EPiC Series in Engineering
Volume 3, 2018, Pages 2000–2010
HIC 2018. 13th International Conference on Hydroinformatics

Technical Breakdown of a Time-Series Data Federation system

Ashley Sommer[1], Matt Stenson[1], Ross Searle[1]

[1] Commonwealth Scientific and Industrial Research Organisation, Brisbane 4102, Australia

Corresponding author: ashley.sommer@csiro.au

Abstract. A very large volume of climatic and agricultural data is captured and recorded by on-farm monitoring devices that is uploaded to various different data service providers. It is consistently difficult for land managers to discover, access, understand and use the data due to its disparate nature, limited access to it and multiple proprietary formats used. The *Soil Sensing* project is developing tools and technologies to help improve the ability to discover, access, understand, and use time-series farm-scale data across disparate data providers. This is achieved by the development and deployment of loosely-coupled web services in the form of a Data Streams Integrator system (DSI), which implements a combined brokered and federated data supply chain pattern. The DSI is composed of the Data Brokering Layer, the Observations and Measurements translation layer, the Sensor Observation Service interface and a metadata registry and repository.

Keywords: Soil, Agriculture, Farm data, Metadata, Federation, Observations & Measurements, Sensor Observation Service, O&M, SOS, DBL, TimeseriesML, SensorML

1 Introduction

Over the past decade in Australia, the amount of farm-scale time-series data collected and stored has increased immensely, due to the decreasing cost and increasing prevalence of easy-to-deploy and easy-to-use commercial, on-farm, sensor devices [1]. Meteorological data such as temperature, humidity, rainfall and wind-speed, as well as agricultural domain properties including soil moisture, soil temperature and other soil properties are regularly uploaded to sensor data services [1]. The end user is usually given access to the gathered data only via a web interface, and often in a restricted or limited manner. When a downloadable format or an API endpoint is provided, it typically uses a proprietary data format, often via a bespoke interface with ad-hoc API calls. Modern farms in Australia often have multiple sensor devices deployed from various different manufacturers and, as a result, the farm data are stored across multiple, different services. It is consistently difficult for land managers to discover, access, understand and use the data due to the disparate nature, limited access and multiple proprietary formats used by sensor data service providers.

The *Soil sensing – new technology for tracking soil water availability, managing risk and improving management decisions* project jointly funded through *CSIRO* and the *Australian Government's National Landcare Program* (the *Soil Sensing* project) has developed a system to federate access to meteorological and agricultural data across various disparate data service providers. This is achieved by the development and deployment of loosely-coupled services in the form of a Data Streams Integrator (DSI) system [1]. The goal of the DSI is to provide an implementation of a combined Brokered and Federated [3] data supply chain pattern as described in *Data Specification Framework for the Foundation Spatial Data Framework* [3].

2 The Data Stream Integrator Architecture

The standards and formats to be adopted and used by the components of the DSI were determined by a workshop comprising stakeholders from industry, academic partners and client application developers. It was decided to base the system on published Open Geospatial Consortium (OGC) standards, specifically, using the Observations and Measurements [13] data model, with embedded TimeseriesML [14] and SensorML, and exposed via a Sensor Observation Service interface. It was agreed that the use of open published standards is key to encourage the adoption of the DSI.

Components of the DSI (Figure 1) include:

- Data Brokering Layer (DBL) – This concept was developed in the eReefs project [2]. In the DSI, the DBL acts as a *metadata* harvester, metadata cache and semantic search tool. Used for discovering available data sources, it provides a machine-readable linked-data-compatible API for searching and discovering all known datasets in a consistent format.
- SOS Native Data Service (SOS service layer) - An implementation of the Open Geospatial Consortium Sensor Observation Service (SOS) Interface Standard, that enables client applications to interface with using existing standard client libraries.
- O&M Translator (O&M translation Layer) – A data broker consumes the bespoke and proprietary observational data formats used by the various data service providers and presents it to clients through the SOS interface using the standard Observation and Measurements (O&M) data model. Optionally, the

O&M Translator has the ability to deliver a TimeseriesML representation of observation results for a cleaner, less verbose output.

- Additional Metadata (SinNC Metadata Repository) - An external and stand-alone metadata registry and repository created to augment the limited Data Provider metadata captured by the DBL. This was created to be a general-purpose tool, not specific to this use-case or this project.

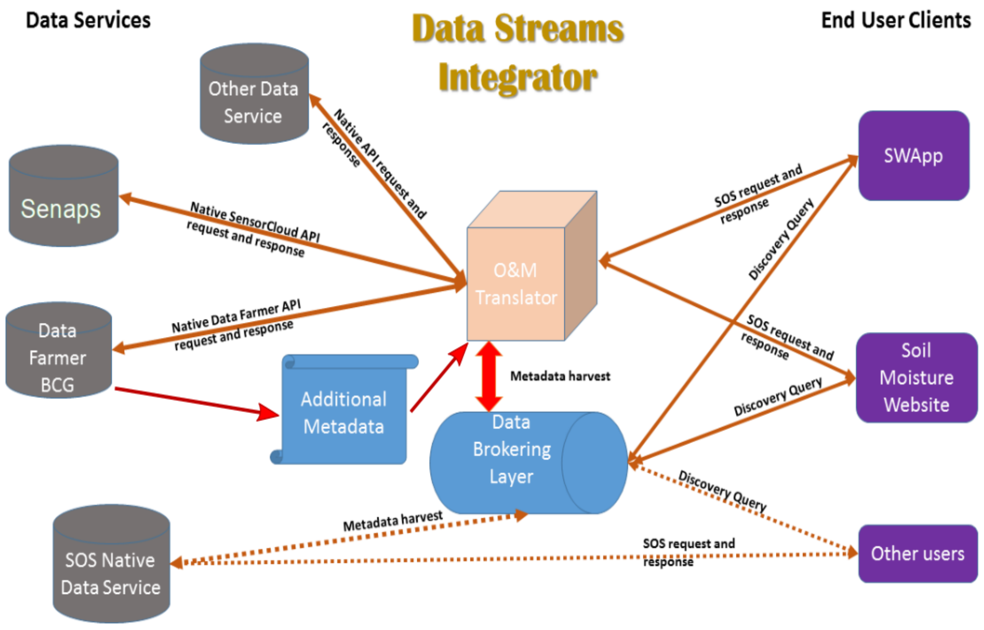

Figure 1 - A simplified visual representation of the components within the Data Streams Integrator system.

The task of bringing together disparate sources of data, with the goal of making it visible and accessible via a common interface in a common format, is not new. The FSDF project analysed a variety of data supply chains and proposed a taxonomy based primarily on the location of the transformation process within the architecture (Figure 2). The Data Streams Integrator is a hybrid brokered/federated system, incorporating aspects of both rows 4 and 5 in the diagram. The "federated" scenario represents the provision of a native SOS interface directly by the data provider for the end user to consume. Most data service providers, however, do not implement systems in this way, therefore to utilise the data in a SOS format, it is processed by the O&M Translation Layer matching the "brokered" scenario in Figure 2.

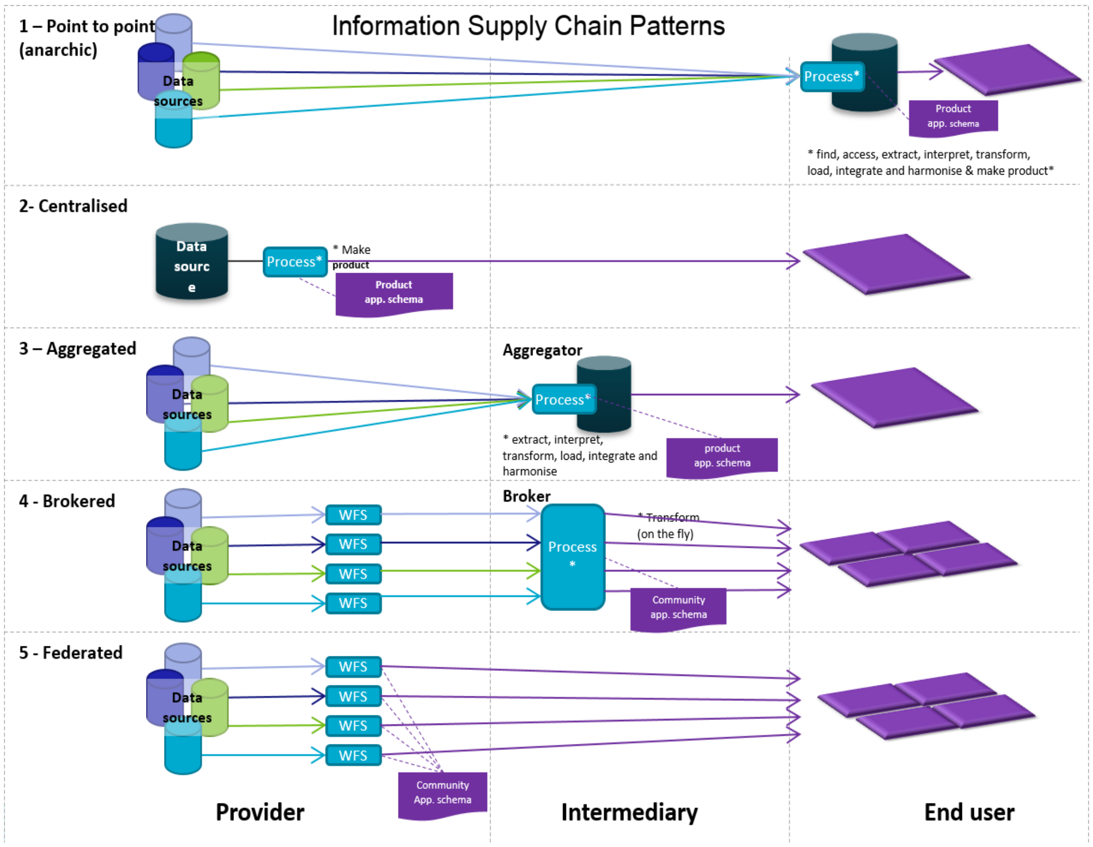

Figure 2 - Supply Chain Patterns, after Figure 5 in [3]

2.1 Data Brokering Layer

Originally developed as part of the eReefs project in 2013-2014, the Data Brokering Layer (DBL) [2] is a middleware service that acts as a *metadata* harvester, metadata cache, metadata mediator and semantic search engine. The service is implemented in Python as a Flask[1] web service. Known data providers are described as Data Provider Nodes (DPNs) using the DPN Ontology [4]. The DBL is able to gather a specific subset of metadata from many data provider service types using a number of different harvester implementations. The DBL harvester periodically queries defined service endpoints of known data providers for a catalogue of datasets, then extracts a selection of metadata from each to store in a common semantic format within the DBL metadata cache. The metadata cache is implemented using a MongoDB[2] document storage database both in-memory and on-disk. The DBL metadata cache

1 http://flask.pocoo.org

2 https://en.wikipedia.org/wiki/MongoDB

provides the capability for users of the DBL to access a mediated aggregation of metadata for all datasets from all known service providers. This enables development of lightweight client applications, which can simply use a single API endpoint of aggregated metadata to power their functionality. The DBL also allows users to perform semantic search functions across the metadata cache to discover specific datasets, rather than rely on simple keywords filtering or consuming a broad set of unfiltered data layers, therefore streamlining discovery and access to relevant data.

The eReefs implementation of the DBL was ported to a newer version of Python and support was added for non-gridded data provider sources, specifically time-series data sources.

Both gridded and time-series dataset types are now supported by the DBL. In the case of previously supported gridded datasets, metadata harvesters have been implemented for several THREDDS Data Server (TDS) [5] service types including ISO [6], OpenDAP [7] and Web Map Service (WMS) [8]. For newly supported time-series datasets, metadata harvesters have been implemented for Senaps (formerly Sensorcloud) [9], DataFarmer [10], SILO Climate data from the Queensland Department of Science, Information Technology and Innovation [11] and the WeatherStation API from Western Australia Department of Agriculture and Food (DAFWA) [12].

2.2 The O&M Translation Layer

The majority of new work for the *Soil Sensing* project was development of the O&M Translation Layer. It works closely with the SOS interface described in Section 2.3 below. Translation of data representation from the native forms to O&M is implemented using Object-Oriented Python according to the O&M standard [13]. This component is built with a modular, extensible, backend system, with support for each different data provider written as drop-in modules. At the time of writing, backend connector modules have been written for Senaps [9], DataFarmer [10], SILO [11] and DAFWA [12].

The SOS interface receives requests for observations with optional temporal and spatial filters, or filters for other properties, from client applications and passes them to the O&M Translation layer. This component determines the correct data source to service the request, and uses that backend to retrieve the required data from the provider in the provider's native format. After receiving the results, the component reshapes the data (using Pandas Dataframes[3]), into a structure that can be used to

3 https://pandas.pydata.org/pandas-docs/stable/dsintro.html

build the O&M observation data representation. For each backend there is also the option to construct a TimeseriesML [14] representation of the observation results, which provides a less verbose response payload.

Finally, the O&M translation layer adds additional required metadata to the response, serialises it to the required format and passes it back to the SOS interface. The default O&M serialisation format is XML+GML [15]. However, an RDF/OWL representation [16] is partially supported and a JSON implementation [18] is planned.

2.3 The Sensor Observation Service

The SOS component is the primary and preferred way for clients to interface with the system. It is built in accordance with the required SOS Interface Standard [19] using SOAP bindings, and has two other service bindings; KVP (key-value-pair) and REST. The SOAP binding is implemented using Spyne; a Python Remote-Procedure-Call (RPC) service library for building RPC and SOAP interfaces. The KVP and REST bindings are built using Sanic, a fast, modern, asynchronous micro-service library. While the SOAP and KVP bindings are defined by the SOS standard, the REST binding was developed locally, based on the KVP binding specification, with a REST-like interface documented using the SwaggerUI (OpenAPI)[4] presentation tool. All three of the bindings implement four standard SOS endpoint functions; *GetCapabilities*, *GetObservation*, *DescribeSensor* and *GetFeatureOfInterest*. Each binding can receive a request from a client, ensure it is in the correct structure with valid parameters, present the request to the O&M Translation layer for execution by the appropriate backend implementation, and finally return the response to the client.

2.4 The Metadata Repository

The newest component of the DSI is the Metadata Repository (SinNC). In most cases the metadata provided by the data services is extremely lightweight and not backed by standardised and governed terms, leading to ambiguity and decreasing the potential for the data to be understood, trusted and used. As part of the DSI system we have implemented a flexible metadata system based on RDF (Resource Description Framework[5]), that allows for the capture of extra context information about either the data, the sensor or its use through provenance.

4 https://en.wikipedia.org/wiki/OpenAPI_Specification

5 https://www.w3.org/2001/sw/wiki/RDF

The repository's purpose is to store *all* of the metadata about data service providers known to the DSI, plus metadata about datasets exposed by the service providers. This is required because the DBL metadata cache stores only the smaller subset of metadata that is required to perform DBL functions. The *DescribeSensor* and *GetFeatureOfInterest* endpoints on the SOS interface are able to generate SOS responses that contain descriptive metadata models using SensorML [16] and GML [17]. The extended metadata used to populate the SOS metadata models is retrieved from our Metadata Repository, on-demand, at runtime.

The Metadata Repository is populated using harvester scripts that periodically crawl known data provider backends. Populating SinNC for a particular data service is normally a one-off job involving mapping existing metadata to managed vocabularies, and linking to extra metadata that may be known, but not exposed, through the existing service.

SinNC is built using modern RDF and Linked Data[6] technologies. The database is a RDF Quad-store implementation powered by Apache Jena Fuseki[7], with a Python service interface to allow easy metadata searching, storing and retrieving.

3 Retrospective

It was decided early that the DSI's O&M and SOS layer implementations should be as light-weight as possible, supporting only the features needed to get a minimum compliant O&M and SOS output generated. This meant forgoing existing open-source implementations like *52 North* [20], and writing a bespoke O&M and SOS stack from scratch. However, even a lightweight implementation was a much bigger task than originally anticipated, especially when introducing support for additional technologies such as TimeseriesML and SensorML. It required implementing not only the externally visible O&M and SOS class models, but also the internally used SWE, SWES, OWS and GML class models that O&M and SOS are built upon. This has advantages over using existing off-the-shelf solutions in terms of flexibility and customisation, at the expense of additional development time.

An alternative service interface standard came to the attention of the project team when the OGC published the SensorThings API specification [21]. While too late to change the project scope and direction at that time, it was recognised that the SensorThings API may have been a better choice than SOS for the service interface. It would have allowed the ability to keep the O&M data model, but have a faster and

6 https://www.w3.org/standards/semanticweb/data

7 https://jena.apache.org/documentation/fuseki2/

more light-weight interchange format, and would have been easier to implement client-side applications for consumers of the service.

4 Looking Forward

The project team is working toward having data from more Data Providers exposed through the Data Streams Integrator. While the existing backend connectors are expected to cover the majority of the known use cases, there will be instances that will require writing new modules for the O&M Translation layer. We will also work with data providers to directly provide O&M and SOS service interfaces by default so that no on-the-fly translation is necessary. This will allow us to move more toward the ideal Federated Supply Chain scenario.

There are currently several Data Providers that do not provide a publicly accessible, fast and easy-to-use API that can be accessed at runtime for real-time data translation through the O&M translation layer. For these cases recent data is regularly "screen-scraped" from the service provider ahead of time and stored in a cloud-based Senaps instance. The translator then uses Senaps as the data source at runtime for O&M translation. This solution is less than ideal because duplicating, storing and maintaining Data Providers' data is an additional cost to the project. The project team is working with Data Providers to get access to APIs that can be used at runtime, to avoid scraping the data ahead of time.

The project team is working with client application developers, and end-users to test our proposition that this style of system adds value in the form of easier to access data, aggregated views of data across multiple providers, and time savings.

There are currently no usage metrics captured by the Data Streams Integrator nor any of its components thus it is not possible, at this stage, to analyse usage of the system. It is anticipated that the addition of detailed usage metrics will allow the team to fine-tune the features, capabilities, and performance of the system in response to how it is being used.

A final thought is the potential to move the whole Data Streams Integrator to an operational home at the end of its development period. That would allow the development team to hand off the day-to-day running of the system to a dedicated operational team.

5 Conclusions

Through the adoption and extension of existing standards, technologies and the application of new technologies, a function time-series data stream aggregation system that cleanly matches the ideologies of a brokered and federated Information Supply Chain was built. The Data Brokering Layer was successfully re-purposed and extended to facilitate the harvesting of time-series data from new data service providers, and the new functionality kept available for use in other projects. The new O&M Translator component creates valid OGC compliant representations of data sourced from a variety of data provider services, and is able to communicate with client applications through a standardised SOS interface. The new RDF-based Metadata Repository augments the SOS capabilities with additional metadata to add important context to the Sensor and Observation data. Work on the system is continuing; tasks to add new functionality including adding support for more backend service provider types, and implementing new O&M representation types is planned. However, the system in its current form is deployed and working as intended, with clients in the Agriculture sector and mobile application developers using it in an evaluation and testing capacity. Figure 3 shows a screenshot of an example web application used to showcase the capabilities of the DSI.

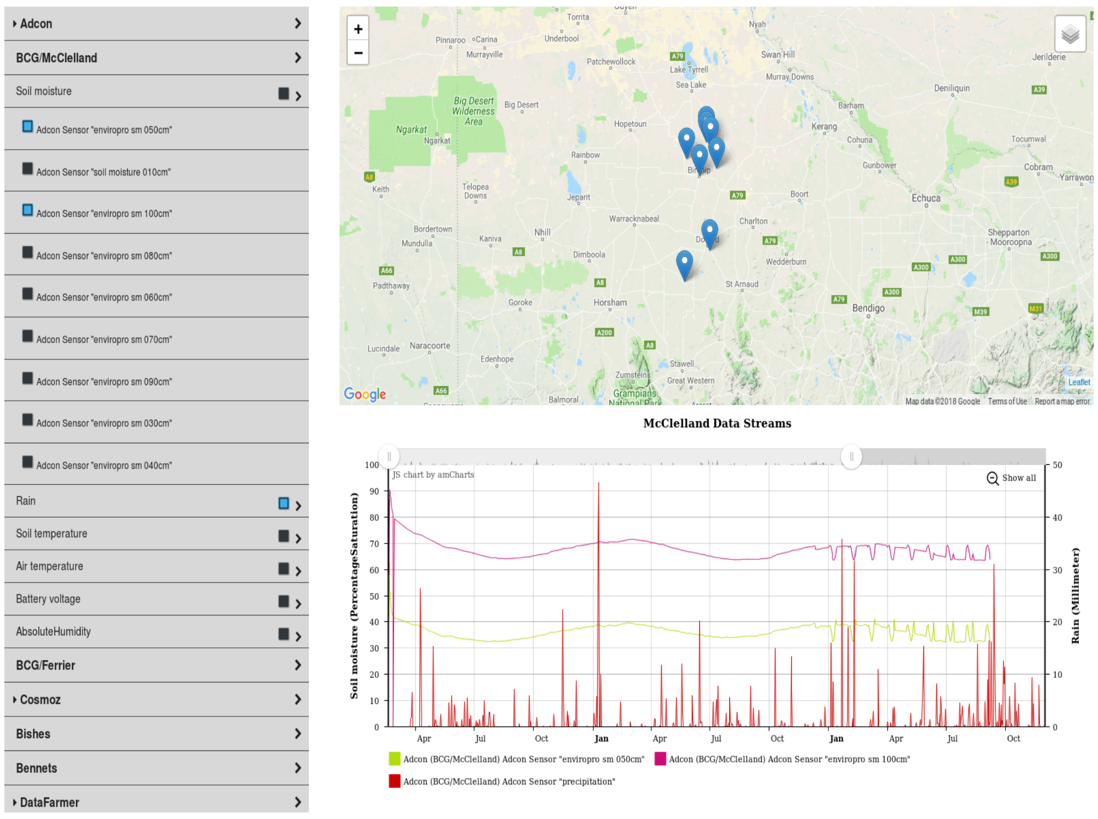
Adcon
BCG/McClelland
Soil moisture
Adcon Sensor "enviropro sm 050cm"
Adcon Sensor "soil moisture 010cm"
Adcon Sensor "enviropro sm 100cm"
Adcon Sensor "enviropro sm 080cm"
Adcon Sensor "enviropro sm 060cm"
Adcon Sensor "enviropro sm 070cm"
Adcon Sensor "enviropro sm 090cm"
Adcon Sensor "enviropro sm 030cm"
Adcon Sensor "enviropro sm 040cm"
Rain
Soil temperature
Air temperature
Battery voltage
AbsoluteHumidity
BCG/Ferrier
Cosmoz
Bishes
Bennets
DataFarmer
McClelland Data Streams
JS chart by amCharts
Show all
Soil moisture (PercentageSaturation)
Rain (Millimeter)
Apr
Jul
Oct
Jan
Adcon (BCG/McClelland) Adcon Sensor "enviropro sm 050cm"
Adcon (BCG/McClelland) Adcon Sensor "enviropro sm 100cm"
Adcon (BCG/McClelland) Adcon Sensor "precipitation"

Figure 3 – Screenshot of an Example Application

Acknowedgement

This project is jointly funded through *CSIRO* and
the *Australian Government's National Landcare Program*

References

[1] M.P. Stenson, A. Sommer, R. Searle, *Federating and harmonising disparate soil moisture data sources*, Submitted - 13th International Conference of Hydroinformatics (HIC2018) Palermo, Italy, 2018.

[2] J. Yu, B. Leighton, N. Car, S. Seaton, J. Hodge, *The eReefs data brokering layer for hydrological and environmental data*, Journal Of Hydroinformatics, 18 (2015) 152-167.

[3] P. Box, B. Simons, S. Cox, S. Maguire, *A Data Specification Framework for the Foundation Spatial Data Framework*, CSIRO, 2015.

[4] S. Cox, J. Yu, *The Data Provider Node Ontology*, CSIRO, 2015.

[5] B. Domenico, J. Caron, E. Davis, R. Kambic, S. Nativi, *Thematic Real-time Environmental Distributed Data Services (THREDDS): Incorporating Interactive Analysis Tools into NSDL*, Journal Of Digital Information, 2 (2006).

[6] U. Voges, F. Houbie, N. Lesage, M. Vautier, *OGC I15 (ISO19115 Metadata) Extension Package of CS-WebRIM Profile 10*, Standard Implementation Specification, 2014.

[7] P. Cornillon, J. Gallagher, T. Sgouros, *OPeNDAP: Accessing data in a distributed, heterogeneous environment*, Data Science Journal, 2 (2003) 164-174.

[8] Open Geospatial Consortium Inc., *OpenGIS Web Map Service version 130*, Specification Document, Open Geospatial Consortium (OGC), 2006.

[9] M. Coombe, P. Neumeyer, J. Pasanen, C. Peters, C. Sharman, P. Taylor, *Senaps: A platform for integrating Time-Series with Modelling Systems.*, 22th International Congress on Modelling and Simulation, Hobart, Australia, 2017

[10] Birchip Cropping Group, *DataFarmer*, Company webpage, online at https://www.datafarmer.com.au/, accessed 2018-01-31.

[11] Queensland Department of Science, Information Technology and Innovation, *SILO*, Government department website, online at https://www.longpaddock.qld.gov.au/silo/, accessed 2018-01-30.

[12] Western Australia Department Of Agriculture And Food, *DAFWA Weatherstation API*, Governement department website, online at https://www.agric.wa.gov.au/weather-api-10, accessed 2018-01-30.

[13] S. Cox, Open Geospatial Consortium Inc. *Observations and measurements*, OGC Abstract standard specification, OGC, 2013

[14] J. Tomkins, D. Lowe, *Timeseries Profile of Observations and Measurements*, OGC Standard Implementation Specification, OGC, 2016

[15] S. Cox, *Observations and Measurements - XML Implementation*, OGC Standard Implementation Specification, OGC, 2011

[16] S.J.D. Cox, *An explicit OWL representation of ISO/OGC Observations and Measurements*, Proceedings of the 6th International Workshop on Semantic Sensor Networks, 2013

[17] Portele, C., Cox, S. J. D., Daisey, P., Lake, R., & Whiteside, A., OpenGIS® Geography Markup Language (GML) Encoding Standard, OGC Implementation Specification. OGC 07-036, 2007

[18] S.J.D Cox, P. Taylor, *OGC Observations and Measurements - JSON Implementation*, OGC Standard Implementation Specification, OGC, 2015

[19] A. Bröring, C. Stasch, K. Echterhoff, *Sensor Observation Service Interface Standard*, OGC Standard Interface Specification, OGC, 2012

[20] 52° North Initiative for Geospatial Open Source Software GmbH, *52 North*, Company website, online at http://52north.org, accessed 2018-02-01.

[21] Liang, Steve H.L., Chih-Yuan Huang, and Tania Khalafbeigi. *OGC SensorThings API Part I:Sensing,* OGC, 2016