2014

OpenSDM - An Open Sensor Data Management Tool

David Camhy

Günter Gruber

David Steffelbauer

Thomas Hofer

Dirk Muschalla

OPENSDM – AN OPEN SENSOR DATA MANAGEMENT TOOL

DAVID CAMHY (1), GÜNTER GRUBER (1), DAVID STEFFELBAUER (1), THOMAS HOFER (1), DIRK MUSCHALLA (1)
(1): Institute of Urban Water Management and Landscape Water Engineering, Graz University of Technology, Stremayrgasse 10/I, Graz, 8010, Austria

Exchange of scientific data and metadata between single users or organizations is a challenging task due to differences in data formats, the genesis of data collection, ontologies and prior knowledge of the users. Different data storage requirements, mostly defined by the structure, size and access scenarios, require also different data storage solutions, since there is no and there cannot be a data format which is suitable for all tasks and needs that occur especially in a scientific workflow. Besides data, the generation and handling of additional corresponding metadata leads us to the additional challenge of defining the "meaning" of data, which should be formulated in a way that it can be commonly understood to get out a maximum of expected and shareable information of the observed processes. In our domain we are able to take advantage of standards defined by the Open Geospatial Consortium, namely the standards defined by the Sensor Web Enablement, WaterML and CF-NetCDF working groups. Even though these standards are freely available and some of them are commonly used in specialized software packages, the adaption in widespread end-user software solutions still seems to be in its beginnings. This contribution describes a software solution developed at Graz University of Technology, which targets the storage and exchange of measurement data with a special focus on meteorological, water quantity and water quality observation data collected within the last three decades. The solution was planned on basis of long-term experience in sewer monitoring and was built on top of open-source software only. It allows high-performance storage of time series and associated metadata, access-controlled web services for programmatic access, validation tasks, event detection, automated alerting and notification. An additional web-based graphical user interface was created which gives full control to end-users. The OpenSDM software approach makes it easier for measurement station operators, maintainers and end-users to take advantage of the standards of the Open Geospatial Consortium, which usage should be promoted in the water related communities.

INTRODUCTION

Starting point for developing the OpenSDM (Open Sensor Data Management) software solution have been experiences in long-term sewer monitoring in Austria (see Gruber *et al.* [1] and Gruber *et al.* [2]). Observed hydraulic and water quality data like flow, turbidity, conductivity, TSS (Total Suspended Solids) or COD (Chemical Oxygen Demand) of several sewer-monitoring stations in Austria have been recorded continuously (3 minutes time step during dry weather and 1 minute time step during wet weather conditions) and stored since 2002. These

datasets not only consist of one-valued time-series but also contain absorbance spectra (fingerprint data) for each timestamp, due to the use of UV-Vis spectroscopy sensors (Langergraber *et al.* [3]). Additionally, the storage of hydro-meteorological data, simulation results and other complex algorithms became more and more important during the last decades. Storage requirements are growing and the management of the actual measurement data and its corresponding metadata has become a challenge, which we are trying to meet with the OpenSDM software solution.
Moreover, different scientific workflows can cause different requirements on data storage technologies, mostly defined by the nature, size and access scenarios of the datasets.

Data storage

On the one hand, the direct accessibility of a complete time-period of a dataset and its corresponding metadata using different programming languages can be required for a subsequent scientific analysis and validation of the data. This can be a time-consuming task, since often a first visual analysis supported by domain and expert knowledge of the analyzing people is necessary to select suitable algorithms, parameters and ranges for validation. Therefore, data-access in real-time is not always necessary. In fact, these types of users generally want to have offline access to the datasets to i) be able to avoid possible restrictions in terms of speed or accessibility and to ii) run an analysis, e. g. special kinds of event detections also weeks or months after the actual measurements.

On the other hand, automatic notification and alerting tasks, which are frequently part of a measurement workflow, need to react in real-time but typically only access a small time-period of the dataset. Depending on the kind of intended application, querying and aggregation capabilities of the data store can be important. Taking the amount of measurement stations and their observed variables into account the data store has to be dimensioned accordingly in terms of insert-performance and storage size.
Furthermore, user expectations also play a crucial role in choosing the right data storage technology. In times of web portals, which serve millions of people simultaneously and give access to a great amount of information in near-real-time, it is often expected by end-users that they can access arbitrarily large datasets in the shortest amount of time only limited by the speed of their internet access.

Data formats and storage solutions change over time depending on the technological advancements and our expectations on them. Different working groups are interested in different kinds of datasets. They work with them differently and have different expectations on performance, availability and quality of the data. Of course, this is mirrored by their chosen software solution. The OpenSDM software framework was designed to be independent of the actual data storage technology. Nevertheless, a data store was designed which is tailored at our needs of high-performance time-series storage.

Data exchange

While stepping up to the next advancement of storage technology or exchanging data with other people, one has to guarantee that as little information as possible is lost when transferring data from one data store to the other. By giving “meaning” to the data the loss of information can be minimized. This “meaning” is often referred to as metadata, which is attached to the actual datasets.
By using standards for the exchange of data and metadata we can make sure that this “meaning” is commonly understood. By using concepts from the semantic web and commonly available ontologies (e.g. the Semantic Web for Earth and Environmental Terminology “SWEET” from Raskin and Pan [4]), when describing our available datasets and sensors we can build on the experience of others and further simplify the exchange of metadata. In our domain we are also

able to take advantage of the standards developed by the Open Geospatial Consortium (OGC). One of their projects, namely the *Sensor Web Enablement*/SWE (Botts *et al.* [5]) initiative provides standard encodings for sensor system and process descriptions (SensorML), a schema for encoding Observation and Measurements (O&M) and amongst others web services for sensor/process tasking and discovery. The OGC standard WaterML (Taylor [6]) provides a standard to exchange many kinds of hydro-meteorological observations and measurements. Also the netCDF data format (Rew and Davis [7]) and the accompanying netCDF Climate and Forecast Metadata Conventions (Eaton *et al.* [8]), are widely used across the worldwide scientific community and bring a whole ecosystem of applications and server solutions with it. NetCDF has also become an official OGC Standard in 2011 (refer to Domenico [9]) and currently there are discussions to allow a possible use of netCDF as representation of WaterML time series data (Palmer [10])
The XML-Encoding of the SensorML (Botts and Robin [11]) and WaterML ((Taylor [6])) standards, which was the main focus in our first release of the OpenSDM platform, allows linking and annotations (Maué [12]) to external dictionaries for definition of parameters and terms. This facilitates the integration of semantic web standards.

METHODOLOGY – THE OPENSDM APPROACH

The OpenSDM software solution is built on the following various principles which were identified to be crucial for a sustainable data management at our institute:

- OpenSDM is based on open-source technologies only to be independent of commercial software vendors.
- The framework itself is independent of the underlying data storage technology.
- Use of standards wherever possible to not reinvent already proven technologies and to facilitate a standardized data exchange.
- OpenSDM should be easily accessible by scientific users, as well as by end-users that want a GUI for the data and metadata.

The software consists of several parts which will be described in the following corresponding chapters.

The OpenSDM Graph Engine

The OpenSDM Graph Engine is the basis of the OpenSDM software solution. The Graph engine is an implementation of a property graph model consisting of vertices and relationships. So every vertex or relationship can have an arbitrary number of properties and there are an arbitrary number of relationships between these vertices. All data, except the actual measurement data, is kept in this graph model. A simple graph schema validation engine was built which is able to validate properties and relationships. Every vertex or relationship is labeled with one or more concept types and every property can have restrictions depending on the concept it is applied to.

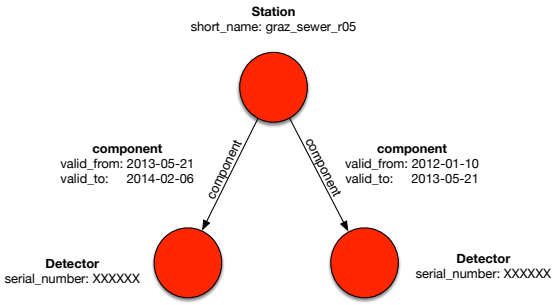

Figure 1. A simplified example of the implemented property graph model describing relationships between a measurement station and two detector components. By defining relationship properties it is easy to model a time-dependence.

This schema engine is not an attempt to build an ontology or knowledge representation language. It is a way to allow modeling domain specific concepts, relationships and properties in an extensible way. That means that even end-users are allowed to extend and edit the schema through a web-based graphical user interface. So for example the end-user can define new types of measurement stations, sensors, detectors and their associated properties. On the other hand, these domain specific elements can be annotated with links to external vocabularies and ontologies to allow users to take advantage of existing semantic descriptions. These annotations are neither validated nor are there any constraints to them.

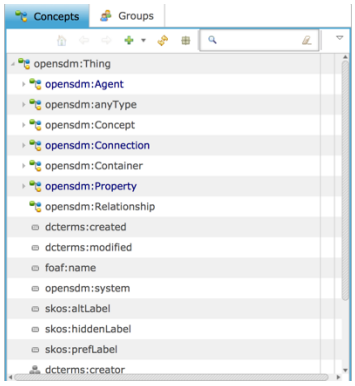

Figure 2. Screenshot of the OpenSDM concept browser. For the purposes of demonstration here some properties were created that were annotated with links to external sources.

An integrated ontology browser helps users to find and link to external ontologies (Figure 3).

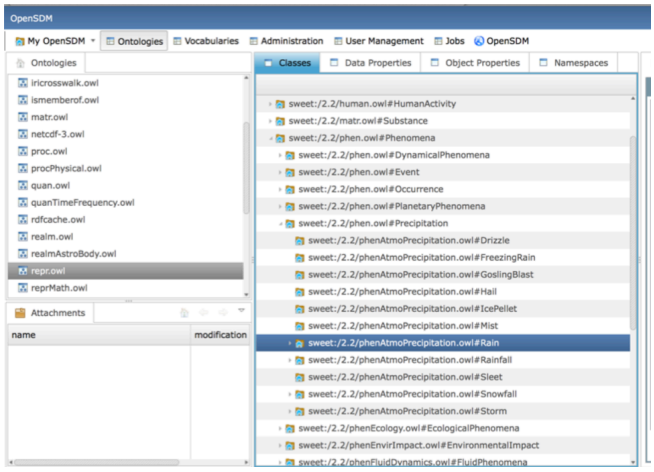

Figure 3. Screenshot of the integrated ontology browser with an excerpt of the SWEET ontology.

Besides defining and extending the schema, administrative users are also able to define property and relationship templates, which act as a basis for data input and output. These templates can be registered by concept type and are also access controlled, which implies that specific user types are only allowed to edit or view specific properties.
They can be rendered dynamically by the Web-GUI and provide the users forms for data input. On the other hand, these templates can also be accessed programmatically to help generating defined output formats like SensorML or WaterML.

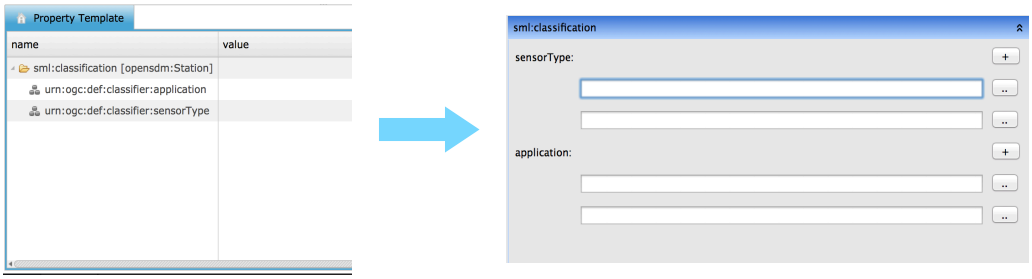

Figure 4. Template of a SensorML classification element, which is for the purpose of demonstration registered for concepts of type “opensdm:Station”. The form based user interface is dynamically rendered based on the definition of the template.

Time-Series Data Storage

Storing data in an efficient and versatile way is not an easy task, because of the different kinds of datasets that appear especially in a scientific workflow and the different ways of data access. Furthermore the size of the datasets increased during the last decades and many different technological achievements were made to handle this vast amount of data.
In this field we are concentrating on small to mid-sized time series datasets where each dataset can consist of some ten millions of entries and values cannot only be scalars but also relatively small arrays.
Since the OpenSDM-system provides a storage-independent layer (optimized for time series data in the first release), we are able to choose between several different data storage technologies:

1. **Relational Databases**
   Relational databases seem to be a good fit for storing time-series data of multi-dimensional values, since their indexing technologies are very advanced and some of the modern database solutions also provide specific array types. However, our tests revealed some problems with tables consisting of tens of millions of rows. Therefore we were forced to change our database schema to store more than one timestamp in a row, which had some consequences on memory consumption, because now also a lot of arrays are stored in the same row. While it probably would have been possible to build a database schema, which would be targeted at our needs, we have also been looking at alternative solutions.
2. **Scientific Data Formats**
   Scientific data formats like netCDF (Rew and Davis [7]) or HDF5 (HDF Group [13]) are tailored to store multi-dimensional data and provide very good performance when accessed locally. These datasets can also easily be used offline and so they seemed to be the ideal storage solution for our application. There exists a whole ecosystem of applications and server-solutions that provide access to the data e.g. by using THREDDS (Thematic Realtime Environmental Distributed Data Services; Domenico *et al.* [14]), which provides access through OpenDAP, WMS, http and other protocols. Also web applications like ERDDAP (Environmental Research Divisions Data Access Program; Simons [15]) offer web services that provide access to data stored in these formats. Metadata can be put directly into the datasets and there also exists a widely used metadata vocabulary (CF-Convention [8]). Problems occurred because parallel writing of these datasets is not easily achievable. When receiving data from many measurement stations at the same time, the access to the files has to be synchronized or there has to be some kind of caching solution which accepts the data in real-time and creates these files afterwards.
3. **Dedicated Timeseries Databases**
   The emergence of open-source time-series databases seems to go hand in hand with the need of operators of big computer clusters to monitor their servers. While in the financial market very common, dedicated time-series databases are rare in the open-source environment. Tools like rrdtool [16] and graphite [17] provide fixed-sized databases, which deliver excellent performance when plotting long periods of time-based data with the compromise that the data is aggregated in the background, resulting in a loss of the older data. Newer achievements like OpenTSDB [18] and KairosDB [19] do not aggregate the data and are built on modern NoSQL solutions like HBase [20] and Cassandra [21]. In our tests these solutions provided the best performance and seemed to easily handle any amount of scalar data that we put inside. Furthermore, they provided excellent aggregation capabilities, which could be used for delivering end-users near-real-time plotting experiences. KairosDB in its newest incarnations also provided a way to define custom data types, which we were able to use in our development to store time-dependent array data.

The currently used storage solution is based on KairosDB with a custom-built backend that was newly developed together with the OpenSDM system. This backend is very lightweight, consists only of some hundred lines of code and uses the MapDB [22] library, which is used with the same principles that the original Cassandra backend is built on. It delivers satisfying performance for our application needs and data size. Using the KairosDB core allows us to use all the other features like aggregators and the webservice-endpoints with our own solution. On the other hand, this data is also exported to NetCDF using the CF-Conventions, allowing end-users the offline usage of these datasets and to guarantee a standardized way to exchange datasets.

Scheduling and Events

A data management system has to handle a lot of recurring tasks that have to be carried out at certain moments of time. Examples for these recurring tasks are the transfer of data from the measurement stations, validation tasks or export operations. Other tasks like alerting or notification are dependent on certain events like errors on transfers, warnings on data values that have to stay in a certain range or failures in measurement systems.
The OpenSDM framework delivers a scheduling and event-framework, which can be extended by plugins or scripts from users. These plugins and scripts read their parameters from job instance vertices that are defined in the graph engine. The GUI enables users to change job intervals and event-parameters.

The REST (Representational State Transfer) Interface

Every element of the graph (vertices and relationships) and also every dataset and its components can be seen as resources, which are accessible by an URL with their unique IDs. By using HTTP request methods like GET, POST, PUT and DELETE, every element can be accessed and modified by using standard HTTP clients (Figure 5). The OpenSDM system defines interfaces, where plugins can implement service components, which can be registered with one or more concept types, HTTP methods and selectors that are mirrored as elements of the URL. For example, there is a service component that provides CSV output of time-series datasets, which is registered for concepts of type “time-series” and the selector “csv”. URL parameters are accessible by the component and can provide more information for the intended operation.

METHOD	RESOURCE	SELECTOR	PARAMETERS
GET	http://<host>/<opensdm_id>	.json (.rdf,.xml)	?parameter1=value1&parameter2=value2

Figure 5. Example for element access

OGC Standards like SensorML and WaterML are already implemented as components of the software. In this case the rendering component, which outputs the resulting xml-File also uses property templates that are registered for the accessed concept type.

CONCLUSIONS

Storage and exchange of continuously recorded and preferably validated time series data and corresponding metadata between single users or organizations is a very challenging task but a necessary and desirable basis to describe environmental processes and occurring phenomena to get out a maximum of resulting information.
By developing the platform independent OpenSDM software solution we make it easier for measurement station operators, maintainers and end-users to handle, store and exchange collected online time series data. They can take advantage of several standards for data and metadata exchange. By designing the software with a user extendible schema engine and a storage independent abstraction layer it is adaptable to various scientific workflows and requirements. In the future the implementation of standards like SOS (Sensor Observation Service) and SPS (Sensor Planning Service) is planned.

REFERENCES

[1] Gruber G., Bertrand-Krajewski J.-L., De Bénedittis J., Hochedlinger M., Lettl W. “Practical aspects, experiences and strategies by using UV/visible sensors for long-term sewer monitoring”. *Water Practice & Technology* Vol. 1 No. 1 (2006), pp 8.

[2] Gruber, G., Winkler, S. and Pressl, A. “Continuous monitoring in sewer networks an approach for quantification of pollution loads from CSOs into surface water bodies.” *Water Science and Technology* Vol. 52, No.12, (2005), pp 215-223.

[3] Langergraber G., Fleischmann N. and Hofstaedter F. “A multivariate calibration procedure for UV/VIS spectrometric quantification of organic matter and nitrate in wastewater.” *Water Science and Technology* Vol. 47, No. 2, (2003), pp 63-71

[4] Raskin, Robert G. and Michael J. Pan. “Knowledge Representation in the Semantic Web for Earth and Environmental Terminology (SWEET).” *Computers & Geosciences* Vol. 31, No. 9, (2005), pp 1119–25.

[5] Botts, M., Percivall, G., Reed, C. and Davidson, J. “OGC® Sensor Web Enablement: Overview and High Level Architecture.” (2007) http://portal.opengeospatial.org/files/?artifact_id=25562 (accessed 27 March 2014)

[6] Taylor, P. “OGC® WaterML 2.0: Part 1- Timeseries” (2014) https://portal.opengeospatial.org/files/?artifact_id=57222 (accessed 27 March 2014)

[7] Rew, R. and Davis, G. “NetCDF: an interface for scientific data access.” *Computer Graphics and Applications, IEEE*, Vol. 10, No. 4, (1990), pp76-82.

[8] Eaton, B., Gregory, J., Drach, B., Taylor, K., Hankin, S., Caron, J., Signell, R., Bentley, P., Rappa, G., Höck, H., Pamment, A., Juckes, M. NetCDF Climate and Forecast (CF) Metadata Conventions Version 1.6, (2011) http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.6/cf-conventions.html (accessed 31 March 2014)

[9] Domenico, B. “OGC Network Common Data Format (NetCDF) Core Encoding Standard version 1.0”. http://portal.opengeospatial.org/files/?artifact_id=43732 (accessed 27 March, 2014)

[10] Palmer, D. “WaterML 2.0 – Timeseries – NetCDF Discussion Paper”. http://www.opengis.net/doc/DP/waterml-timeseries-netcdf (accessed 27 March, 2014)

[11] Botts, M. and Robin, A. (2007) OpenGIS® Sensor Model Language (SensorML) Implementation Specification. http://portal.opengeospatial.org/files/?artifact_id=21273 (accessed 20 March 2014)

[12] Maué, P. “Semantic annotations in OGC standards”. https://portal.opengeospatial.org/files/?artifact_id=34916 (2009) (accessed 27 March 2014)

[13] The HDF Group http://www.hdfgroup.org/HDF5/ (2013) (accessed 20 March, 2014)

[14] Domenico B., Caron, J., Davis, E., Kambic, R. and Nativi, S. “Thematic Real-time Environmental Distributed Data Services (THREDDS): Incorporating Interactive Analysis Tools into NSDL.” (2002), *Journal of Digital Information* Vol. 2 No. 4

[15] Simons, B. “ERDDAP” http://coastwatch.pfeg.noaa.gov/erddap/index.html (2014) (accessed 20 March 2014)

[16] Oetiker, T. “RRDTool” http://oss.oetiker.ch/rrdtool/ (2014) (accessed 31 March 2014)

[17] “Graphite – Scalable Realtime Graphing”, http://graphite.wikidot.com/

[18] “OpenTSDB” http://opentsdb.net/ (2014) (accessed 20 March 2014)

[19] “KairosDB” https://code.google.com/p/kairosdb/ (2014) (accessed 20 March 2014)

[20] The Apache Software Foundation. “Apache HBase” https://hbase.apache.org/ (2014) (accessed 20 March 2014)

[21] The Apache Software Foundation. “Apache Cassandra” http://cassandra.apache.org/ (2014) (accessed 20 March 2014)

[22] “MapDB – java beyond heap” http://www.mapdb.org/ (2014) (accessed 20 March 2014)