20th International Congress on Modelling and Simulation, Adelaide, Australia, 1–6 December 2013
www.mssanz.org.au/modsim2013

Describing models on the web using OGC standards

Kym Watson [a], Hylke van der Schaaf [a]

[a] *Fraunhofer IOSB, Fraunhoferstraße 1, 76131, Karlsruhe, Germany*
Email: kym.watson@iosb.fraunhofer.de

Abstract: Environmental decision support systems normally require a data processing workflow based on models to explore alternatives. The typical workflow to handle environmental modelling involves several steps covering data discovery, access, pre-processing, model execution and validation, concluded by result visualization. These time consuming steps are usually setup for a particular scenario and set of input data. Scientists normally create their models in specific languages such as R or MATLAB. A challenge is understanding the data model of the scientists and getting the data into the model. In general, scientists are also not able to make their models available as web services. To make scientific models that fuse sensor data fit better into a service-oriented architecture, a software framework called *Fusion4Decision* was developed. The software framework provides a standard interface to processing algorithms, the so-called *Fusors*. The term Fusor refers to a general fusion or processing of input data, including through a model based computation. A common fusor is the spatiotemporal interpolation of measurement data. The Fusor is written in any software code that can be integrated into a Java environment, such as MATLAB, R, Python, and C variants. This framework based on Open Geospatial Consortium (OGC) standards can make scientific models available as a web service with standardized interfaces.

The OGC services used in Fusion4Decision are: (a) *Sensor Observation Service (SOS)* to access sensor observations with queries filtering on the phenomenon (property) and the spatial and temporal domains of the observations, and (b) *Sensor Planning Service (SPS)* to parameterize and task (schedule and execute) assets such as sensors, sensor platforms (e.g. satellites), models or even persons (e.g. to conduct ex-situ measurements). The OGC information models *Observation & Measurement Model (O&M)* and *Sensor Model Language (SensorML)* also play a fundamental role.

The main operations of the SPS are DescribeTasking (to get the tasking parameters), GetFeasibility (to ascertain if the asset can be tasked with the given parameters) and Submit (to actually execute the task). During the execution the operations GetStatus and Cancel are available. In Fusion4Decision we apply the SPS to models and the model result(s) become new observations for a SOS, i.e. *the model is considered to be sensor* and its meta-data is described in OGC SensorML. The SPS operations are functionally richer than those of the Web Processing Service (WPS) that is also often used to wrap processing modules as a web service.

The formal description of the input and output arguments of the models in a language suitable both for scientists and client software is essential. The model description is encoded as a JSON object and consists of fields for the model name, a human readable descriptive text as well as formal descriptions of the inputs and outputs. The inputs and outputs allow for arrays of the basic variable types scalar, string, time, URL and file. Their description includes a) units of scalars, b) default, minimum and maximum values of scalars and optionally c) an annotation as a URI linking to an authoritative definition in an ontology. This covers the requirements of typical scientific models and also encourages the inclusion of comprehensive meta-data needed to convey full understanding of the model algorithm and its limitations. The JSON description of a model can be automatically translated into SensorML for use by the OGC services SOS and SPS.

The increasing proliferation of sources of geospatial data on the web as well as models to process the data and derive new information underlines the need for a standardised framework to better link data, models and their results. Standards of the Open Geospatial Consortium can be used to integrate data access and models into web services, thus being a step towards the Model Web in which scientists and decision makers can work together effectively. The paper proposes a simple way of describing the input and output arguments of a model using JSON. This JSON description can be readily understood and generated by model providers and also translated into the sensor description language SensorML. The latter is the basis for applying the sensor concept in the OGC standards SOS and SPS to models ("model as a sensor"). This approach bridges the gap between scientists and IT specialists.

Keywords: *Modelling, open geospatial standards, JSON, SensorML*

1 Introduction

Environmental decision support systems normally require a data processing workflow based on models to explore alternatives. The typical workflow to handle environmental modelling is as follows:

1. discover and select sources of (e.g. sensor) data
2. access and pre-process the source data
3. select and parametrize a model based algorithm
4. execute the model using additional parameters, e.g. to control convergence
5. validate and calibrate the model
6. make the results available for downstream processing and visualization as maps, diagrams etc.

These steps are often tedious due to the heterogeneity of the data types and formats. They are usually setup for a particular scenario with the aid of scripts written in languages such as R or MATLAB, while the model itself is a software module with specific input and output data sets. A challenge is understanding the data model of the scientists and getting the data into the model. This approach is difficult to sustain when new data sources or models are to be incorporated.

There is a wide range of different classes of models as clearly explained in Jakeman et al. (2006). Some models may require an extensive set of parameters that in part need to be calibrated with many model runs. Models may not necessarily have a deterministic or even guaranteed bounded execution time. The model outputs can be numerical arrays, maps or diagrams. In this paper a model can be thought of as a procedure or algorithm that takes input data of a defined structure and produces one or more outputs. In the environmental domain, both input and output are normally geospatial data structures describing environmental observations in a spatiotemporal context.

The Open Geospatial Consortium (OGC) is a non-profit, international voluntary consensus standards organization that is leading the development of standards for geospatial and location based services. The standards are applied to a wide range of domains in geosciences. The primary objective of the standards is to support information discovery, access, processing, fusion and decision-making. An introductory overview of OGC standards and their application in the geosciences is given in the on-line tutorial Percivall (2010).

The hydrology domain has traditionally been faced with major modelling challenges and has been an incubator for modelling techniques. This domain has given considerable thought to the standardization of data access, and map and modelling services using the standards of the Open Geospatial Consortium (OGC) and the World Meteorological Organisation (WMO). There is a joint OGC-WMO Hydrology Domain Working Group and an OGC Standards Working Group (SWG) for WaterML 2.0 development (to encode hydrological observations), cf. `http://external.opengis.org/twiki_public/HydrologyDWG/`. The Architecture Implementation Pilot (AIP) Phase 6 is conducting work in the GEO Water Societal Benefit Area with this specific scope. This is in support of the Integrated Global Water Cycle Observations activity in GEO.

Over the last 10-15 years several frameworks have been developed to realize modelling workflows. Some frameworks focus on linking modules in a tightly coupled software environment whereas others aim for a loosely coupled system of models and processing components. Rizzoli et al. (2008) gives an overview of conceptual types of modelling frameworks. Jagers (2010) describes 8 different frameworks for linking models and model components. Argent (2005) presents a software environment for catchment and water quality modelling. The creation of models from software components with a focus on hydrological processes is the subject of the Jena Adaptable Modelling System (JAMS) described in Kralisch and Fischer (2012). Domain specific languages for setting up and executing models have been proposed for example in David et al. (2012). The fundamental issue of defining the semantics of the data with the aid of ontologies is addressed by Rizzoli et al. (2008).

OGC standards such as those in the Sensor Web Enablement (SWE) suite, WPS, WFS, WCS and the Observation and Measurement Model (O&M ISO 19156) to describe observations and SensorML to describe sensors are opening up new opportunities to create modelling frameworks in a service oriented web, see e.g. Watson and Watson (2012). These standards along with sensor technology developments are making it easier to deploy new sensor networks, both professionally and by citizens. Coupled with the increased availability of environmental data in general through initiatives such as GEOSS, this implies a wealth of environmental information to be harnessed by decision makers. Moreover, the very same standards can be used by scientists to publish their results as new derived environmental observations of higher quality. The objective is not only to facilitate data access and data publishing, but also to establish the so-called Model Web. The vision of the Model Web is to have an integrated, dynamic system of databases and models able to answer any questions of decision makers. This represents a real paradigm shift with new perspectives but posing challenges as well.

2 The Fusion4Decision Framework

A software framework called Fusion4Decision was developed to make scientific models that fuse sensor data fit better in a service-oriented architecture in the spirit of the Model Web. This framework based on the Open Geospatial Consortium standards can make scientific models available as a web service with standardized web interfaces. The OGC services used in Fusion4Decision are: (a) *Sensor Observation Service (SOS)* to access sensor observations with queries filtering on the phenomenon (property) and the spatial and temporal domains of the observations, and (b) *Sensor Planning Service (SPS)* to parameterize and task (schedule and execute) assets such as sensors, sensor platforms (e.g. satellites), models or even persons (e.g. to conduct ex-situ measurements). The main operations of the SPS are DescribeTasking (to get the tasking parameters), GetFeasibility (to ascertain if the asset can be tasked with the given parameters) and Submit (to actually execute the task). During the execution the operations GetStatus and Cancel are available. In Fusion4Decision we apply the SPS to models and the model result(s) become new observations for a SOS, i.e. *the model is considered to be sensor*. The SPS operations are functionally richer than those of the Web Processing Service (WPS) that is also often used to wrap processing modules as a web service.

The services are accompanied by two fundamental information models:

- Sensor Model Language (SensorML). It describes the static metadata of a sensor such as manufacturer, operator, measurement quantities and procedure and its accuracy, as well as the sensor position, if fixed. In Fusion4Decision the input / output variables and exposed parameters of a model are described in SensorML just like a traditional physical sensor.
- Observation & Measurement Model (O&M Model) is the core information model. It describes the relations between an observation, the associated feature of interest (what was observed) and the associated sensor procedure (how was the observation performed). The O&M Model can be extended to describe the uncertainty of observations or model calculations in a syntax called UncertML.

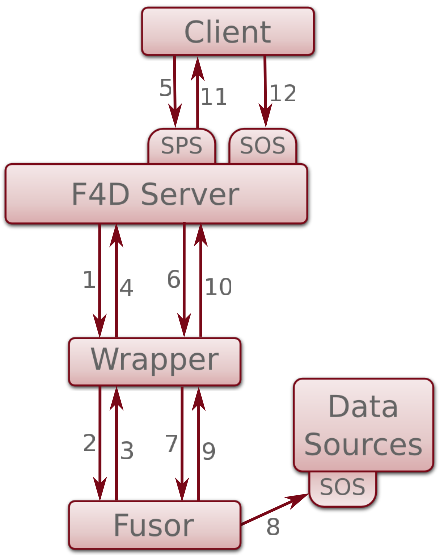

Figure 1. Architecure of Fusion4Decision

The OGC web services and information models are very generic and flexible, they provide full metadata of all geospatial information, but rely on rather complex XML schemas that can be difficult for non-specialists to apply in practice. On the other hand, scientists are used to working with data structures such as vectors, arrays and tabular data in CSV files, but not with data in XML syntax. We propose an approach that tailors OGC services and information models to the needs of scientists, allowing them to use their normal working environment. Thus models can be readily integrated into a service oriented web with a minimum of adaptation.

A model is considered to be a black box with defined set of input and output arguments. The input arguments include data as well as exposed model parameters. The output arguments are in general a collection of scientific data structures, maps and charts encoded as observations with complete metadata for further processing.

The software framework *Fusion4Decision* (F4D) shown in Fig. 1 uses the OGC web services SOS and SPS to provide a standard interface to processing algorithms, the so-called *Fusors*. Details of the usage of the OGC services SOS and SPS in this context are explained in Watson and Watson (2012). The term Fusor has been coined to describe a general fusion or processing of input data, including through a model based computation. A common fusor is the spatiotemporal interpolation of measurement data. The Fusor is written in any software code that can be integrated into a Java environment, such as MATLAB, R, Python, and C variants.

Since the SensorML required to describe a model is quite complex and verbose, the F4D system allows scientists to define the inputs and outputs of their model in JSON (JavaScript Object Notation, `http://json.org`), a lightweight data-interchange format that is both easy for humans to read and write and easy for machines to parse.

3 Model Definition

How the model is exactly invoked depends on the language the model is implemented in and each modelling language will require a specific wrapper to allow the model to be executed from Java. Regardless of the modelling language the model should implement the following functions *getDescription*, *getFeasability* and *run*. When the model is first loaded into the F4D server, the F4D server calls the *getDescription* function on the model, through the wrapper (arrows 1 & 2 in Fig. 1). The *getDescription* function returns (arrows 3 & 4) a formal description of the model, encoded in JSON as described below. Based on the JSON model description the F4D server generates the SensorML document that is used to present the model as a procedure in SOS and SPS services. For instance the JSON definition in example 1 is be translated into the SensorML shown partly in example 2. The *getFeasability* function checks, given a set of inputs, whether the model can complete a *run*. This function is always called before the *run* function, which executes the model with the given set of inputs.

When a client submits a job for the model through the SPS interface of the F4D server (arrow 5 in Fig. 1), the F4D server performs a first check of the inputs based on the model definition. If the inputs are in their defined bounds the server calls the *getFeasability* function on the wrapper with the inputs as parameter. The wrapper passes the call on to the model and returns the result back to the F4D server. If the run is feasible, the F4D server passes the inputs to the *run* function on the wrapper, which then starts the model (arrows 6 & 7). The model accesses any required data sources (arrow 8) and returns the result of the run back (arrows 9 & 10). The F4D server notifies the client that the run has finished (arrow 11) and the client can then access the results through the SOS interface of the F4D server (arrow 12).

The model description is encoded as a JSON object and consists of the following fields (cf. example 1):

name A short name for the model.

description A piece of text of arbitrary length describing the model in human readable text.

inputs The named inputs, each described by the following fields:

- **description** A piece of text of arbitrary length describing the input in human readable text.
- **annotation** Optional. A URI linking to an authoritative definition of this input in an ontology.
- **type** The type of the input. Allowed values are: scalar, string, time, URL, file.
- **size** Optional. If the input is an array this field gives the size of the array in each of its dimensions. The array is homogeneous, each element has the same type, unit, minimum, maximum, and allowed values. If the array is one dimensional this field is a single integer. If the array is multidimensional then this field itself is an array.
- **unit** The unit of the input. It is mandatory for scalar inputs, ignored for all other types.
- **default** Optional. The default value of the input.
- **minimum** Optional. The minimum value of the input. Only used for scalar inputs.
- **maximum** Optional. The maximum value of the input. Only used for scalar inputs.
- **allowedValues** Optional. A list of values that are allowed for this input. This can be used for all input types, including multi-dimensional arrays.

outputs The named outputs. The outputs are described by the fields: description, type, size and unit, in the same way as the inputs.

```
{
  "name": "example_1",
  "description": "A demonstration model with some example inputs.",
  "inputs": {
    "pressure": {
      "type": "scalar",
      "unit": "Pa",
      "description": "The pressure at t=0."
      "annotation": "http://sweet.jpl.nasa.gov/2.3/propPressure.owl#
        AtmosphericPressure",
      "default": 1.1e5,
      "minimum": 1.0e5,
      "maximum": 1.5e5
    },
    "pressureField": {
      "type": "scalar",
      "unit": "Pa",
      "description": "A 2 by 3 array of scalar values.",
      "size": [2,3],
      "default": [
        [1.1, 1.2, 1.3],
        [2.1, 2.2, 2.3]
      ]
    },
    "startTime": {
      "type": "time",
      "description": "The time, in ISO 8601, of the start of the run.",
      "default": "2013-07-05T12:14Z"
    },
    "covarianceModel": {
      "type": "string",
      "description": "A string, from a limited set.",
      "allowedValues": ["exponential","gaussian","linear"],
    },
    "samplingFeature": {
      "type": "URL",
      "description": "A link to the sampling feature.",
      "default": "http://f4d.server.de/features/12345"
    }
  },
  "outputs": {
    "evaporationField": {
      "type": "scalar",
      "unit": "m",
      "description": "The size of the array depends on the inputs.",
      "size": [10, 20]
    }
  }
}
```

Example 1: An example model definition in JSON

Inputs and outputs of type *time* are encoded as ISO 8601 strings, allowing for both dates and times. The URL type can be used to reference an external file or information resource (in any format understood by the model). The file type is used to reference files in the working directory of the model. For inputs of the type file, the file will be placed in the directory before model execution. For outputs of the type file, the file will be fetched from the directory after model execution and made available through a REST interface. The URL where the file can be accessed is added to the observation describing the model results. Typical output files are ASCII grids and images (jpg, png, etc.) For outputs that contain geospatial information the client can transfer the data to a Web Map Server to make the data available through an OGC WMS (Web Map Service) interface. Units should be specified in the UCUM format `http://unitsofmeasure.org/`. Additional information can be provided by semantic annotation, e.g. with the help of a SWEET ontology (`http://sweet.jpl.nasa.gov/`). The outputs of a model can not always be defined exactly. For instance, the size of an output array might depend on the inputs. In this case the size of the output array is not known.

```
<?xml version="1.0" encoding="UTF-8"?>
<sml:SensorML version="1.0.1" xsi:schemaLocation=...>
  <sml:member>
    <sml:ProcessModel gml:id="Process_1">
      <gml:methodName>example 1</gml:methodName>
      <sml:inputs>
        <sml:InputList>
          <sml:input name="pressure">
            <swe:Quantity>
              <gml:description>The pressure at t=0.</gml:description>
              <swe:uom code="Pa"/>
              <swe:value>1.1e5</swe:value>
            </swe:Quantity>
          </sml:input>
          <sml:input name="startTime">
            <swe:Time>
              <gml:description>The time of the start of the run.</gml...>
              <swe:value>2012-11-27T14:03:19.307+01:00</swe:value>
            </swe:Time>
          </sml:input>
          <sml:input name="covarianceModel">
            <swe:Category>
              <gml:description>A string, from a limited set.</gml:des...>
              <swe:constraint>
                <swe:AllowedTokens>
                  <swe:valueList>
                    exponential gaussian linear
                  </swe:valueList>
                </swe:AllowedTokens>
              </swe:constraint>
            </swe:Category>
          </sml:input>
        </sml:InputList>
      </sml:inputs>
    </sml:ProcessModel>
  </sml:member>
</sml:SensorML>
```

Example 2: Part of the model definition translated from example 1, but in SensorML. Only part of the inputs section is shown. Some lines are truncated.

4 Conclusions

The increasing proliferation of sources of geospatial data on the web as well as models to process the data and derive new information underlines the need for a standardised framework to better link data, models and their results. Standards of the Open Geospatial Consortium can be used to integrate data access and models into web services, thus being a step towards the Model Web in which scientists and decision makers can work together effectively. The paper proposes a simple way of describing the input and output arguments of a model using JSON. This JSON description can be readily understood and generated by model providers and also translated into the sensor description language SensorML. The latter is the basis for applying the sensor concept in the OGC standards SOS and SPS to models ("model as a sensor"). This approach bridges the gap between scientists and IT specialists. It is currently being validated in a team of scientists and IT experts for several application scenarios with different types of models with the aim of testing if the interfaces of typical models in the environmental domain can be conveniently described in the JSON format described here.

References

Argent, R. M. (2005, December). A case study of environmental modelling and simulation using transplantable components. *Environ. Model. Softw. 20*(12), 1514–1523.

David, O., W. Lloyd, J. C. Ascough II, T. R. Green, K. Olson, G. H. Leavesley, and J. Carlson (2012, July). Domain Specific Languages for Modeling and Simulation: Use Case OMS3. In R. Seppelt, A. Voinov, S. Lange, and D. Bankamp (Eds.), *International Environmental Modelling and Software Society, 2012 International Congress on Environmental Modelling and Software Managing Resources of a Limited Planet: Pathways and Visions under Uncertainty, Sixth Biennial Meeting, Leipzig, Germany*, pp. 1201–1207. iEMSs. http://www.iemss.org/society/index.php/iemss-2012-proceedings. ISBN: 978-88-9035-742-8.

Jagers, H. B. (2010). Linking Data, Models and Tools. In D. A. Swayne, W. Yang, A. A. Voinov, A. Rizzoli, and T. Filatova (Eds.), *International Environmental Modelling and Software Society, 2010 International Congress on Environmental Modelling and Software Modelling for Environments Sake, Fifth Biennial Meeting, Ottawa, Canada*. iEMSs. http://www.iemss.org/iemss2010/proceedings.html.

Jakeman, A., R. Letcher, and J. Norton (2006, May). Ten iterative steps in development and evaluation of environmental models. *Environ. Model. Softw. 21*(5), 602–614.

Kralisch, S. and C. Fischer (2012, July). Model representation, parameter calibration and parallel computing the JAMS approach. In R. Seppelt, A. Voinov, S. Lange, and D. Bankamp (Eds.), *International Environmental Modelling and Software Society, 2012 International Congress on Environmental Modelling and Software Managing Resources of a Limited Planet: Pathways and Visions under Uncertainty, Sixth Biennial Meeting, Leipzig, Germany*, pp. 1177–1184. iEMSs. http://www.iemss.org/society/index.php/iemss-2012-proceedings. ISBN: 978-88-9035-742-8.

Percivall, G. (2010, July). OGC and the Geosciences - IGARSS 2010 Tutorial, `http://www.ogcnetwork.net/GeosciencesTutorial`.

Rizzoli, A., G. Leavesley, J. Ascough II, R. Argent, I. Athanasiadis, V. Brilhante, F. Claeys, O. David, M. Donatelli, P. Gijsbers, D. Havlik, A. Kassahun, P. Krause, N. Quinn, H. Scholten, R. Sojda, and F. Villa (2008). Chapter Seven Integrated Modelling Frameworks for Environmental Assessment and Decision Support. In A. Jakeman, A. Voinov, A. Rizzoli, and S. Chen (Eds.), *Environmental Modelling, Software and Decision Support*, Volume 3, pp. 101–118. Elsevier. http://dx.doi.org/10.1016/S1574-101X(08)00607-8, ISBN: 978-0-08-056886-7.

Rizzoli, A. E., M. Donatelli, I. N. Athanasiadis, F. Villa, and D. Huber (2008). Semantic links in integrated modelling frameworks. *Mathematics and Computers in Simulation 78*, 412–423.

Watson, V. and K. Watson (2012, July). Design of a software framework based on geospatial standards to facilitate environmental modelling workflows. In R. Seppelt, A. Voinov, S. Lange, and D. Bankamp (Eds.), *International Environmental Modelling and Software Society, 2012 International Congress on Environmental Modelling and Software Managing Resources of a Limited Planet: Pathways and Visions under Uncertainty, Sixth Biennial Meeting, Leipzig, Germany*, pp. 1216–1223. iEMSs. http://www.iemss.org/society/index.php/iemss-2012-proceedings. ISBN: 978-88-9035-742-8.