Development of Sensor Web Applications with Open Source Software

Arne Bröring[1],
Eike Hinderk Jürrens[1],
Simon Jirka[1],
Christoph Stasch[2]

[1] 52° North Initiative for Geospatial Open Source Software GmbH
{broering, ehjuerrens, jirka}@52north.org

[2] Institute for Geoinformatics, University of Muenster
staschc@uni-muenster.de

Abstract

When developing client applications for OGC Web services, it is necessary to implement connectors that are able to interact with the according service interfaces. Since the interaction with OGC Web services is standardized and thus common for multiple client applications, the open source initiative 52°North started in 2006 the development of the OX-Framework – a software framework whose architecture can be used to ease and encapsulate the utilization of OGC Web Services. This framework has gained maturity in the past years and has recently been used as the technological basis for innovative Sensor Web applications in several projects. This work gives an overview of the framework's architecture and subsequently presents examples of open source Sensor Web applications built on top of it.

Introduction

The importance of sensor networks for capturing (real-time) information about phenomena has grown continuously within the last few years. Besides sensor networks which have been in use for very long time periods (e.g. weather station networks) new technologies and especially the miniaturisation of electronics has lead to a widespread use of often embedded sensors. The availability of such a broad base of sensor data offers the possibility to integrate real-time or near real-time measurements with conventional geospatial data (e.g. maps or geometries) for visualisation purposes and for enhancing processing and simulation models.

However, in order to make practical use of sensors it is important to integrate sensor data and metadata as well as functionality into application systems. Especially the heterogeneous character of sensor interfaces makes this a very challenging tasks. In order to facilitate this integration a framework consisting of interface and data formats has been developed by the Open Geospatial Consortium: The Sensor Web Enablement (SWE) framework (Botts et al. 2006).

The SWE framework consists of a set of standards that define data formats for sensor data and metadata and web service interfaces for providing sensor related functionality. Fig. 1 shows an overview of the components forming the SWE architecture.

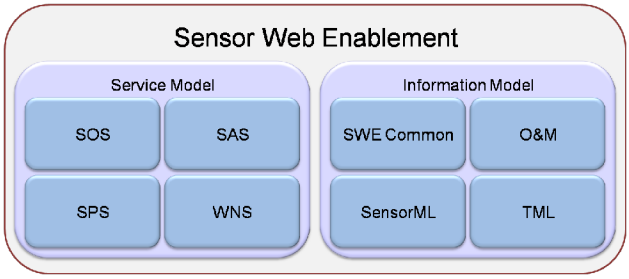

Fig.1: Overview of the OGC SWE framework

As fig. 1 illustrates the SWE framework can be divided into two parts: the service model defining the interfaces of sensor related web service types and the information model comprising those standards which address the specification of data formats. The service model comprises the following four standards: the Sensor Observation Service (SOS) for accessing data measured by sensors (Na&Priest 2007), the Sensor Alert Service (SAS) for dispatching alerts if sensor data fulfil certrain user defined criteria (Simonis 2006), the Sensor Planning Service (SPS) for controlling sensors and setting sensor parameters (Simonis 2007) and finally the Web Notification Service (WNS) which can be seen as a helper service enabling asynchronous communication between SWE components (Simonis&Echterhoff 2006). In the SWE information model the following data formats are defined: SWE Common as a basic data format defining common elements that are re-used by the other SWE standards, Observerations and Measurements (O&M) for encoding data measured by sensors (Cox 2007), the Sensor Model Language (SensorML) for representing the metadata of sensors (Botts 2007) and the Transducer Markup Language (TML) as a hybrid encoding for sensor data and metadata which is especially optimized for streaming data (Havens 2007).

In summary, the SWE framework provides a comprehensive and stable framework that allows building applications that are based on sensors and sensor data. The visualisation of sensor data in conjunction with other geospatial data or results of processing steps is a core element of most sensor web applications. Only by offering well designed and usable client applications a practical benefit for end users is generated.
In this paper, we will address the topic of building client applications that make use of the different SWE standards. In the next section we introduce the 52° North OX-Framework which facilitates the development of SWE based client application by providing a layer that encapsulates all interactions between SWE services and clients.

The OX-Framework

As nowadays different types of geospatial data are available on different sources through OGC Web Services, the interest to integrate them is demanding. This is due to the fact that to carry out reliable decisions different types of data from various sources are required. Thus, a generic framework is needed, which enables integration of these Web Services. Though a number of approaches are available to support users on a rather abstract level (most of the GI-software providers offer so called suites or portal solutions), support for software developers is hard to find.
Tillman & Garnett (2006) support this integrative notion of OGC Web Services, but only by focussing on *client* applications. As the integration of Web Services on *service* side is also demanding especially in the context of sophisticated web processing and service chaining (Kiehle et al. 2006), we propose an integrative approach for *both* environments. Such an extended approach towards service applications would support the development of sophisticated service chains and decrease the complexity of OGC-based software development.
However, until now only specific frameworks are available, either as pure client solutions, such as uDig (http://udig.refractions.net), or as pure service solutions, e.g. deegree (http://deegree.sourceforge.net/). A *generic* solution is still missing. The proposed framework offers developers a customizable and extendable system of cooperating classes supplying a reusable design which is applicable for client and server applications.
Looking from the perspective of the Sensor Web Enablement initiative, different sensors and other support data are required to extract reliable information. This case was the driving force for the SWE Working Group within the 52°North Open Source initiative (http://www.52n.org; Kraak et al. 2005) to come up with an integrated framework named the *OGC Web Service Access Framework* (OX-Framework).

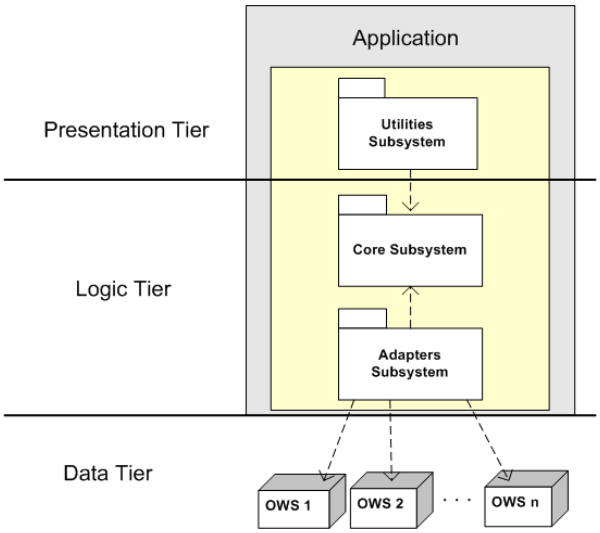

Fig. 2: Three Tier Architecture

The aim of the OX-Framework is to provide an integrative view to access *all* kinds of OGC Web Services and thereafter to visualize and process the queried data. The variety of different services and data encodings makes it necessary to build up a *flexible* architecture. The OX-Framework supports this flexibility by applying three design concepts: a Three Tier Architecture, a Plugin Mechanism as well as a Listener Concept. Those different design aspects are described below.
The Tier Architecture reduces the complexity of the OX-Framework by structuring it into three layers as shown in fig. 2: Adapters Subsystem, Core Subsystem and Utilities Subsystem.
The Adapters Subsystem contains realizations of Service Adpaters for specific OGC Web Services (e.g. SOS). These adapters provide common facilities to the Core for service access in the form of Service Connectors, data visualization engines (Renderers) and feature unmarshalling (Feature Stores).
The Service Connector initializes the *Common Capabilities Model* of the particular service type and is able to trigger its operations. The Renderer converts the received data to a graphical representation. The Feature Store provides unmarshalling facilities for received feature data to the Core's *Feature Model*. All the Service Adapters communicate with each other through the *Core*. This communication is enabled by common data models, which reflect the integrative approach of the OX-Framework.
In detail, the Core incorporates a three-folded data model: The *Common Capabilities Model* implements the OWS Common Specification (Whiteside 2005) and introduces thereby the integrative view on service access to the architecture. The Feature Model provides a basis for accessing, visualizing and processing of vector data based on (Reynolds 2005; Kottman 1999). The *Context Model* enables persistence and exchange functionality for client

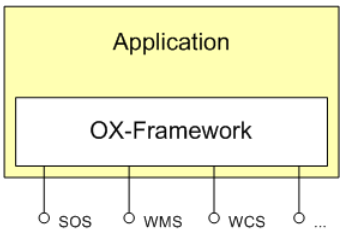

Fig. 3: Plugin Mechanism

projects. It maps a user session – so called "Context" - to an XML-encoding compliant to the Web Map Context Documents specification (Humblet 2003).
The Utilities Subsystem provides functionality for specific UI-frameworks (e.g. Struts or Swing). Those components help the user of the OX-framework to build up client applications or new services using the framework for service-chaining.
The Plugin Mechanism enables the developer to customize and extend the framework with the required Service Adapters in a dynamic way. Hence, it is possible to build up client- and service-oriented applications with the OX-Framework. The idea of this concept is outlined in fig. 3. Implementations of Service Adapters for different OGC service types are plugged into the OX-Framework. The various applications build on top of it can reuse the plugins.
Additionally, the Listener Concept is an important feature of the OX-Framework because it affords a high degree of

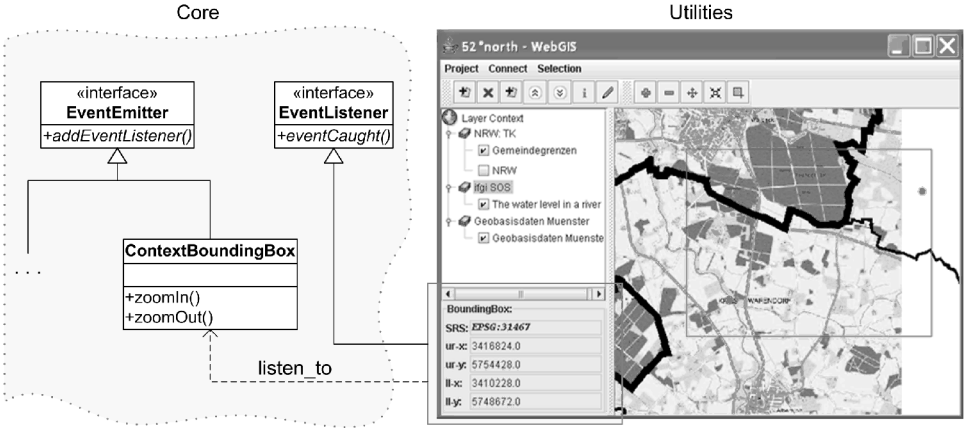

Fig. 4: Example use of the Listener Concept

extensibility and transparency which endows the developer with the absolute control over the framework. Fig. 4 shows exemplarily the functionality of the Listener Concept.
The GUI element displaying the values of a bounding box is part of the Utilities Subsystem and implements the EventListener interface conatined in the Core. It listens to events emitted by the ContextBoundingBox whose instance reflects at runtime the geographical extent of the underlying model. Once an event is received (invoked for example through a window resize), the GUI element automatically updates and adjusts itself.

Applications using the OX-Framework

The described flexible architecture allows the development of various applications upon the OX-Framework. Thus, the different Service Adapters which have already been implemented for specific OGC Web Service types. In the following some of the developed applications built on top of the OX-Framework are described.

Thin SOS Client

This application provides access to sensor data via the user interface of a thin web client runnable in a common Web browser. The client can be used to access and display data gathered by in-situ sensors and served by a Sensor Observaion Service. It allows the user to display time series in form of diagrams or tables and offers additionally the functionality to export the presentation of the data as PDF, Excel or CSV files. Typically, this application is linked and invoked by a web mapping client displaying the position of sensors or features of interest for which observations are hosted by an SOS.
The architecture of this application, as outlined in fig. 5, is divided into two parts: an HTML/AJAX frontend and a server component which is built upon the OX-Framework. The server component acts like a façade to a Sensor Observation Service. This façade server forwards the requests of the HTML frontend to the SOS. The returning SOS responses are parsed and translated by the façade server.
The client, as shown in fig. 6, can be used to display time series from SOS instances compliant to the specification version 1.0.0 serving sensor data in the format defined by Observations & Measurements (also in version 1.0.0). Due

to this fact it is possible to do visual analysis on sensor data hosted by different SOS instances. Sensor observations of different phenomena (e.g. temperature, precipitation and barometric pressure) can be compared. For each phenomenon a separate scale bar is drawn by the client.
The actual time span to be analysed is defined by a start and an end date. This can be done in two ways, entering the dates manually or choosing it from a calendar. To navigate through the data it is possible to use zoom in and zoom out functions. Also, the functionality to browse thruogh time is offered by the client. To store the current view the application is parameterized and a uniquely identifying URL of the current state can be retrieved to share the view with other users. Additionally the data of the current view can be downloaded as a spreadsheet, a CSV formated file, or PDF format.

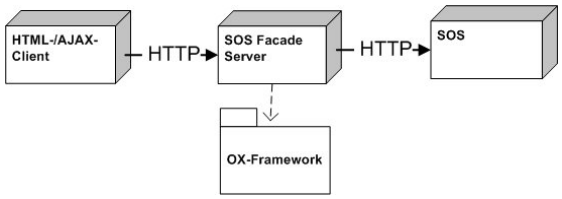

Fig. 5: Thin SOS client and the OX-Framework

Future developments of this project focus on improving the performance, stability, and usability of the application. This includes a major adjustment of the current architectural design as well as the migration to a new rendering library on the server side. This redesign moves the process of sensor data rendering from the AJAX component to the server backend of the system. This new approach is chosen because the amount of sensor data which needs to be processed can be huge (e.g. four time series for each hours of a month: 4 * 24 * 30 = 2880 data points) and took the javascript rendering engine to its limits.

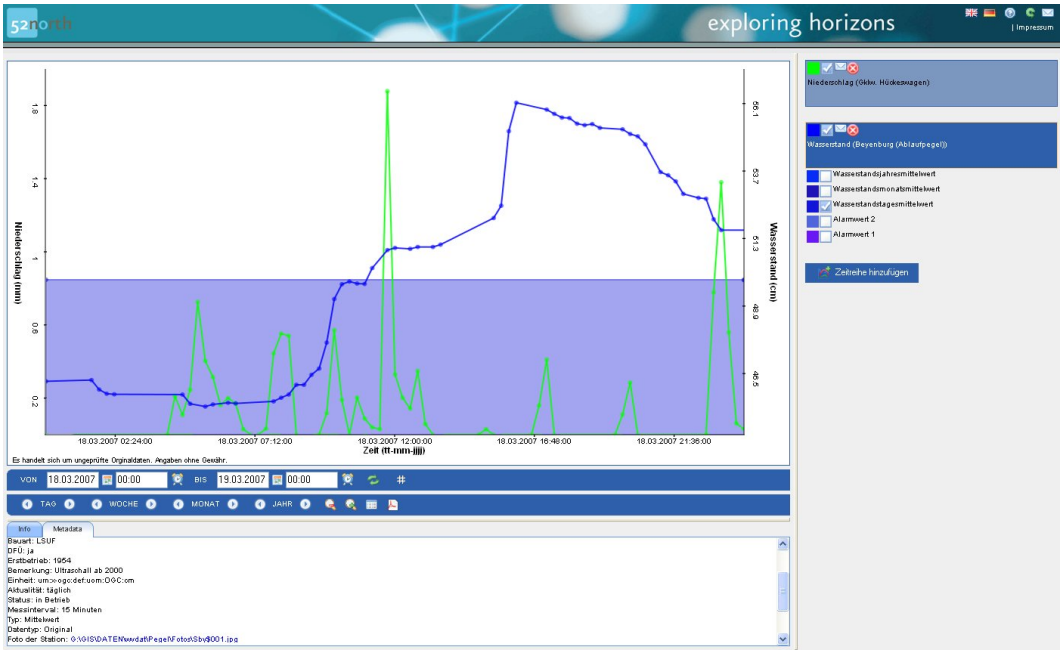

Fig. 6: Thin SOS client

uDig Plugin

uDig is a Desktop GIS written in Java and based on the eclipse platform. It is a free and open source software project driven by Refractions Research (http://udig.refractions.net/). This application adds the ability to the uDig system to access sensor data served by Sensor Observation Services. On the one hand this application allows access of the features of interest observed by the sensors of an SOS. Those features can be displayed in the map view by adding a new layer to uDig. On the other hand it is possible to visualized the observation data served by the SOS in a table view as shown in fig. 7.
In fig. 8 an overview of the architectural design of the application is given. The OX-Framework is used to build the uDig plugin. The functionalities to access the SOS, unmarshal queried data and render visualizations are provided by the framework and reused by the application.
Future steps in the development of this project are the implementation of temporal filter mechanisms. This will enable the users to visualize data for specified time spans.

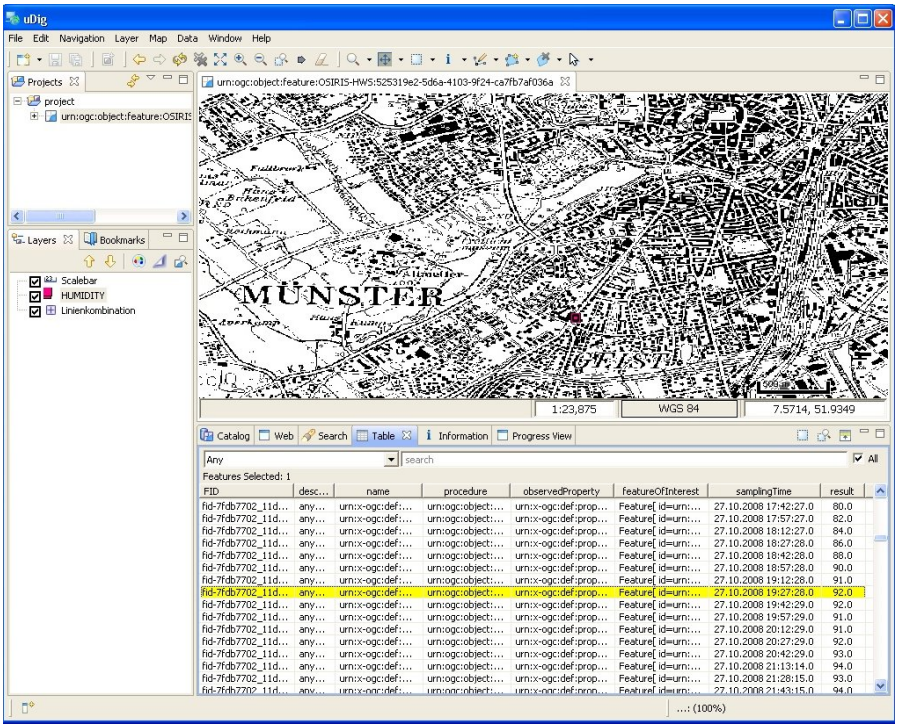

Fig. 7: uDig plugin

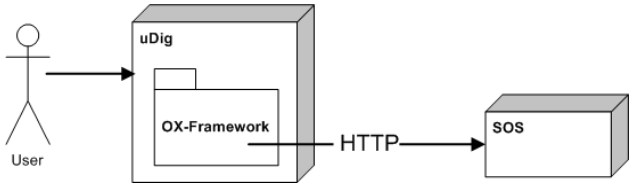

Fig. 8: uDig Plugin and the OX-Framework

Web Map Server

This application conforms to the Web Map Server (WMS) interface specification (De la Beaujadiere 2006). It transforms WMS requests to SOS requests and returns sensor data visualizations to the WMS client. The application realizes an opaque service chaining. The figure below shows the principle of this application.

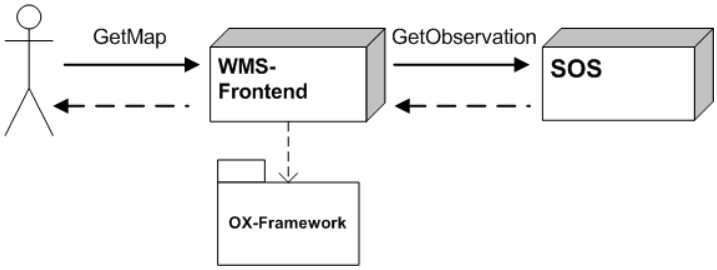

Fig. 9: WMS Frontend

Conclusion

The OX-Framework which was presented in this paper has gained maturity over the past few years and provides a solid foundation for developing OGC-related software. It offers a powerful framework that facilitates the implementation of any kind of application that needs to interact with OGC compliant web services ranging from conventional portrayal or data access services like the Web Mapping Service or the Web Coverage Service to the suite of web service standards belonging to the Sensor Web Enablement suite of standards.

Within various sensor web projects, such as SoKNOS (http://www.soknos.de) or the EU-funded OSIRIS (http://www.osiris-fp6.eu) project, the OX-Framework has been applied in heterogeneous scenarios that pose different requirements to the functionality of client applications. Furthermore, in the last few months several applications in productive environments have been built relying on the OX-Framework. An example is a web based time series client for accessing water level data for all German waterways which was developed for the German Federal Waterways Engineering and Research Institute (http://www.baw.de).

The future development of the OX-Framework will comprise the extension of its functionality by enabling the use of web based processing services coverage handling and the creation of additional Service Adapters. In addition it is expected that further applications using the OX-Framework will be created. For example it is planned to investigate on how further GIS software systems can be SWE- enabled by using the OX-Framework.

In summary the OX-Framework is a powerful framework which significantly facilitates the development of OGC based applications by hiding the complexity of the different OGC standards from the individual developers. As a result the development of customized applications, individually tailored to the need of specific users or use cases becomes possible at much lower costs.

References

Botts, M., G. Percivall, C. Reed & J. Davidson (2007). *OGC Sensor Web Enablement : Overview and High Level Architecture. OGC Whitepaper*. OGC Document Number : 07-165.

Botts, M. & A. Robin (2007). *OpenGIS Sensor Model Language (SensorML) Implementation Specification*. *OpenGIS Implementation Specification*. OGC Document Number : 07-000.

Cox, S. (2007). *Observations and Measurements – Part 1 – Observation schema. OpenGIS Implementation Standard*. OGC Document Number : 07-022r1.

De La Beaujadiere (2006). *OpenGIS Web Map Server Implementation Specification. OpenGIS Implementation Specification*. OGC Document Number : 06-042.

Havens, S. (2007) *OpenGIS Transducer Markup Language (TML) Implementation Specification. OpenGIS Implementation Specification*. OGC Document Number 06-010r6.

Humblet, J.-P. (2003). *Web Map Context Documents. OGC Implementation Specification*. OGC Document Number: 03-036r2.

Kiehle, C., K. Greve & C. Heier (2006). *Standardized Geoprocessing - Taking Spatial Data Infrastructures one Step Further*. Proceedings AGILE 2006. 9th AGILE International Conference on Geographic Information Science: 273–282. Visegrad, Hungary.

Kottman, C. (1999). *The OpenGIS™ Abstract Specification Topic 5: Features (Version 4). OGC Abstract Specification*. OGC Document Number: 99-105r2

Kraak, M.-J., A. Sliwinski & A. Wytzisk (2005). *What happens at 52N? An Open source approach to education and research*. Joint ICA commission seminar 6.-8. July 2005, part of the 22nd ICA conference, ICC 2005: Mapping approaches into a changing world, 2005, 16-20.

Na, A. & M. Priest (2007). *Sensor Observation Service – Implementation Specification Version* 1.0.0. OGC Implementation Specification. OGC Document Number: 06-009r6.

Reynolds, G. (2005). *GO-1 Application Objects*. *OGC Implementation Specification*. OGC Document Number: 03-064r10.

Simonis, I. (2006). *OGC Sensor Alert Service Candidate Implementation Specification. OpenGIS Best Practices*. OGC Document Number: 06-028r3.

Simonis, I. (2007). *OpenGIS Sensor Planning Service Implementation Specification. OpenGIS Implementation Specification*. OGC Document Number: 07-014r3.

Simonis, I. & J. Echterhoff (2006). *Draft OpenGIS Web Notification Service Implementation Specification. OpenGIS Best Practices Paper*. OGC Document Number: 06-095.

Sliwinski, A., I. Simonis et al. (2005). *Boosting the OGC Sensor Web Enablement Initiative by Open Source Web Services – The Case of 52°North*. AGIT 2005, July 6 - 8, 2005.

Tillman, S. & J. Garnett. (2006). *OWS Integrated Client Architecture, Design, and Experience. OGC Discussion Paper*. OGC Document Number: 05-116.

Whiteside, A. (2005). *OGC Web Services Common Specification. OGC Implementation Specification*. OGC Document Number: 05-008.