УДК 52(15).003

PRACTICAL ISSUES OF SENSOR WEB IMPLEMENTATION AND GRIDIFICATION

N. Kussul, M. Korbakov, O. Kravchenko

Space Research Institute of NASU-NSAU,
03680, Kyiv, prosp. Glushkova, 40.
Тел.: +380(44)526 2553,inform@ikd.kiev.ua

In this paper we provide an overview of emerging Sensor Web paradigm and show several practical issues of using Sensor Web technologies for real-world tasks. Issues under study include sensor description using SensorML and database performance for serving observations data. This paper also shows an approach for integrating standard Sensor Observation Service with Globus Toolkit Grid platform.

В данной работе представлен обзор развивающейся парадигмы Sensor Web и рассмотрены практические вопросы использования данной технологии для решения прикладных задач. Рассматриваются вопросы описания численных моделей с использованием языка SensorML и оценки производительности баз данных в задачах обслуживания сервисов Sensor Web. Кроме того, в работе описаны подходы к интеграции сервисов Sensor Web с Grid-платформой Globus Toolkit.

The concept of Sensor Web

Sensor Web is an emerging paradigm and technology stack for integration of heterogeneous sensors into common informational infrastructure. The basic functionality required from such infrastructure is remote data access with filtering capabilities, sensors discovery and triggering of events by sensors conditions.

Sensor Web is governed by the set of standards developed by Open Geospatial Consortium [1]. At present, the following standards are available and approved by consortium:

1. OGC Observations & Measurements [2] – Common terms and definition for Sensor Web domain;
2. Sensor Model Language [3] – XML-based language for describing different kinds of sensors;
3. Transducer Model Language [4] – XML-based language for describing the response characteristics of a transducer;
4. Sensor Observations Service [5] – an interface for providing remote access to sensors data;
5. Sensor Planning Service [6] – an interface for submitting tasks to sensors.

There are also standards drafts that are available from Sensor Web working group but not yet approved as official OpenGIS standards:

1. Sensor Alert Service – service for triggering different kinds of events basing of sensors data;
2. Web Notification Services – notification framework for sensor events.

Sensor Web paradigm assumes that sensors could belong to different organizations with different access policies or, in broader sense, to different administrative domains. However existing standards stack doesn't provide any means for enforcing data access policies leaving it to underlying technologies. One possible way for handling informational security issues in Sensor Web is presented in this paper.

Use case

One of the most challenging problems for Sensor Web technology implementation is global ecological monitoring in the framework GEOSS (Global Earth Observation System of Systems) [7]. In this paper we consider the problem of flood monitoring using satellite remote sensing data, in-situ data and results of simulations.

The problem of floods monitoring by itself consumes data from many heterogeneous data sources such as remote sensing satellites (we are using data of ASAR, MODIS and MERIS sensors), in-situ observations (water levels, temperature, humidity, etc). Floods prediction is adding the complexity of physical simulation to the task. All of these results into the complex dataflow shown on Fig. 1.

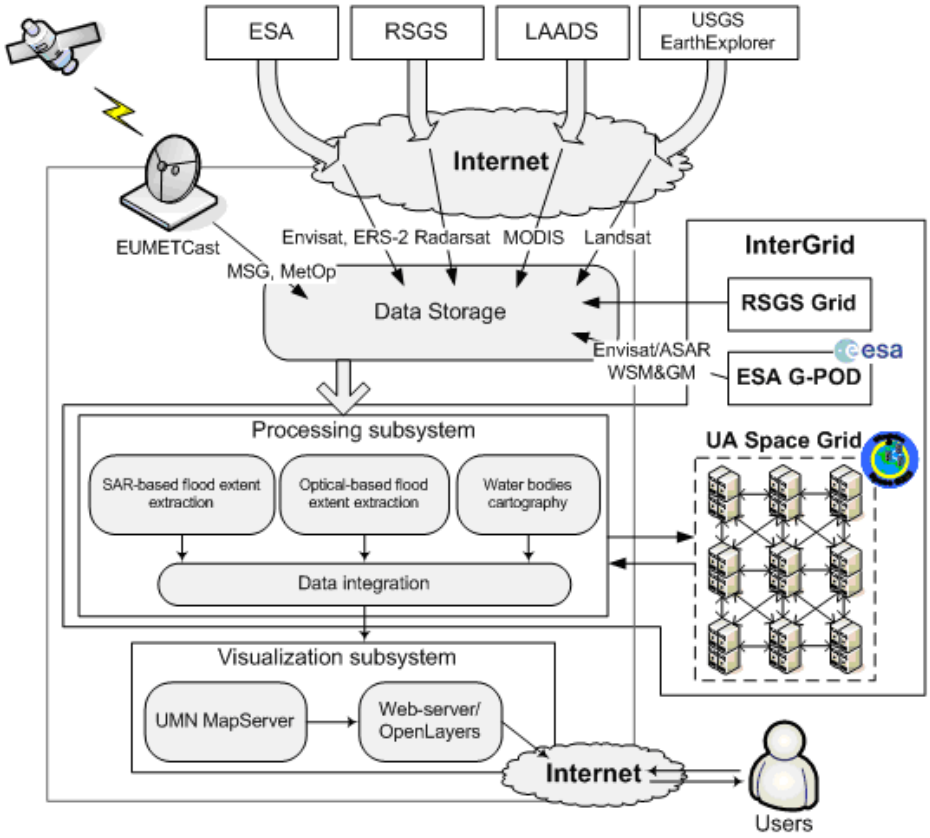

Fig. 1. Data flow perspective of flooding test case

To predict flooding parameters such as rivers stage/discharge and extents of flooded areas we need to use cascade of simulation models: regional numerical weather prediction (NWP) model, hydrological model and hydraulic model (see Fig. 2).

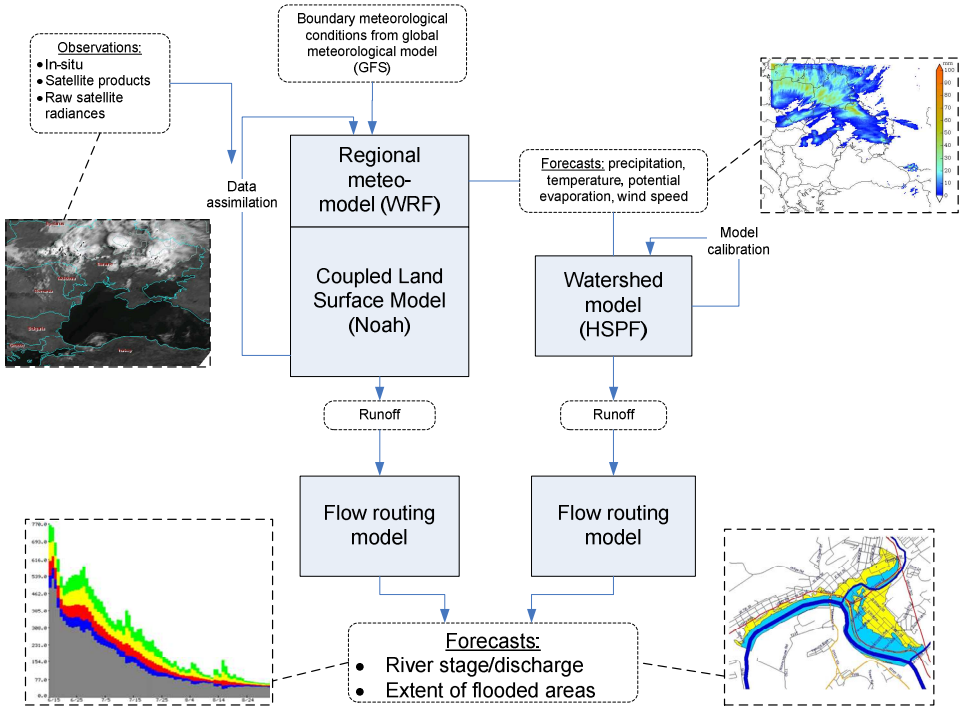

Fig. 2. Simulation cascade to predict flood events

To obtain quantitative estimates of precipitation and other meteorological parameters we use regional NWP model WRF (Weather Research&Forecasting). WRF model is a joint development of a number of USA agencies and universities (http://wrf-model.org). This model was configured and adapted to the territory of Ukraine to run with spatial resolution of 10 km. Currently we routinely produce 72-hours weather forecasts every 6 hours [8]. To drive regional model the additional weather forecasts from global NWP model are used. These data are required to specify external meteorological forcing as boundary conditions for regional weather model. Currently we use forecast frames produced by GFS (Global Forecast System) model operated by NCEP.

The Sensor Web perspective of this test case is depicted on Fig. 3. It shows collaboration of different OpenGIS specifications of Sensor Web. The data from different sources (numerical models, remote sensing, in-situ observations) is accessed through Sensor Observation Service (SOS). Aggregator site is running Sensor Alert Service to notify interested organization of possible flood event using different communication mean. Aggregator site is also sending orders to satellite receiving facility using Sensor Planning Service (SPS) to get satellite imagery only available by preliminary order.

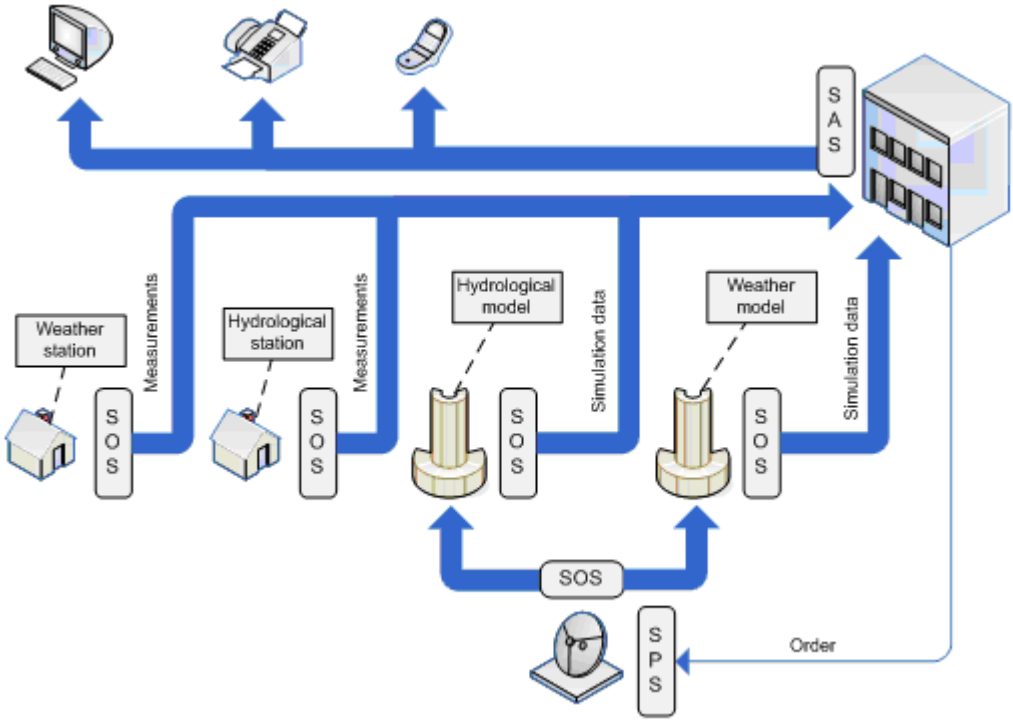

Fig. 3. Sensor Web perspective of flooding test case

SensorML description of WRF weather model

Sensor Modeling Language (SensorML) is the cornerstone of all Sensor Web services. It provides comprehensive description of sensor parameters and capabilities as well as sensor calibration lineage, measure errors characteristics, response curves and other information about sensor. SensorML can be used for describing different kind of sensors:

- Stationary or dynamic;
- Remote or in-situ;
- Physical measurements or simulations.

Modeling and simulation are very important parts of environmental monitoring. The importance of different models in the process of solving of real-world tasks was demonstrated in the previous part of this paper. Sensor Web infrastructure should be able to integrate modelling data and provide remote data access for the as well as other Sensor Web features like discovery, sending orders, etc.

At the bare minimum, SensorML description should contain general information about sensor (time and geographical extents, contact persons, etc) and lists of inputs and outputs. SensorML input could be either physical phenomena or some external measured value. The first case applies to physical measuring devices and second – to models and simulations.

We have tried to describe weather modelling process using WRF numerical model in terms of SensorML. The following listing shows one input of this model.

```
<sml:input name="QVAPOR">
 <swe:DataArray definition="urn:ogc:def:phenomenon:time">
  <swe:elementCount>
   <swe:Count definition="urn:ogc:def:property:OGC:numberOfPixels">
    <swe:value>1</swe:value>
   </swe:Count>
  </swe:elementCount>
  <swe:elementType name="">
   <swe:DataArray definition="urn:ogc:def:phenomenon:altitude">
    <swe:elementCount>
     <swe:Count definition="urn:ogc:def:property:OGC:numberOfPixels">
      <swe:value>30</swe:value>
     </swe:Count>
    </swe:elementCount>
    <swe:elementType name="">
     <swe:DataArray definition="urn:ogc:def:phenomenon:latitude">
      <swe:elementCount>
       <swe:Count definition="urn:ogc:def:property:OGC:numberOfPixels">
        <swe:value>202</swe:value>
       </swe:Count>
      </swe:elementCount>
      <swe:elementType name="">
       <swe:DataArray definition="urn:ogc:def:phenomenon:longtitude">
        <swe:elementCount>
         <swe:Count definition="urn:ogc:def:property:OGC:numberOfPixels">
          <swe:value>219</swe:value>
         </swe:Count>
        </swe:elementCount>
        <swe:elementType name="">
         <swe:Quantity definition="urn:ogc:def:phenomenon:QVAPOR">
          <swe:uom code="kg_kg-1"/>
         </swe:Quantity>
        </swe:elementType>
       </swe:DataArray>
      </swe:elementType>
     </swe:DataArray>
    </swe:elementType>
   </swe:DataArray>
  </swe:elementType>
 </swe:DataArray>
</sml:input>
```

There are nearly 50 inputs and 20 outputs for basic WRF configuration. It's obvious that information density of inputs and outputs descriptions in SensorML is quite low and each of them requires quite significant amount of XML code to be properly described. The problem lies in very verbose description of multidimensional data. Three- and four-dimensional data arrays are very common in environmental modeling but SensorML provides poor experience regarding them.

Authors have raised this problem during thematic meeting and hope that next revision of SensorML will include some elements for simpler description of multidimensional data.

Sensor Observation Service implementation

In order to provide access to hydrometeorological observations over the regions of interest we have deployed Sensor Observation Service implementation on the site of Space Research Institute of NASU-NSAU. We have studied two possible implementations of SOS for particular task of serving temperature sensors data. Implementations under study were:

- UMN Mapserver v5 (http://mapserver.gis.umn.edu/)
- 52North SOS (http://52north.org/)

The advantages and disadvantages of these solutions can be summarized in the following table.

	UMN Mapserver v5	52North SOS
Advantages	1. Very good and reliable abstraction for different data sources (raster files, spatial databases, WFS, etc) 2. Simple application model (CGI executable) 3. Wide set of features beside SOS 4. Open software	1. SOS implementation is stable and complete 2. Platform-independent (Java-based) 3. A part of wider Sensor Web implementations stack (SPS, SAS) 4. Open software 5. Source code is clean and easily reusable
Disadvantages	1. SOS support is declared but far from being working implementation 2. Poor documentation on SOS topic 3. Strange plans for future development (in particular, automatic SensorML generation)	1. No data abstraction: the only data source is relational database of specific structure 2. Database structure is far from optimal (strings as primary keys, missed indexes, etc) 3. Complex application model (Java web application)

The best experience received was with 52North SOS server. Its main disadvantage is complex relational database scheme. However it was possible to adapt existing database structure to the one, required by 52North using a number of SQL views and synthetic tables. The details of database adaptation are given in the next section.

We have used 52North implementation for building a testbed SOS server providing data of temperature sensors over Ukraine and South Africa regions. The server is available by URL http://web.ikd.kiev.ua:8080/52nsos/sos.

SOS output comes as XML document in special scheme, specified by SOS reference document. The standard is describing two possible forms of results, namely “Measurement” and “Observation”. The first form is more suitable to the situations when the service is returning small amounts of heterogeneous data. The second form is most suitable for long time series of homogeneous data. The table below provides an example of SOS output in these two forms and clearly shows the difference.

Measurement

```
<om:Measurement gml:id="o255136">
 <om:samplingTime>
  <TimeInstant
xsi:type="gml:TimeInstantType">
   <timePosition>
    2005-04-14T04:00:00+04
   </timePosition>
  </TimeInstant>
 </om:samplingTime>
 <om:procedure xlink:href=

"urn:ogc:object:feature:Sensor:WMO:33506"/>
 <om:observedProperty xlink:href=

"urn:ogc:def:phenomenon:OGC:temperature"/>
 <om:featureOfInterest>
  <sa:Station gml:id="33506">
   <name>WMO33506</name>
   <sa:sampledFeature xlink:href=""/>
   <sa:position>
    <Point>
     <pos srsName="urn:crs:epsg:4326">
      34.55 49.6
     </pos>
    </Point>
   </sa:position>
  </sa:Station>
 </om:featureOfInterest>
 <om:result uom="celsius">10.9</om:result>
</om:Measurement>
```

Observation

```
<om:result>
2005-03-14T21:00:00+03,33506,-
5@@
2005-03-15T00:00:00+03,33506,-
5.2@@
2005-03-15T03:00:00+03,33506,-
5.5@@
2005-03-15T06:00:00+03,33506,-
4.6@@
2005-03-15T09:00:00+03,33506,-
2.2@@
2005-03-
15T12:00:00+03,33506,1.7@@
2005-03-
15T15:00:00+03,33506,1.7@@
2005-03-
15T18:00:00+03,33506,2.4@@
2005-03-15T21:00:00+03,33506,-
0.7@@
2005-03-16T00:00:00+03,33506,-
1.4@@
2005-03-16T03:00:00+03,33506,-
1.1@@
2005-03-16T06:00:00+03,33506,-
1.1@@
2005-03-16T09:00:00+03,33506,-
1.3@@
2005-03-
16T12:00:00+03,33506,0.5@@
2005-03-
16T15:00:00+03,33506,1.7@@
2005-03-
16T18:00:00+03,33506,1.5@@
</om:result>
```

Database issues

The database of hydrometerological information of Space Research Institute of NASU-NSAU contains nearly 1.5 millions of records with observations started at year 2005 to the present moment. The data is stored in PostgreSQL database with PostGIS spatial extensions. Most of the data records are contained in single table ‘observations’ with indexes built over fields with observation time and station identifier. Tables of such volume requires some special handling so the index for time field was clusterized thus reordering data on the disks and reducing the need for I/O operations. Clusterization of time index reduced typical queries times from 8000 ms to 250 ms.

To adapt this database to the requirements of 52North we have created a number of auxiliary tables with reference values related to SOS (such as phenomena names, sensor names, regions parameters, etc) and a set of views that transforms underlying database structure into 52North scheme. The most important view that binds all values of synthetic tables together with observations data have the following definition:

```
SELECT observations."time" AS time_stamp, "procedure".procedure_id,
feature_of_interest.feature_of_interest_id, phenomenon.phenomenon_id,
offering.offering_id, '' AS text_value, observations.t AS numeric_value, '' AS mime_type,
observations.oid AS observation_id

  FROM observations, "procedure", proc_foi, feature_of_interest, proc_off,
offering_strings offering, foi_off, phenomenon, proc_phen, phen_off

 WHERE "procedure".procedure_id::text = proc_foi.procedure_id::text AND
proc_foi.feature_of_interest_id::text = feature_of_interest.feature_of_interest_id AND
"procedure".procedure_id::text = proc_off.procedure_id::text AND
proc_off.offering_id::text = offering.offering_id::text AND foi_off.offering_id::text =
offering.offering_id::text AND foi_off.feature_of_interest_id::text =
feature_of_interest.feature_of_interest_id AND proc_phen.procedure_id::text =
"procedure".procedure_id::text AND proc_phen.phenomenon_id::text =
phenomenon.phenomenon_id::text AND phen_off.phenomenon_id::text =
phenomenon.phenomenon_id::text AND phen_off.offering_id::text =
offering.offering_id::text AND observations.wmoid::text =
feature_of_interest.feature_of_interest_id;
```

52North’s database scheme uses string primary keys for auxiliary tables instead of synthetic numerical and is far from optimal in sense of performance. It doesn’t have strong impact on performance with record counts in these tables less than one hundred but will surely cause problems in large-scale SOS-enabled data warehouses.

The typical SQL query from 52North service is quite complex (see listing below). An average response time for such query (assuming one month time period) is about 250 ms with PostgreSQL running in virtual environment on 4 CPUs server with 8GB of RAM and 5 SCSI 10k rpm disks in RAID5 array. Increasing of query depth results in linear increasing of response time with estimate speed of 50 ms per month (see Fig. 4).

```
SELECT observation.time_stamp, observation.text_value, observation.observation_id,
observation.numeric_value, observation.mime_type, observation.offering_id,
phenomenon.phenomenon_id, phenomenon.phenomenon_description,
phenomenon.unit,phenomenon.valuetype,observation.procedure_id,
feature_of_interest.feature_of_interest_name, feature_of_interest.feature_of_interest_id,
feature_of_interest.feature_type, SRID(feature_of_interest.geom),
AsText(feature_of_interest.geom) AS geom FROM phenomenon NATURAL INNER JOIN observation
NATURAL INNER JOIN feature_of_interest WHERE (feature_of_interest.feature_of_interest_id
= '33506') AND (observation.phenomenon_id =
'urn:ogc:def:phenomenon:OGC:1.0.30:temperature') AND (observation.procedure_id =
'urn:ogc:object:feature:Sensor:WMO:33506') AND (observation.time_stamp >= '2006-01-01
02:00:00+0300'AND observation.time_stamp <= '2006-02-26 01:00:00+0300')
```

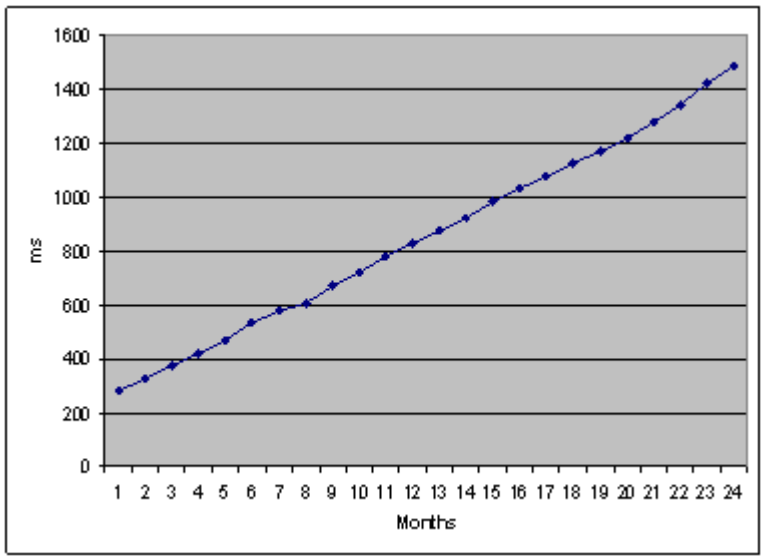

Fig. 4. Dependency between depth of query and response time

SOS Gridification

Sensor Web services like SOS, SPS and SAS can benefit from integration with Grid platform like Globus Toolkit [9]. Many Sensor Web features can take advantage of Grid platform services, in particular:

- Sensors discovery could be performed through combination of Index Service and Trigger Service;
- High-level access to XML description of sensors and services could be made through queries to Index Service;
- Grid platform provides convenient way for implementation of notifications and event triggering using corresponding platform components [10];
- Reliable File Transfer service [11] provides reliable data transfer for large datasets;
- Globus Security Infrastructure [12] provides enforcement of data and services access policies in a very flexible way allowing implementation of desired security policy.

Authors have developed a testbed SOS Service using Globus Toolkit as a platform. For now, this service works as proxy translating and redirecting user request to usual HTTP SOS server (see Fig. 5). The current version uses client-side libraries for interacting with SOS provided by 52North in their OX-Framework. Next version will include in-service implementation of SOS-server functionality.

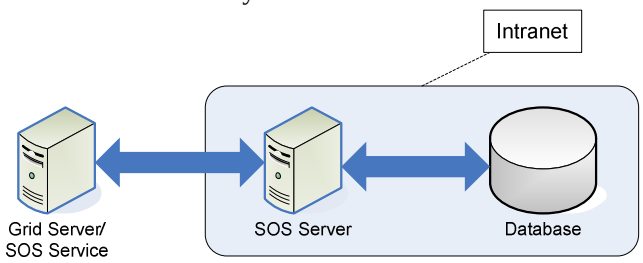

Fig. 5. Grid-based SOS service implementation

Grid service implementing SOS provides the interface specified in SOS reference document. The key difference between interfaces of standard and Grid-based implementations of SOS lies in encoding of service requests. The standard implementation uses custom serialization for requests and responses and Grid-based implementation uses standard SOAP encoding.

To get advantage of the most Globus features SOS service should export service capabilities and sensor descriptions as WSRF resource properties [13]. Traditional way of implementation of such properties requires translation between XML Schema and Java code. However the XML Schema of SOS and related standards (in particular GML [14]) is very complex and there are no available program tools able to generate Java classes from it. We have solved this problem by storing service capabilities and sensor descriptions data as DOM Element objects and using custom serialization for this class provided by Axis framework that is used by Globus Toolkit. Using this approach we can't access particular elements of XML document in object-oriented styled. However SOS Grid service is acting as proxy between user and SOS implementation so it doesn't need to modify XML directly. The following Java listing shows this approach in code.

```
public void initialize() {
  this.propSet = new SimpleResourcePropertySet(RESOURCE_PROPERTIES);
  try {
    serviceCapabilitiesRP = new SimpleResourceProperty(RP_SERVICECAPABILITIES);
    this.serviceCapabilitiesRP.add(new Object());
    this.propSet.add(serviceCapabilitiesRP);
  } catch (Exception e) {
    throw new RuntimeException(e.getMessage());
  }
  try {
    InputStream istream = new ByteArrayInputStream(SOSMethods.getCapabilities());
    DOMParser parser = new DOMParser();
    parser.parse(new InputSource(istream));
    this.serviceCapabilitiesRP.set(0, parser.getDocument().getDocumentElement());
  } catch (Exception e) {
    throw new RuntimeException(e.getMessage(), e);
  }
}
```

With resource properties defined in this way we user can access them using standard Globus API or command-line utilities:

```
wsrf-get-property -s https://gt.ikd.kiev.ua:8443/wsrf/services/SOSService
"{http://www.opengis.net/sos/0.0}Capabilities"
```

Conclusions

Despite of immaturity of Sensor Web technology stack it can provide good experience in serving heterogeneous data of in-situ observations. SOS implementation for serving geospatial raster data that is important for remote sensing data are yet to be implemented.

SensorML descriptions of complex environmental models are too verbose. To allow wide use of models in Sensor Web environment some changes should be made in SensorML to shorten descriptions of multidimensional inputs and outputs.

Integration with Globus Toolkit Grid platform allows Sensor Web service to take advantage of robust information management features of Grids as well as mature mechanisms for data access policy enforcement.

Acknowledgements

This work is supported by ESA CAT-1 project “Wide Area Grid Testbed for Flood Monitoring using Spaceborne SAR and Optical Data” (#4181) and by INTAS-CNES-NSAU project “Data Fusion Grid Infrastructure” (Ref. Nr 06-1000024-9154).

1. *Mike* Botts, *George Percivall, Carl Reed, John Davidson.* OGC Sensor Web Enablement: Overview and High Level Architecture (OGC 07-165). http://portal.opengeospatial.org/files/?artifact_id=25562.
2. *OpenGIS* Observations and Measurements. http://www.opengeospatial.org/standards/o%2526m.
3. *OpenGIS* Sensor Model Language (SensorML). http://www.opengeospatial.org/standards/sensorml.
4. *OpenGIS* Transducer Markup Language. http://www.opengeospatial.org/standards/tml.
5. *OpenGIS* Sensor Observation Service. http://www.opengeospatial.org/standards/sos.
6. *OpenGIS* Sensor Planning Service Implementation Specification. http://www.opengeospatial.org/standards/sps.
7. *Global* Earth Observation System of Systems (GEOSS), 10-Year Implementation Plan Reference Document // ESA Publication Division, Netherlands, 2005. — 209 p.
8. *Shelestov A., Kravchenko O., Ilin M.* Geospatial data visualisation in Grid system on Ukrainian segment of GEOSS/GMES // Proc. of the V-th International Conference “Information Research&Applications”. — Varna (Bulgaria). — June 26-30, 2007. – Vol. 2. – P. 422–428.
9. *Foster I.*. Globus Toolkit Version 4: Software for Service-Oriented Systems // IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, 2005, pp. 2-13.
10. *Humphrey M., Wasson G., Jackson K., Boverhof J., Rodriguez M., Bester Joe, Gawor J., Lang S., Foster I., Meder S., Pickles S., and McKeown M.*. State and Events for Web Services: A Comparison of Five WS-Resource Framework and WS-Notification Implementations // 4th IEEE International Symposium on High Performance Distributed Computing (HPDC-14), Research Triangle Park, NC, 24-27 July 2005.
11. *Allcock W., Bresnahan J., Kettimuthu R., Link M., Dumitrescu C., Raicu I., Foster I.* The Globus Striped GridFTP Framework and Server // Proceedings of Super Computing 2005 (SC05), November 2005.
12. *Welch V., Siebenlist F., Foster I., Bresnahan J., Czajkowski K., Gawor J., Kesselman C., Meder S., Pearlman L., Tuecke S.* Security for Grid Services // Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), IEEE Press, 2003.
13. *Foster I. (ed), Frey J. (ed), Graham S. (ed), Tuecke S. (ed), Czajkowski K., Ferguson D., Leymann F., Nally M., Sedukhin I., Snelling D., Storey T., Vambenepe W., Weerawarana S.* Modeling Stateful Resources with Web Services // http://www-106.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf.
14. *OpenGIS* Geography Markup Language (GML) Encoding Standard. http://www.opengeospatial.org/standards/gml.