important concepts on which this project is based. It has been also presented the advantages and main features they have as well as a general description of SOS standard and the operations that compose it. In addition to this, there are presented real applications and projects of sensor networks and SWE including also real examples.

CHAPTER 3
ANALYSIS

Before starting a project it is necessary to make a previous analysis to define the requirements of the system and also the basic architecture in which the project is going to be based. In this section not only it is presented the system requirements analysis but also the use case model and the activities diagram for having a better understanding of the activities this system has to achieve and which are the basic workflows.

3.1 System requirements

The implemented system has to satisfy the next requirements:

- It must implement the seven operations specified in the SOS Standard version 1.0.0 for the Enhanced Profile. These operations are: *GetObservationById, GetResult, GetFeatureOfInterest, GetFeatureOfInterestTime, DescribeFeatureType, DescribeObservationType* and *DescribeResultModel*.
- It has to provide a user interface based on a thin-client architecture.
- Follow the GWT package structure which differentiates among client-side code and server-side code.
- Implement the RPC mechanism for sharing attributes through the client-side code and the server-side code
- The user interface has to fulfill these requirements:
    - The main layout should contain these elements:
        - Title
        - Drop-down list box for selecting the server
        - Drop-down list box for selecting the operation of the enhanced profile
        - Map based on Google maps centered in France with three marks pointing the three Universities participating in this Master: Universitat Jaume I (Castelló), Universidade Nova (Lisbon) and University of Münster (Münster).

- For each operation it has to be provided a window to fill out with the parameters

- It has to support a successful connection with a SOS Server:
  - Create a SOS request
  - Connect with the SOS server
  - Save the response on a xml file
- The response has to be parsed to extract the useful information for being depicted on a map based in Google Maps.

3.2 Use Case Model

Next figure represents a use case diagram for the application:

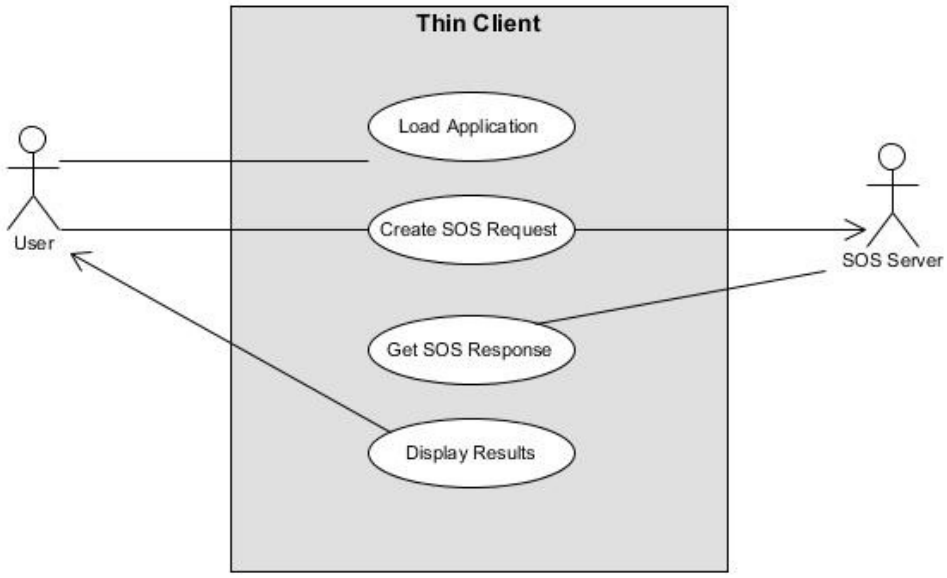

Figure 7: Use Case Diagram

This is a very simple use case model where only four use cases and two actors are involved:

- **Load application:** Starts when the user loads the application in a web browser.
- **Create SOS request:** This use case involves the activities of selecting an operation from the SOS standard enhanced profile, writing the required parameters, create a SOS request with this information and send it to the SOS server.
- **GetResponse:** The SOS server sends the response file to the client. This response is stored in the local file system.

- **Display results:** The results obtained are going to be displayed on a map, if the response has geographical information associated or in a text label if not. This use case includes also the parsing process of the SOS response.

As seen, the two actors interacting with this application are the user and the SOS server:

- **User:** The user roll is to load the application from a browser and select one operation to be done. The application shows then the results to the user.
- **SOS Server:** The thin-client sends the request to the SOS server and based on this request, the server creates a response and sends it back to the client.

3.3 Activities diagram

In a typical workflow, the user launches the application in a web browser. Once the application is loaded, there is a list of servers that can be selected and a list of operations corresponding to the enhanced profile of SOS. When the server and the operation are selected, then the program shows a new window consisting in a form to be filled out with the correct parameters required for the specified operation. If the parameters are correct, the client sends the request to the server and the server respond sending a response in O&M format. The parser now searches and extracts the useful information to be depicted on a map.

It is important to note that in some cases, the operation **has no** geographic information associated so the results can't be depicted on a map, and then the program returns the xml file and shows the results as a text field on the screen.

These are the major activities that can be extracted from the application. The next diagram consists in a graphical representation of the workflow explained before.

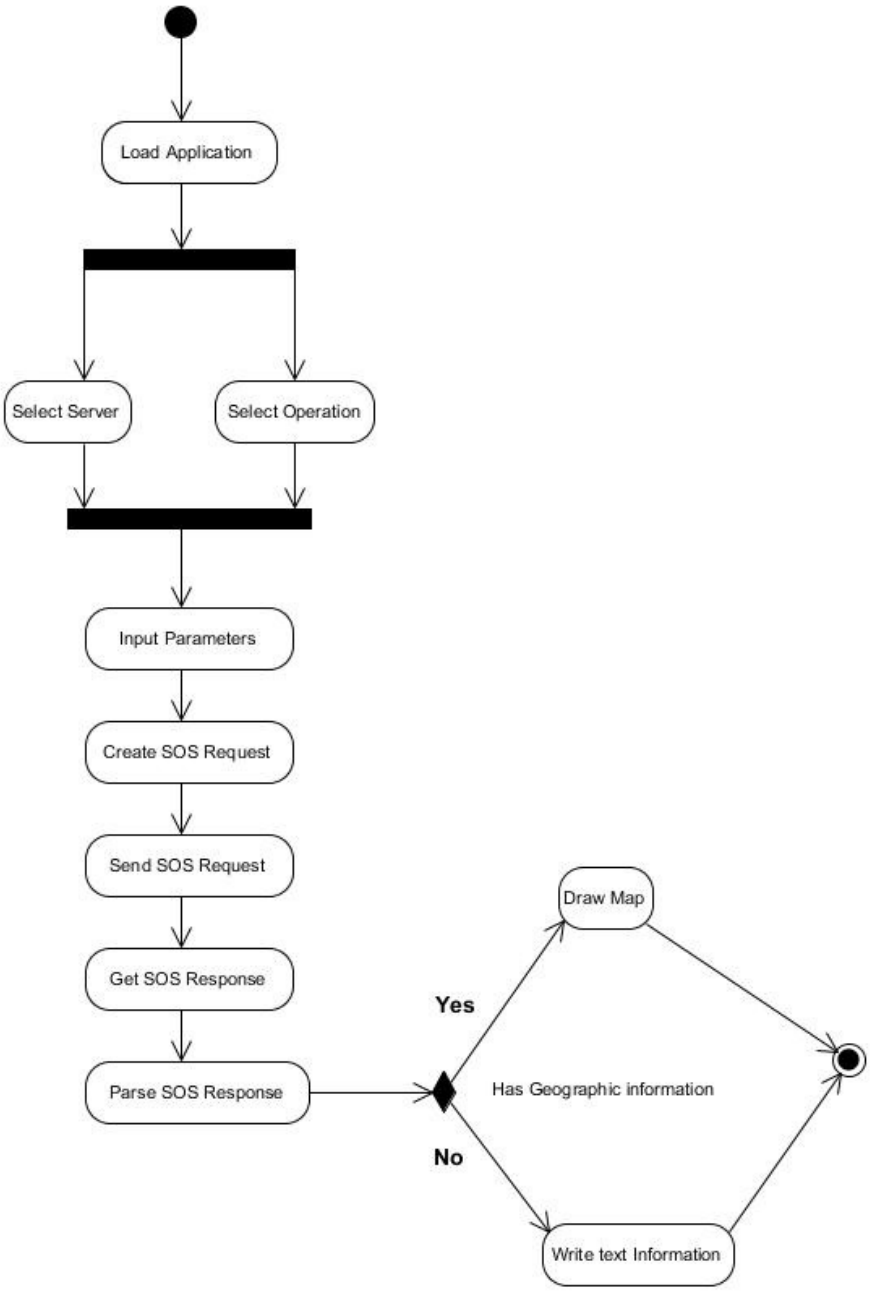

Figure 8: Activities diagram

3.4 High Level Architecture

The architecture of this system is mainly based in the division of the code between the client-side and the server-side as shown in figure X. This fact is due to the use of GWT for programming this thin-client. Following this paradigm, the translatable code to Javascript must be located in the client-side and the implementation, or not-translatable code must be located in the server.

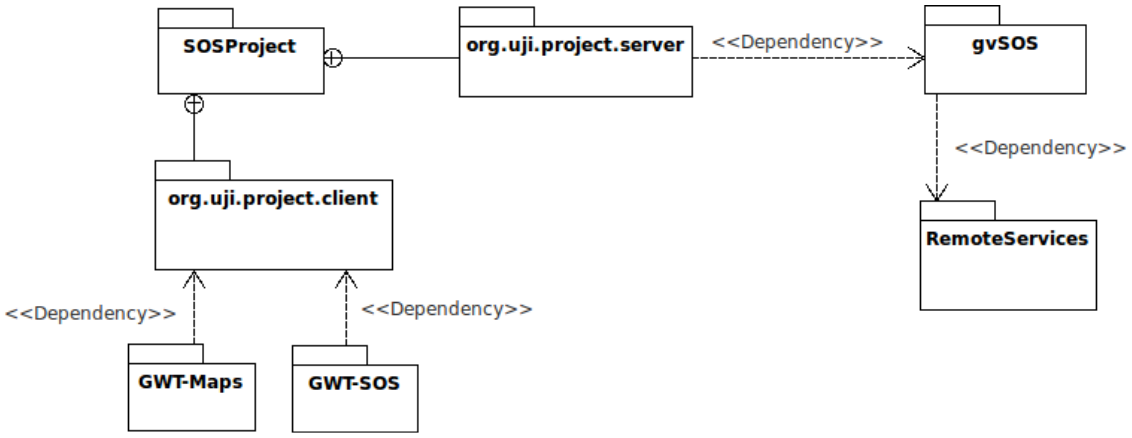

Figure 9: SOSProject Package Diagram

As can be seen in the previous figure, the client's side consists of two logical packages and also, there is a dependence with the extSOS and libRemoteServices packages.

- **GWT Maps:** this package represents all actions and classes that are dedicated to draw the maps, therefore, they will use code and libraries related with Google Maps.
- **GWT SOS:** Classes concerning the management of SOS request from the client side. From these classes are included the necessary steps to perform the RPC for sending the parameters.
- **gvSOS:** This directory is closely related with SOSProject SRC directory. gvSOS has the ability to correctly create SOS requests and connect with a SOS server when its url is given. The inclusion of this package with these functionalities is necessary and makes it possible the communication with the server.
- **libremoteServices:** this package contains the low level elements used in the communication with the server.

3.5 Summary

In this section an analysis has been done. First, there are presented the system requirements that the system has to accomplish. There have been defined two diagrams: use case model and activities diagram; with this diagrams and looking at the information they contain it is possible to recognize the general activities the system is going to implement and also the overall workflow of the actions. The general architecture is presented as a package composed of two layers or parts (client and server) that have an important dependency with the extSOS package.

CHAPTER 4
DESIGN AND IMPLEMENTATION

After the design phase of the system in this chapter it is going to be shown the implementation and design part. In this part it will be explained the structure of the packages and also the system communication diagram. This chapter finishes with the explanation of a real use case using some screenshots of the system.

4.1 Package structure

GWT projects are divided in different Java packages. In this case, under the main project directory there are the following directories:

- **src folder:** containing the java source code programmed for this applications
- **war folder**: which contains the static files in HTML or CSS like the host page, style sheet files or images
- **srcExtSOS:** Consisting in the source code for the SOS communication
- **srcLibRemoteServices:** contains tools to allow the communication together with the srcExtSOS

4.1.1 SRC Package

This package is the main folder where the programmed Java code is located. For this project, inside SRC directory there are three packages containing java classes that can be seen in the following image:

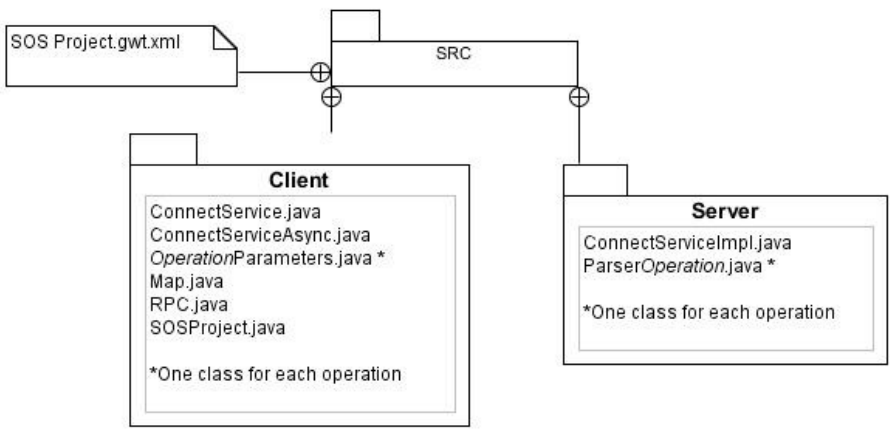

Figure 10: Structure of SRC package

The previous figure represents the package diagram of the SRC folder. There are two sub-packages *client* and *server* which refer to the client-side and server-side code respectively and they are part of the thin client application. It is important to not confuse the server package with the external SOS server.

- **org.uji.project :** It is the root package in the project and contains the *.gwt.xml* file. In this file it is determined the entry point for the application or the file that must run when the service is involved. The entry point of this project is the class SOSProject. In this file it is also specified the inheritance of the module of Google Maps: com.google.gwt.maps.GoogleMaps

- **org.uji.project.client:** This package contains all the classes of the project programmed with translatable code. This is the client side of the application and the main purpose of the package is to contain the methods for drawing the user interface. So the interface design, including forms, buttons, labels or text boxes has to be written in this package.

- **Org.uji.project.server:** This package contains all the classes of the project that are not directly translatable to Javascript. Here are located the processing actions in order to respond the client requests.

4.1.2 extSOS Package

This is an external folder linked with the project. extSOS was created and developed originally by Alain Tamayo (Tamayo A. 2009) and extended by Irene Garcia and Mauri Benedito (Benedito M., Garcia I. 2010) as a end-of-degree project.

The extSOS whole project is a gvSIG extension for connecting with SOS providers. The extSOS client extension allows gvSIG user to interact with SOS servers displaying the information gathered by sensors in a layer composed by features (Tamayo A. 2009).

In the original version of extSOS there are lots of packages related with gvSIG management which are not necessary in this project. The packages used for the

development of this application are only the ones related with the communication process with a SOS server and also the ones dedicated to creating the SOS requests for the operations. Next diagram shows the packages used for this project.

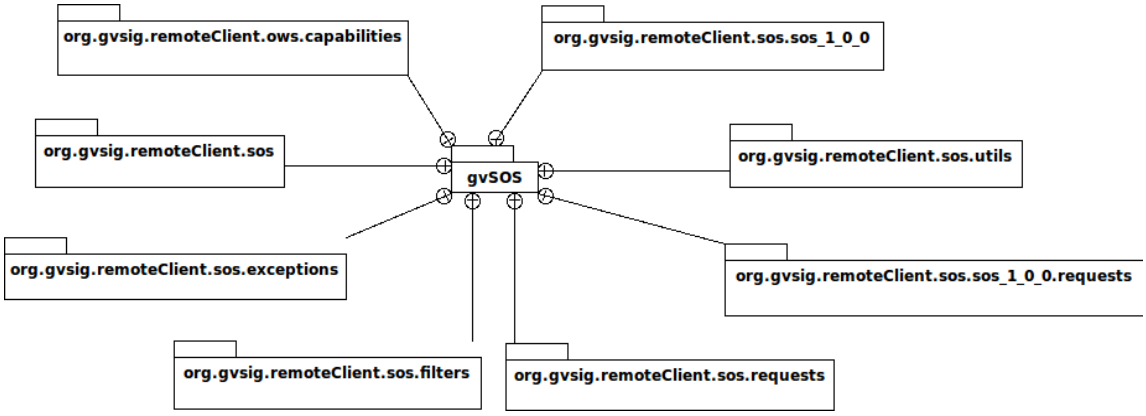

Figure 11: Structure of gvSOS Package

- **Org.gvsig.remoteClient.ows.capabilities:** contains the classes corresponding to all the possible objects appearing in the server capabilities document.

- **Org.gvsig.remoteClient.sos:** Includes the functionality to connect with the SOS server and get the responses.

- **Org.gvsig.remoteClient.sos.exceptions:** This package has a list of exceptions that can be generated when a connection is running. It includes: send a request, wait for the response, retrieve the data on the server and get the response.

- **Org.gvsig.remoteClient.sos.filters:** contains spatial and temporal filters. It is used for filtering the SOS response.

- **Org.gvsig.remoteClient.sos.requests:** In this package there are the different abstract classes for sending and receiving SOS requests. There is one class for each operation and is version independent.

- **Org.gvsig.remoteClient.sos.sos_1_0_0:** contains a handler for creating the requests but exclusive for the version 1.0.0

- **Org.gvsig.remoteClient.sos.sos_1_0_0.requests:** It implements the requests for the version 1.0.0 of SOS. This package is the specialization of Org.gvsig.remoteClient.sos.requests.

- **Org.gvsig.remoteClient.sos.utils:** In this package there is only one class containing all tags corresponding with the static values that appear in the capabilities document.

4.1.3 LibRemoteServices Package

This library contains the core functions to connect with a remote server using HTTP protocol. This connection is generic and it is not only specific for SOS, and there are implemented the low-level aspects for getting a communication channel. Although libRemoteServices belongs to gvSIG, extSOS package described before uses it so, it is necessary to include it in the project.

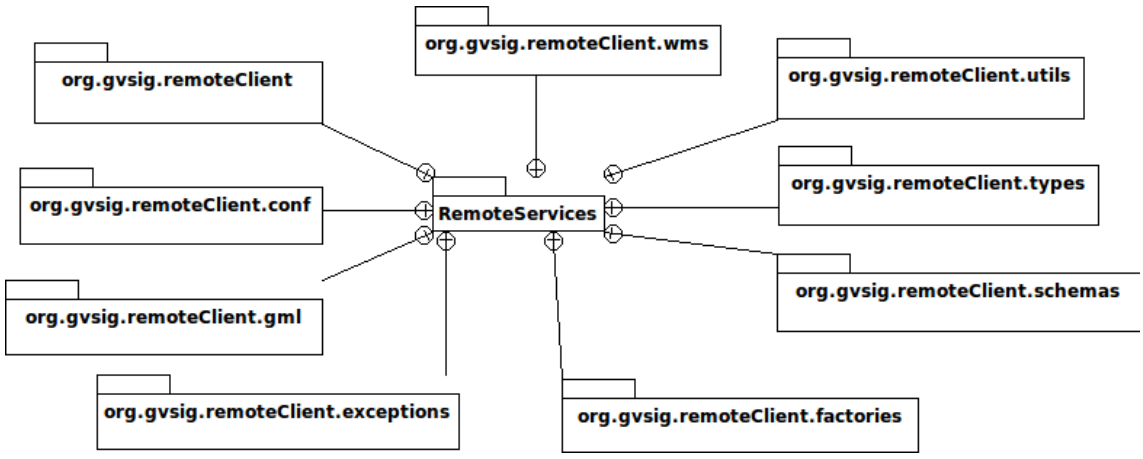

Figure 12: Structure of libRemoteServices Package

4.2 System communication diagram

Figure 13 shows the communication diagram of the system:

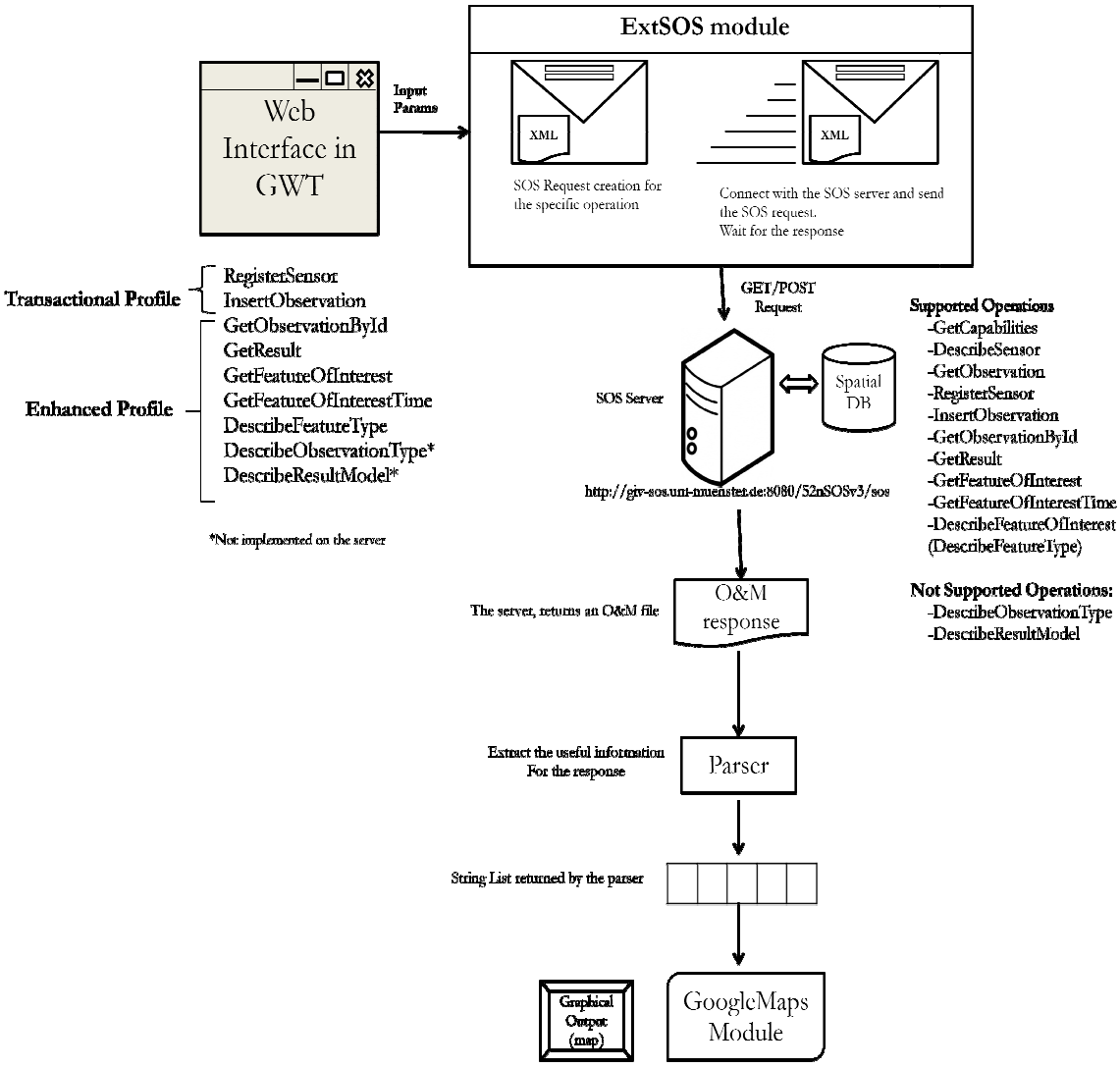

Figure 13: System communication diagram

At first instance, the user interacts with a GWT interface. As seen before, this interface allows, the selection of an operation corresponding to the SOS standard. The user then inserts the required parameters and click the accept button. Until this moment, the GWT interface is only working with the client part and it is necessary to create an RPC[1] connection to send the parameters to the server side, where all the process has to be done.

[1] RPC Mechanism described in section 4.3

Once in the server side, the next step is to create a SOS Request with the parameters specified by the user; in this part of the process it is needed an external source: the extSOS extension (Tamayo A. 2009, Garcia I. 2010 and Benedito M. 2010). As said before, extSOS extension has the necessary functions to connect with the SOS server, create a valid request with the parameters, send it to the server and store the response in an XML file. This communication is one of the essential parts in the project because is there where the SOS database is stored and all information is available to request it.

When a SOS request is received in the SOS server, it extracts the useful and significant information to check which operation is and which are the parameters involved. In this project, the server used is the one deployed by 52North and the operations accepted are: GetCapabilities, DescribeSensor, GetObservation, RegisterSensor, InsertObservation, GetObservationById, GetResult, GetFeatureOfInterest, GetFeatureOfInterestTime and DescribeFeatureType.

If the requested operation is one of the named above, the server searches in its spatial database for the necessary data and creates a response, in O&*M* format, following the standard rules to send it to the client.

Before sending the response to the client side, in GWT there is an "intermediate" parsing process[2]. As seen in previous sections, GWT applications differentiate between client and server sides; one of the goals of this application is to depict the information on a map based on Google Maps. For doing this, it is necessary to extract the geographic coordinates of the sensor and other useful information from the response file in O&*M* format. Thus, is necessary a parsing process that must to be done in the server side of GWT because this is a non-translatable code.

Once the parsing process is completed, the obtained information is sent again to the client side; there it is going to be depicted on a map as a result for the operation selected by the user.

The following diagram shows the logic flow of the system based in the logical sequence of the actions:

[2] Parsing process described in section 4.4

Figure 14: General Sequence diagram

4.3 RPC Mechanism

An application in GWT has two different parts: the client side and the server side and this structure is automatically created when deploying a new GWT project in any IDE. The code of the classes located at the client side must be translatable to Javascript, and the code on the classes located at the server side must not. So the classes for any application have to be placed in the correct side.

RPC is the methodology used for communicating client and server packages in a GWT application. RPC framework makes it easy for the client and server components of the web application to exchange Java objects over HTTP.

The java components of an RPC are three:

- The service that runs on the server (*ConnectServiceImpl.java*)
- The client code that invokes the service
- The Java data objects that pass between the client and server

In the next diagram it can be seen the sequence of the RPC mechanism in this project for the operation GetObservationById [3]:

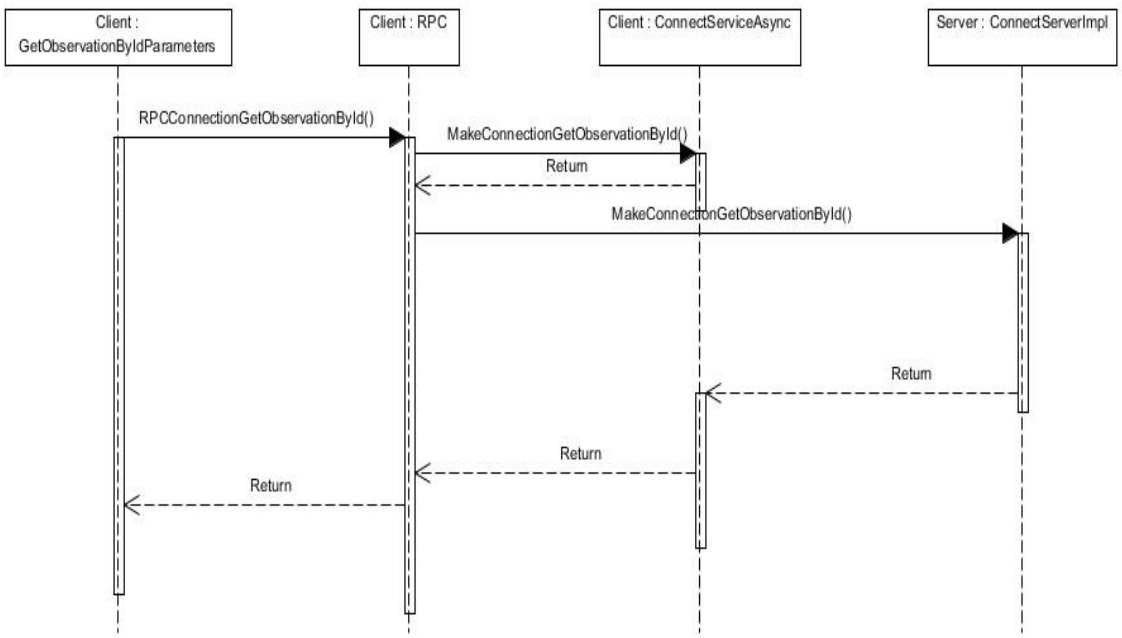

Figure 15: RPC sequence diagram

[3] GetObservationById operation is an example. All operations follow the same mechanism

There are 4 classes involved in this RPC mechanism:

- GetObservationByIdParameters.java: this class contains a call to the RPC mechanism sending the parameters that are going to be passed to the server-side code.
- RPC.java: Represents the client-side call. In this class there are three steps to be done: First step is to create the client proxy. After creating the service interface it is casted to the asynchronous version of the interface because GWT generates automatically a proxy who implements the asynchronous interface. The second step is to create an asynchronous *callback* to handle the result. Finally the third one is to make the call.
- ConnectServiceAsync.java: Contains the definition of an asynchronous interface to the service to be called from the client-side.
- ConnectServiceImpl.java: This class extends the RemoteServiceServlet and contains the implementation of the interface created in previous steps.

4.4 Parsing process

One of the functionalities of this application is that it has to depict on a map based in Google Maps, the geographical information contained in the response received by the SOS server.

As seen in previous sections, the response for one operation contains a field of information indicating the coordinates where the sensor is placed, but not only this information is useful; the O&M file obtained from the SOS server has some other parameters which are important in order to obtain a satisfactory result. This file also contains the numeric value of one observation, the name of the sensors or the offerings id.

The purpose of this part of the application is to get all this data for using it when the map is depicted. As a result, the map will show a mark located in the coordinates extracted from the response file and this mark will have attached a globe with the other necessary information that does not have geographical reference. At this point, it is necessary to use a parser.

The parser used for this application is XPath due to its simplicity and because it can be included in the project without making major changes, just creating some new classes dedicated to the parsing. This tool searches through the document for the specified labels, and returns the value which is contained on these labels. XPath is not the best option for parsing large files on big projects, because it is not efficient but in this project, the O&M files that XPath has to parse are not very large, so the inefficiency in this case is not that important.

Next diagram represents the parsing process for a GetObservationByIdResponse:

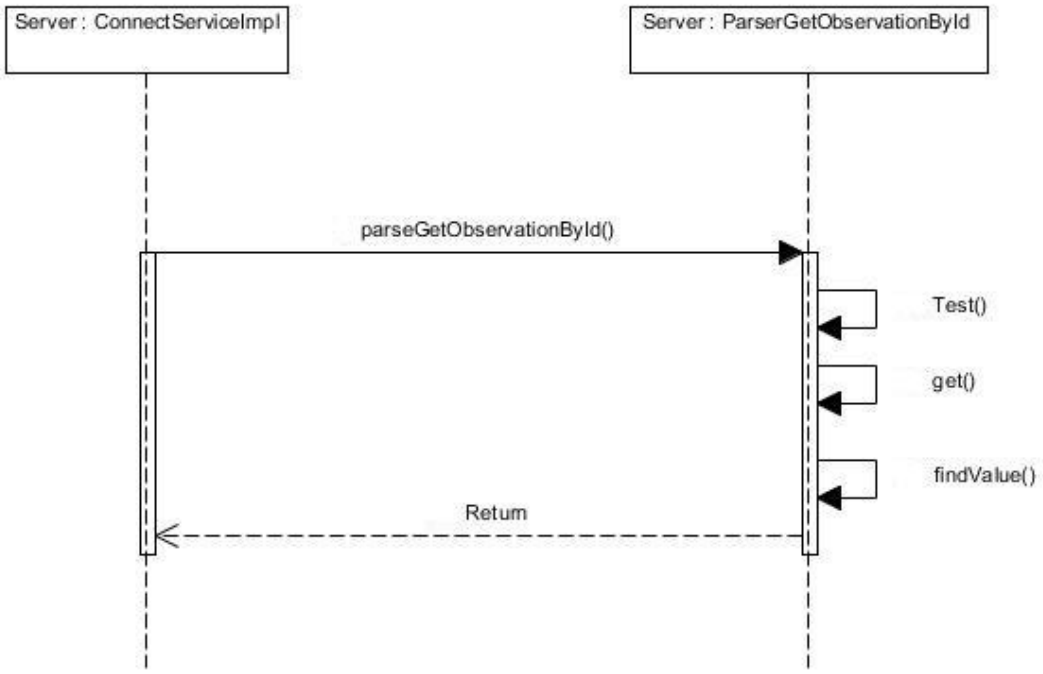

Figure 16: Parser Process sequence diagram

4.5 Use Case example

In this section it is going to be presented a use case example using the resulting interface designed on this project with real data coming from the SOS server deployed in the University of Münster.

This is the main screen showed when the user loads the application:

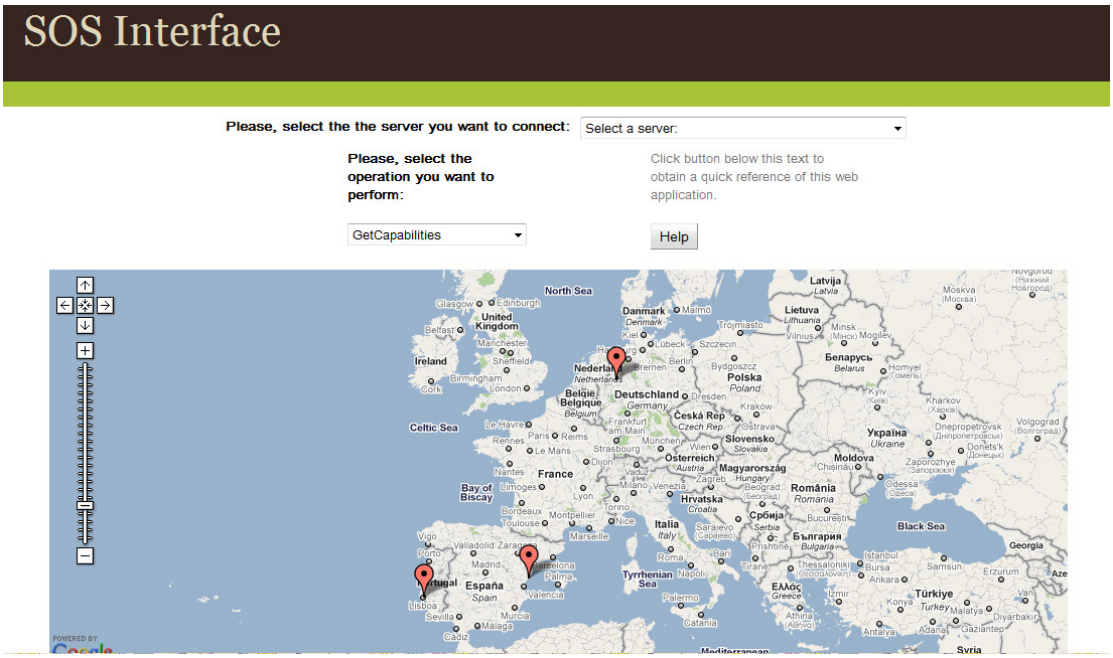

Figure 17: Application's main screen

The user selects the server and clicks on the drop-down list box to select the GetObservationById operation. After the selection of this operation, the application shows a window with the parameters that can accept the GetObservationById operation. The user inserts *o_9* as the id of the observation and *text/xml;subtype="om/1.0.0"* to specify the response format[4]:

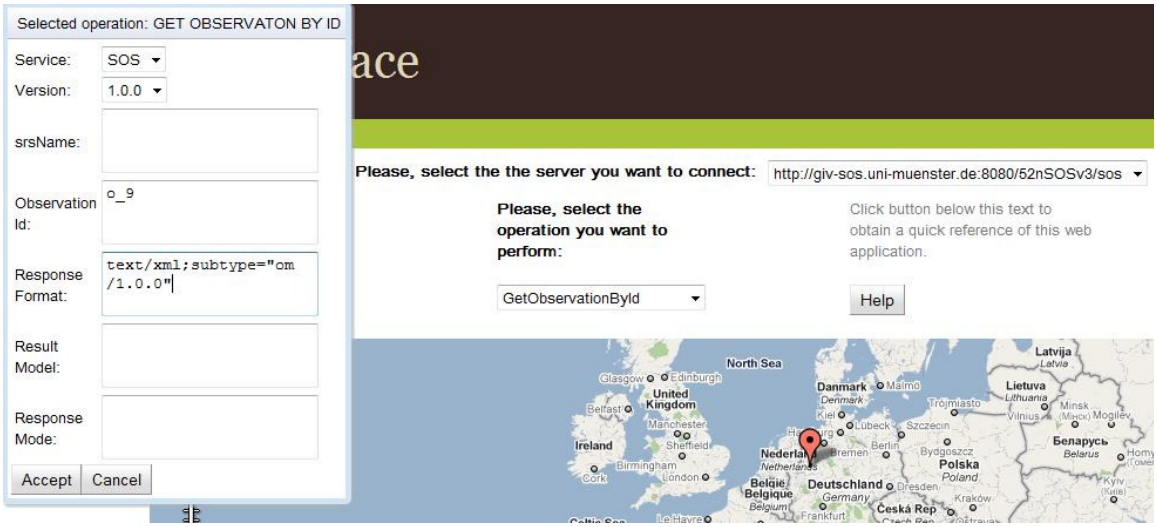

Figure 18: Parameters window for GetObservationById operation

[4] Note that in this example only the mandatory parameters for the getObservationById operation are defined

After accepting the operation the client waits for the server response and depicts a map as a result:

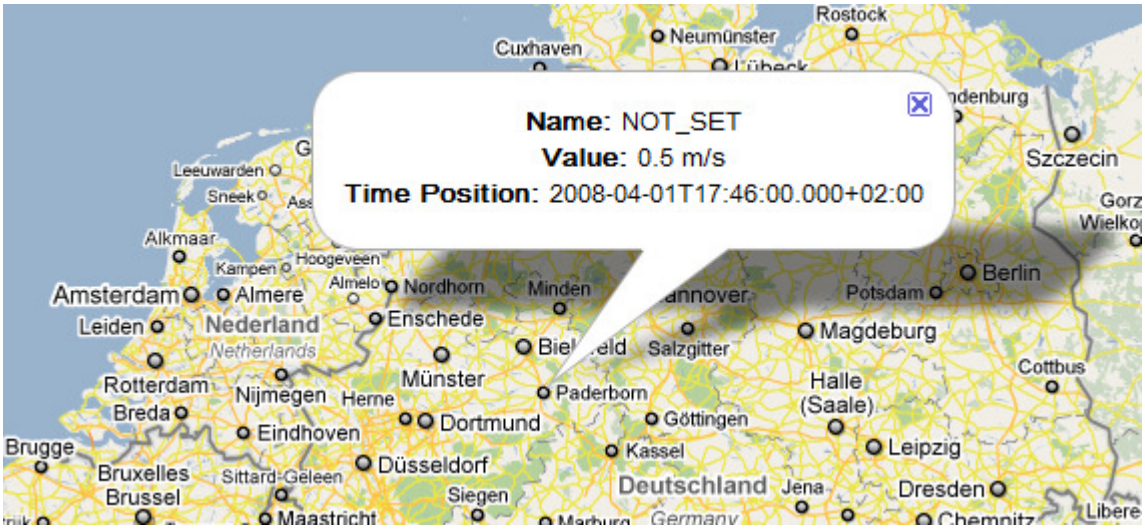

Figure 19: Result map for GetObservationById operation

In this map it can be seen that there is a balloon located in the coordinates of Padeborn city. The name of the sensor is not present in the server. The observation selected has been *o_9* and this one belongs to the feature of interest foi_2001 which is measuring the water speed, so the "value" in the balloon represents the water speed measured for this sensor. Time position represents the year, month, day, and hour when the observation was taken and also the time zone, in this case +2 hours.

As said in previous sections, there are operations that do not contain geographic information associated; the alternative of these ones is to show the data in a text field.

This is an example of the GetFeatureOfInterestTime operation result which does not contain geographic information in the response.

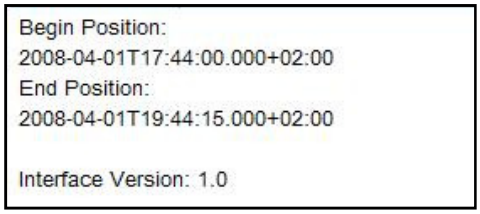

Figure 20: GetFeatureOfInterestTime result

4.6 Summary

In this section it has been presented the design and the implementation process of this project. Also, it is explained the RPC mechanism to send variables from the client-side to the server-side. To understand the structure of the project, some diagrams have been presented and also there is an example of a real use of the resulting application.

CHAPTER 5
FUTURE WORK

The purpose of this project is to obtain a functional web application based on a thin-client programmed in GWT able to connect with a SOS server and represent the information on a map. At this stage, the communication between the web application and the SOS server is successful, so the main objective is fulfilled. However, due to time restrictions there are some aspects, mostly related with the design that could be improved in the future.

5.1 Application design improvement

The actual design of the application contains the basic necessary information for the user to connect with the server using very simple window forms. Most of the times, the user experience when using a software is better if the design of the application is good. To achieve this goal, not only should the improved the quality of the screens, but also it is interesting to perform a usability analysis to obtain an optimal result that makes the application intuitive and easy to use.

5.2 Adding a KML parser

As mentioned in the previous section the INIT group of the UJI is working in the development of a parser from O&M to KML. When the new parsed is finished it will be replaced the XPath parser, which is the one used now. The reason is that in this application, the information is displayed in a Google Maps map which can interpret KML files. XPath is useful by now because it is simple to implement and, the inclusion of it in the project has been easy; at this moment the O&M files returned from the SOS server to parse are not very long and contain not very much info so, the inefficiency of Xpath is not a crucial aspect to take into account. In the future, it would be possible to add more SOS servers for requesting information then, the O&M files returned can be larger and more complicated to parse so the necessity of a KML parser is a fact. With it, the O&M file will be translated into KML and sent to the Google Map to interpret and depict it without major problems.

5.3 Adding operations

At this moment not all the operations corresponding to the enhanced profile of SOS are working. The implementation in the client side for these operations is done, but the server does not accept all them. Although DescribeObservationType, DescribeResultModel and DescribeFeatureType functionality is implemented in the client, the server does not accept them and it has been not possible to verify if they work properly. GetResult operation has a different way of working and the implementation of it in the client is pending.

5.4 Visualization improvement

This first version of the application simply shows in the map, a default red mark in the correct coordinates, with a white globe attached containing a plain text with the other significant information, like the observation value of the corresponding sensor. It would be a good option, to improve the visualization of the information in the map. For now, the information on the SOS database is not very large so, the result of one operation contains very few values. In the future, the communication is going to be done using different servers from which it can be obtained responses with more information. In this case, it is necessary to create tables for showing the information to the user or change the visualization in the globe including bold fonts or pictures.

CHAPTER 6
CONCLUSION

In this document, it has been presented the implementation process of a new GWT thin-client for working as a SOS service. The aim of this project has been to program a web based interface in GWT which allows the user to perform the operations corresponding to the SOS standard enhanced profile.

For the implementation of this application it has been used technologies like GWT or parsing tools. It is very convenient to have a previous knowledge about all technologies that are going to be used when programming an application; it can save lot of time trying to solve many problems.

Also, during the implementation, it was necessary to use some external packages already developed. In this case, the external packages allow the connection with the server following the SOS standard. The more comprehension of the content of these packages the more possibility of avoiding future difficulties.

This application is based in standards like SOS from OGC and it has been developed with using the GWT free technology, this makes that this application can be interoperable with any platform that uses the same standards and therefore, due to the use of code based in standards, the data discovery and access to the different SOS servers will be possible without using different technologies for each one.

REFERENCES

- (52 North, 2011) 52 North, *Homepage*. http://52north.org/. [9 January 2011].

- (Benedito M., Garcia I. 2010) Benedito, M and Garcia, I, *Implementation of the operations corresponding to Enhanced Profile in a SOS Client*, Degree thesis, 2009, University Jaume I.

- (Cooper RT., Collins CE. 2008) Cooper RT and Collins CE, *GWT in Practice*, Manning Publications, Cambridge 2008 U.S.A.

- (Google AppEngine 2011) Google App Engine, *Homepage*, http://code.google.com/intl/ca/appengine/. [9 January 2011].

- (Google GWT 2011a) Google Web Toolkit, *Building User Interfaces*, http://code.google.com/intl/ca/webtoolkit/doc/latest/DevGuideUi.html. [9 January 2011]

- (Google GWT 2011b) Google Web Toolkit, *Communicating with a Server*, http://code.google.com/intl/ca/webtoolkit/doc/latest/tutorial/clientserver.html. [20 January 2011]

- (Google GWT 2011c) Google Web Toolkit, *Organizing Projects*, http://code.google.com/webtoolkit/doc/1.6/DevGuideOrganizingProjects.html . [20 January 2011]

- (Google GWT 2011d) Google Web Toolkit, *Developer's Guide*, http://code.google.com/intl/ca/webtoolkit/doc/latest/DevGuide.html. [20 January 2011]

- (Google GWT 2011e) Google Web Toolkit, *Making Remote Procedure Calls*, http://code.google.com/intl/ca/webtoolkit/doc/latest/tutorial/RPC.html. [20 January 2011].

- (Google GWT 2011f) Google Web Toolkit Designer, *GWT Designer User Guide*, http://code.google.com/intl/ca/webtoolkit/tools/gwtdesigner/index.html. [20 January 2011].

- (Kerei 2010) Kereki, F, Essential *GWT: Building for the Web with Google Web Toolkit 2*, Pearson Education 2010, Boston, U.S.A.

- (OOSTethys 2011) OOSTethys, *Homepage,* http://www.oostethys.org/. [20 January 2011]

- (OGC, 2007) Open Geospatial Consortium. *Sensor Observation Service*. 1.0.0. Reference: OGC 06-009r6. 2007.

- (OGC, 2006) Open Geospatial Consortium. *OGC Sensor Web Enablement: Overview and High Level Architecture*. Reference: OGC 06-050r2. 2006

- (Sierra K., Bates B, 2005) Sierra, K and Bates B, *Head First Java*, O'Reilly Media, 2005, Sebastopol.U. S. A.

- (Tamayo A. 2009) Tamayo, Alain, *GVSOS: A new client for OGC SOS interface standard*. M. Sc. Thesis, University Jaume I, University of Münster, Universidade Nova de Lisboa.

- (52 North, 2010) 52 North, *ArcGIS SOS Extension*, http://52north.org/communities/sensorweb/clients/ArcGIS_SOS_Extension/index.html [10 December 2010]

- (Glacsweb, 2011) Glacsweb, *Homepage*, http://envisense.org/glacsweb/index.html [17 January 2011]

- (CENS, 2011) Center for Embedded Networked Sensing, *CENS: Embedding the Physical World*. http://research.cens.ucla.edu/about/ [17 January 2011]

- (WFS, 2011) WFS Technologies, *WFS Energy and Environment,* http://www.wfstech.com/index.php/environment/ [17 January 2011]

- (EO2HEAVEN, 2011) EO2HEAVEN, *Earth Observation and environmental modelling for the mitigation of health risks*. http://www.eo2heaven.org/node/2 [17 January 2011]

- (Osiris, 2011) OSIRIS, *Homepage*,.http://www.osiris-fp6.eu/. [17 January 2011]

- (Akyildiz et al., 2002) I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, *"Wireless sensor networks: a survey"*, *Computer Networks* 38, 393–422., 2002.

- (Kevin A. 2011) Kevin A. Delin, *Networking & Communications: The Sensor Web: A Distributed, Wireless Monitoring System :* http://www.sensorsmag.com/networking-communications/the-sensor-web-a-distributed-wireless-monitoring-system-942 [17 January 2011]

- (Bacharach S. 2011) Sam Bacharach, *Government/Military: New Implementations of OGC Sensor Web Enablement Standards:* http://www.sensorsmag.com/networking-communications/government-military/new-implementations-ogc-sensor-web-enablement-standards-1437 [20 January 2011]

- (Cantoria S. 2011) Ciel S. Cantoria, *Bright Hub: Thin Client review - The pros and cons of thin-client computing.* http://www.brighthub.com/environment/green-computing/articles/66417.aspx [20 January 2011]

- (Garret S. 2011) Sam Garret, *The advantages of thin-client computing.* http://www.ehow.com/list_6637572_advantages-thin-client-computing.html [20 January 2011]

- (gwt.org.ua 2011) Gwt.org.ua, *Overview of GWT application architecture based on GWT-PF* http://gwt.org.ua/en/documentation/architecture/ [9 January 2011]

- (Chaganti 2007) Prabhakar Chaganti, *Google Web Toolkit GWT Java AJAX Programming.* *A practical guide to Google Web Toolkit for creating AJAX applications with Java. Chapter 2: "Creating a New GWT Application"*, Packt Publishing. February 2007

- (Yang S. et al. 2003) S. Jae Yang et al. *Web Browsing Performance of Wireless Thinclient Computing.* Budapest, Hungary. May 2003

- (Kanter, J. 1999) Joel Kanter. *Understanding Thin-Client/Server Computing. Chapter 1: The Thin-Client/Server Computing Model.* 1999

- (Román R, Javier L. 2009) Rodrigo Roman and Javier Lopez. *Integrating wireless sensor networks and the internet: a security analysis.* Emerald. Málaga, Spain. 2009

- (Lewis F. L. 2004) F. L. Lewis. *Wireless Sensor Networks*. University of Texas. 2004

- (Jirka S. Et al) Simon Jirka, Arne Bröring, Christoph Stasch. *Applying OGC Sensor Web Enablement to Risk Monitoring and Disaster Management.* Münster, Germany.
- (Szewczyk R et al. 2004) Robert Szewczyk, Eric Osterweil, Joseph Polastre, Michael Hamilton, Alan Mainwaring, Deborah Estrin. *Habitat monitoring with sensor networks.* Comunications of the ACM. June 2004.

- (Mainwaring A. Et al. 2002) Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, John Anderson. *Wireless Sensor Networks for Habitat Monitoring.* Atlanta, Georgia, USA. September 2002.

- (Zhou J, De Roure D. 2006) Jing Zhou and David De Roure. *FloodNet: coupling adaptive sampling with energy aware routing in a Flood Warning System.* University of Southampton, U.K. September 2006.

- (Chong, 2003) Chee-Yee Chong and Srikanta P. Kumar. *Sensor Networks: Evolution, Opportunities, and Challenges.* August 2003

- (Bott M. Et al. 2008) Mike Bott, George Percivall, Carl Reed and John Davidson. *OGC Sensor Web Enablement: Overview and High Level Architecture.* 2008

- (Heinzelman W. Et al. 2004) Wendi B. Heinzelman, Amy L. Murphy, Hervaldo S. Carvalho, and Mark A. Perillo. *Middleware to Support Sensor Network Applications.* University of Rochester. February 2004

Masters Program in **Geospatial Technologies**

Supported by:
Education and Culture
ERASMUS MUNDUS