Paper

The Integration, Analysis and Visualization of Sensor Data from Dispersed Wireless Sensor Network Systems Using the SWE Framework

Yong Jin Lee[1], Jarrod Trevathan[2], Ian Atkinson[1], and Wayne Read[3]

[1] *eResearch Centre, James Cook University, Townsville, Australia*
[2] *School of Information and Communication Technology Griffith University, Brisbane, Australia*
[3] *School of Engineering and Physical Sciences, James Cook University, Townsville, Australia*

Abstract—Wireless Sensor Networks (WSNs) have been used in numerous applications to remotely gather real-time data on important environmental parameters. There are several projects where WSNs are deployed in different locations and operate independently. Each deployment has its own models, encodings, and services for sensor data, and are integrated with different types of visualization/analysis tools based on individual project requirements. This makes it difficult to reuse these services for other WSN applications. A user/system is impeded by having to learn the models, encodings, and services of each system, and also must integrate/interoperate data from different data sources. Sensor Web Enablement (SWE) provides a set of standards (web service interfaces and data encoding/model specifications) to make sensor data publicly available on the web. This paper describes how the SWE framework can be extended to integrate disparate WSN systems and to support standardized access to sensor data. The proposed system also introduces a web-based data visualization and statistical analysis service for data stored in the Sensor Observation Service (SOS) by integrating open source technologies. A performance analysis is presented to show that the additional features have minimal impact on the system. Also some lessons learned through implementing SWE are discussed.

Keywords—environmental data, environmental monitoring, sensor technologies, standardization, web-based visualization.

1. Introduction

In recent decades Wireless Sensor Networks (WSNs) have been dramatically advanced and adopted by many domains to remotely monitor environments [1]. The development of low-cost sensor technologies that are capable of capturing various properties of physical phenomena has led to the growing popularity of WSNs. This has made it easier to observe many environmental aspects [2]. WSNs can also reduce the time needed for collecting large amounts of data on key environmental factors. Furthermore, WSNs provide access to the collected data via the Internet, thereby allowing environmental scientists and decision makers to gain a better real-time understanding about the observed environment.

Analyzing sensed data requires a significant amount of time and effort. Such analysis involves the discovery and integration of data from multiple sources (e.g. various and different types of sensors), assessing quality issues (e.g. missing/suspicious data), hypothesis testing, and visualizing the test results to support decision making. Manual analysis of large amounts of heterogeneous and spatiotemporal data is difficult and complicated. Automatic integration, analysis, and visualization of sensed data from multiple sources can reduce the workload needed for addressing data quality issues and understanding environmental conditions. Such automation can also minimize human mistakes during the analysis phase. However, different WSN systems provide different encodings, models, and services for their sensor data. This makes the integration of differing sensor technologies and network systems problematic. Furthermore, the encodings, models, and services are typically designed for a particular application, which makes it difficult to reuse these services for other WSN applications. A standardized model, encoding, and service for WSN data would avoid the constant and inefficient need to “reinvent the wheel”, and can facilitate the discovery and exploitation of sensor data.

This paper describes a system architecture based on the Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) framework [3], [4]. A middleware integration platform has been designed to collect and integrate sensor data from disparate WSN systems, referred to as the James Cook University (JCU) Sensor Federation (JSF). JSF provides a flexible solution for automating the process of transforming sensor data into the corresponding SWE encoding and storing the data in the Sensor Observation Service (SOS) via the web service interface. Furthermore, additional features have been added to the existing SOS web service interface to provide web-based access to the data and statistical analysis tools. Several real world WSN projects of varying scales and complexities have been integrated into one SOS using JSF to demonstrate the system’s versatility. A performance analysis indicates that the additional features have minimal impact on the system.

This paper is organized as follows. Section 2 describes the SWE framework, provides a brief overview of the WSN projects that authors have been involved with, and presents the motivation for the work presented in this paper. Section 3 describes the JSF system architecture, and proposes a middleware integration platform for the automation of transforming data from multiple WSNs into the SWE encodings. Section 3 also shows how SOS is integrated with open source freely available technologies to support web-based data visualization and statistical analysis of the data stored in the SOS. Section 4 analyses the performance of the enhanced SOS and discusses some of the issues authors had when implementing a SWE system. Section 5 provides some concluding remarks and avenues for future work.

2. Related Work and Problem Motivation

2.1. Sensor Web Enablement

Historically, WSN applications have been completely proprietary. A specific vendor would provide all of the sensor technologies, hardware, software, and network infrastructure. This predicament meant that WSNs were very technical, application-specific, inflexible, and expensive to purchase and maintain. There was limited scope to integrate heterogeneous sensor technologies (i.e. sensors from different vendors). Furthermore, the sensed data was formatted/encoded according to the vendor's own standards, which restricted data sharing and reuse.

In recent years, the concept of the *Sensor Web* has gained momentum [3]–[10]. The Sensor Web's aim is to make all sensors interoperable (regardless of the vendor) so that heterogeneous sensor technologies can be combined to create low-cost, non-proprietary WSNs. Furthermore, collected data becomes available to the Sensor Web which promotes data sharing and reuse. The data can be reused by other consumers for purposes that may be unrelated to, or extend upon the original motivation for collecting the data. This is possible as the WSNs and the data they collect adhere to a set of mutually accepted standards.

The OGC is made up of representatives from academia, industry, and enthusiasts to develop the standards behind the Sensor Web. The OGC SWE framework provides a set of standards that enables all types of sensors, transducers and sensor data repositories to be discoverable, accessible and usable via the Web [3], [4]. The SWE framework consists of following standards and services:

- Observations and Measurements (O&M) – defines XML schemas for accessing and exchanging observations, measurements, procedures, and metadata of sensor systems;
- Sensor Model Language (SensorML) – defines standard models and XML schemas for describing the processes within sensor and observation processing systems. SensorML provides a functional model of the sensor system, where all components including sensors, transducers, actuators, and processors are modeled as processes;
- Sensor Observation Service (SOS) – enables the querying of observations, sensor metadata and representations of observed features, registration/deletion of sensors, and inserting new observations of a registered sensor. SOS is essentially a data repository at the heart of an SWE WSN;
- Sensor Planning Service (SPS) – defines interfaces for queries that provide information about the capabilities of a sensor and how to task the sensor;
- Sensor Alert Service (SAS) – provides a standard web service interface for publishing and subscribing to alerts from sensors; and
- Web Notification Services (WNS) – provides a standard web service interface for asynchronous delivery of messages or alerts from SAS and SPS web services.

The SWE architecture has reached broad acceptance by sensor network application developers. Schade *et al.* [11] applied the SWE framework to volunteered geographic information sensing and event detection techniques. Shafi *et al.* [12] introduced an automated detection/alert system based on the SWE framework (SOS, SAS and WNS) that detects radiation leakage and sends a notification to its subscribed users. Hu *et al.* [13] extended the SensorML model to support sensor observation capability information, i.e. depth, quality, frequency, and range, that enables the accurate discovery of qualified sensors. Srimathi *et al.* [2] proposed a sensor grid architecture that combines a metamodeling tool, the SWE framework, and sensor grid (Hadoop framework). Back *et al.* [14] presented a conceptual design for bridging two domains: a supervisory control and data acquisition system and a *Geographic Information System* (GIS), where the SOS is used to provide a standardized service model for GIS.

Churcher *et al.* [15], [16] describe their experiences with applying SWE to a telecare application involving a number of projects using bespoke sensor hardware, interfaces, and communications. Guru [17] show how they are using the a river catchment WSN to evaluate specifications for SWE in terms of its ability to facilitate water resource management tools. Markovic *et al.* [8] also describe a system for river pollution monitoring and alerts using architecture based on SWE. Lee and Reichardt [18] discuss how open standards for sensor interfaces and data formats can aid in speeding up the identification of threats to homeland security. Samadzadegan *et al.* [19] developed a system architecture for monitoring air quality observations using SWE standards (i.e., SOS, SAS, SPS and WNS) for integrating/interoperating heterogeneous sensors and discovering air pollution to send a notification.

2.2. Proposed Sensor Network Projects

The authors have been involved in several projects where WSNs were deployed in different locations and operate independently. The WSN projects differ in size, complexity, and application. These WSN projects include:

Smart Environmental Analysis and Technologies (SEMAT) [20]: The SEMAT project revolved around constructing smart sensor networks that can be deployed in aquatic settings for the purposes of conducting marine studies. Authors have undertaken SEMAT deployments at Deception Bay and Heron Island in Queensland Australia. The system was designed to take a heterogeneous, low-cost approach, which allowed for near real-time access to data. In each deployment, five buoys containing on-board electronics (Gumstix Computer-On-Module) equipped with various sensors from Dataflow Systems (temperature, light, water pressure, and salinity) were positioned in shallow water environments. The buoys communicated sensor data back to the end user via a base station located near-by on land. This project grappled with significant WSN issues including limited power supply, communications over and underwater, and problems with marine fouling and water ingress.

Digital Homestead Project: This project involved building a low-cost and smart WSN suitable for applications in a digital homestead (i.e. remote farming properties) and urban environments. The project's initial WSN deployment at Rowes Bay in Queensland Australia used and Seeeduino Stalker with eight DS18B20 temperature sensors, DHT22 humidity sensors, and analogue light sensors placed under different types of roofing materials for observing the energy efficiency measures. This study is being used to explore renewable energy solutions that can benefit biodiversity maintenance through planned urban landscapes.

Greening Federation Place: This project's goal was to demonstrate how heritage buildings in tropical environments can evolve into sustainable buildings while retaining cultural significance. Federation Place is a heritage listed building located in Townsville Australia. A WSN containing DS18B20 temperature sensors was deployed at Federation Place to examine the thermal properties of the building and identify fine scale sources of temperature variation.

Over time, difficulties arose as a result of each of the individual WSN deployments using different types of sensors (with different capabilities) and requiring different setup configurations. From a software perspective, each of the projects contained its own models, encodings, and services for sensor data. Also, differing amounts and types of data were available to describe each deployment's characteristics, e.g. the positioning of nodes and sensors are available for the SEMAT deployments, but not for the Rowes Bay deployment. Furthermore, each deployment was initially integrated with different types of visualization and analysis tools based on individual project requirements [21].

As the number of projects grew and their complexity increased, the need for standardization of sensor configuration, data, storage, communication, and a generic web-based user interface for data visualization and analysis became apparent. The solution required was more comprehensive than the existing solutions proposed by the literature in Subsection 2.1 due to a number of factors. The existing proposals from the literature were either for a specific project, or proposed frameworks that were too broad to be applied in practice. A system that could be used over multiple disparate WSN projects with completely different applications was desired. The system also needed to remove the manual process of generating documents that adhere to SWE standards (i.e., SensorML, O&M). When performed manually, this process is tedious, repetitious of work conducted in other WSNs, and is often error-prone. Automating this process would increase the speed of setting up a WSN and reduce the possibility of errors in the SWE documents. Furthermore, to authors' knowledge little or no literature exists on providing a general web-based interface and statistical analysis features that can interact with the SWE framework.

3. JSF System Architecture

In order to integrate and interoperate sensor data from the disparate WSNs described in Subsection 2.2, the SWE framework was extended by creating a *middleware integration platform*. The intention is to provide an interface (referred to as the SWE API) for each WSN deployment that facilitates interoperability according to SWE standards. The SWE API provides common encoding/decoding functions that can be used by any WSN. Encoding functions specific to a particular WSN are abstracted from the SWE layer and are implemented in an extension level. For example, with the SEMAT Heron Island deployment, the HeronSOSEncoder implements functions specific to the Heron Island deployment, by extending the SWE SOSEncoder. The point of this approach is that changes to any particular WSN do not affect how any other WSN application interacts with the SWE framework. Furthermore, the system automates the generation of the SWE documentation (i.e., SensorML, O&M) to ease WSN set-up time or changes in the WSN configuration, thereby reducing the potential for errors in adhering to SWE standards.

The authors also decided to extend SWE's SOS standard by providing a generic web-based user interface and statistical analysis functionality. The combination of the WSN projects and our extended SWE functionality is referred to as the JCU Sensor Federation (JSF). Figure 1 presents the proposed JSF system architecture. The system is comprised of:

- A Data User – this is the individual user/stakeholder who is interested in accessing and viewing the data from any WSN connected to the system;
- A SOS with support for web-based data visualization and statistical analysis – this provides storage of sensor metadata and sensor observations. The extended

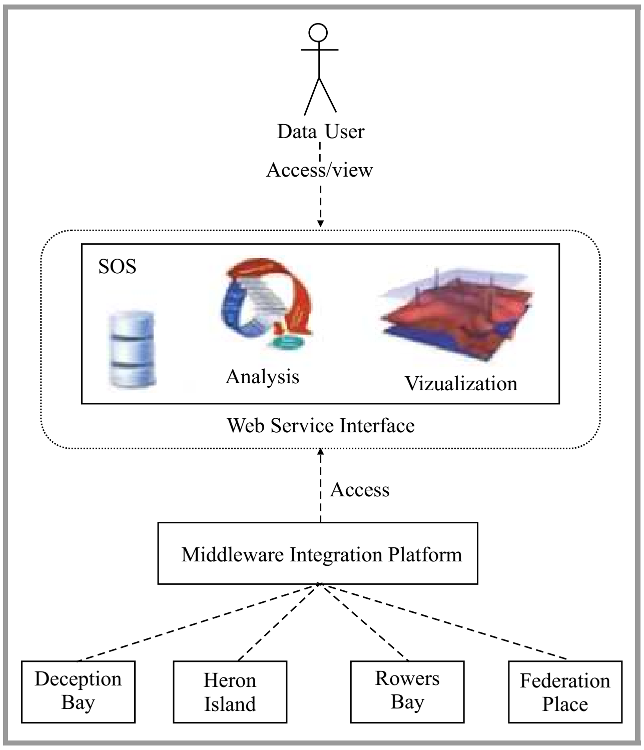

Fig. 1. The JSF system architecture for the integration and interoperability of sensor data from multiple data sources.

functionality allows for automated input, integration, visualization, and analysis of sensed data. Environmental scientists can use these features to enhance the decision making process and/or the discovery of new information from the environment under observation;

- A Middleware Integration Platform – this hides the heterogeneity of models, encodings, and web service interfaces for sensor data. This enables a user to access the sensor data via the Internet without having to learn individual models and service interfaces, and also facilitates the integration and interoperability of the heterogeneous data through automating the process of generating SWE-compliant documents;

- Individual WSN Projects – the projects undertaken as part of the work presented in this paper. These include the SEMAT deployments at Deception Bay and Heron Island, the Digital Homestead deployment at Rowes Bay, and the Greening Federation Place deployment (refer to Subsection 2.2). Conceivably the number of projects can scale with the system.

A major benefit of employing the SWE framework is for easy discovery and use of the data. The decision was made to not utilize different SOSs for each deployment as a distributed approach reduces this benefit. For example, the distributed approach would require a user to send requests to different SOS URLs. Furthermore, users would need to be informed every time a new SOS is added to the system. Having a single SOS provides a single service interface to multiple WSN deployments' data/metadata. Therefore, the proposed architecture brings all of the WSN deployments together in one SOS.

3.1. The JSF Middleware Integration Platform

Sensor data stored in a SOS must be available via the SOS web service interface. However, this process requires significant effort in practice to achieve. For example, registering a sensor via the SOS web service interface requires three steps:

- the sensor data need to be mapped into the respective message encoding (i.e., the InsertSensorDocument),

- the document needs to be formatted based on a SOS protocol binding, e.g., Simple Object Access Protocol (SOAP),

- the formatted document is then transferred to the SOS via its web service interface.

Furthermore, it is necessary to have a common agreement on how to apply SWE within a specific domain. This is because there are different SWE specifications available, where the encodings, models, and services are different. For example, SOS version 2.0 provides the ability to store observation metadata and data through different transactions, whereas this ability is not available in SOS version 1.0. That is, SOS version 1.0 requires metadata to be transferred every time an observation is to be stored. However, SOS version 2.0 only requires the metadata to be stored once. Therefore, only the observation data is required, which reduces the amount of data transferred and makes transfers faster. Also, the SOS implementations can vary based on the needs of a particular domain (e.g. protocol bindings). Due to the aforementioned reasons, and that SOS version 2.0 has a richer array of functionality, for JSF the SOS version 2.0 is used.

JSF was developed to integrate the sensor data from multiple WSNs using the SWE architecture. JSF manages adapted SWE specifications of an individual domain application to map, format, and store its sensor and sensed data. JSF provides the ability to extract sensor data from a domain WSN application, and transforms the data into the corresponding request document using its SWE API implementation. Then, the encoded document is formatted and transferred using the Transaction API, which provides binding protocols (i.e. XML binding or SOAP binding) and the HTTP functions (request/response). This abstracts the underlying WSN projects from SWE and also automates the process of generating SWE-compliant documents.

Figure 2 shows the design of the JSF middleware integration platform. This platform consists of three layers:

- Extraction layer – extracts sensor network data from web services or databases and converts the data into a corresponding SWE API function,

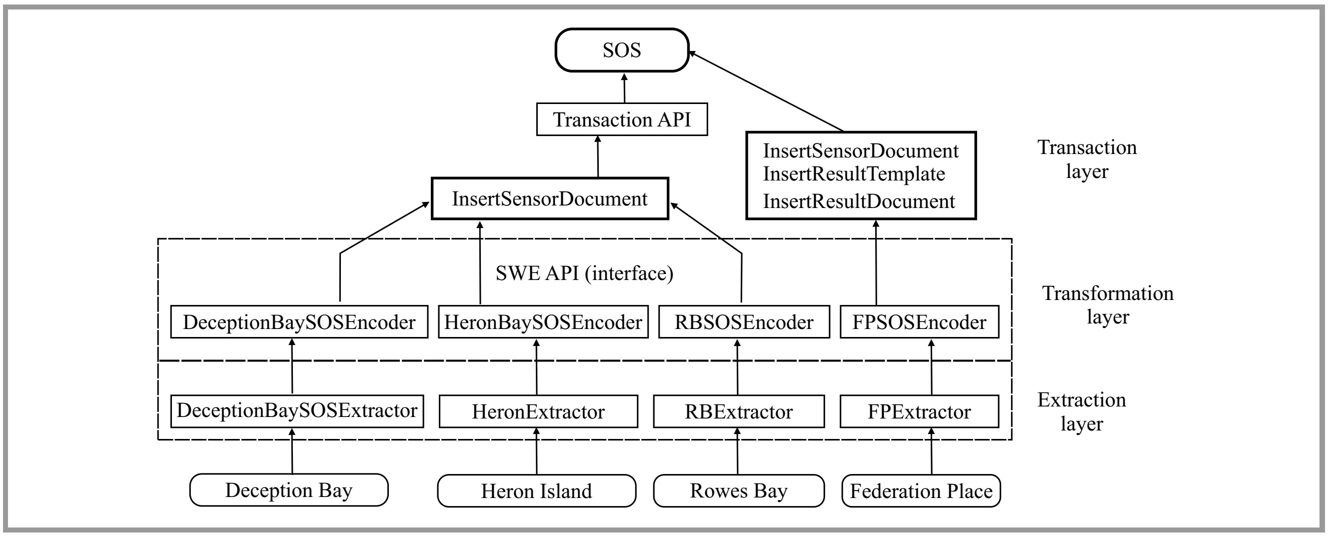

Fig. 2. JSF middleware integration platform design (the arrows denote the direction of the flow of information).

- Transformation layer (SWE API) – maps data into a corresponding SOS request document (i.e., InsertSensorDocument, InsertObservationDocument, InsertResultTemplateDocument, or InsertResult),
- Transaction layer – binds/transfers the encoded documents to a corresponding SOS web service interface.

Consider the SEMAT Deception Bay deployment. The data is first extracted by the DeceptionBayExtractor (i.e. the Extraction layer) and appropriate SWE API function is called. The data is then encoded into a SOS-compliant document by the DeceptionBaySOSEncoder (i.e. the Transformation layer). Finally, the SOS request document is transferred and stored in the SOS according to the SOS web service (i.e. the Transaction layer). A similar process occurs for data from any of the other WSN deployments, i.e. Heron Island, Rowes Bay, and Federation Place.

3.1.1. The SWE API

The SWE Application Programming Interface (API) provides standardized and portable system abstractions that allows JSF to transform data into SWE encodings (i.e. the Transformation layer). The SWE API consists of the following components:

- Interface layer – provides five interfaces for the SOS API. Each interface corresponds to the individual SWE standards (SWEFrame_SOSEncoder, SWEFrame_SMLEncoder, SWEFrame_OMOLEncoder, SWEFrame_SASEncoder, and SWEFrame_SPSEncoder);
- Abstract layer – is associated with a particular SWE implementation that provides a list of commonly used functionality for the particular SWE implementation, i.e. encoding data, query request documents, and binding and transmission operations;
- Implementation layer – is an extension of the Abstract layer that maps sensor network data from a particular system type into the SWE documents.

A deployment must first be registered with JSF. During registration, an API is created for the deployment. A deployment's API implementation contains two properties "identifier" and "version". The combined value must be unique in order to store it into the API container. The registered implementation can be retrieved from the container by passing the respective identifier and version onto the container interface. The Abstract layer can simply be extended to add a new encoding, binding, or transmission process.

Figure 3 presents an example of an SWE API implementation for the data from the SEMAT Heron Island deployment. It shows the identifier, version, and methods for each class, and relationships between classes. The HeronSOSEncoder class uses all of the other classes. Individual classes represent each corresponding SWE component, e.g. HeronSOSEncoder for SOS. These classes transform sensor data into their respective SWE encodings (e.g., the getInsertSensorDocument functionality maps sensor data into the SOS InsertSensorDocument).

The HeronSensorMLEncoder implements the AbstractSensorMLEncoder that maps the sensor data into an InsertSensorDocument. The HeronOMLEncoder implements the AbstractOMLEncoder that maps the observation data into an InsertObservationDocument, where its observation metadata and data can be mapped in separate documents by the HeronResultTemplateEncoder and HeronResultEncoder respectively. The HeronSosEncoder provides a single interface to access to the aforementioned implementations.

To describe a sensor in a SWE-compliant way requires the following attributes:

- Identification – this requires the user to supply uniqueID ("urn:ogc:def:identifier:OGC:uniqueID") and offeringID (urn:ogc;def:identifier:OGC:offeringID), where an uniqueID attribute is used for

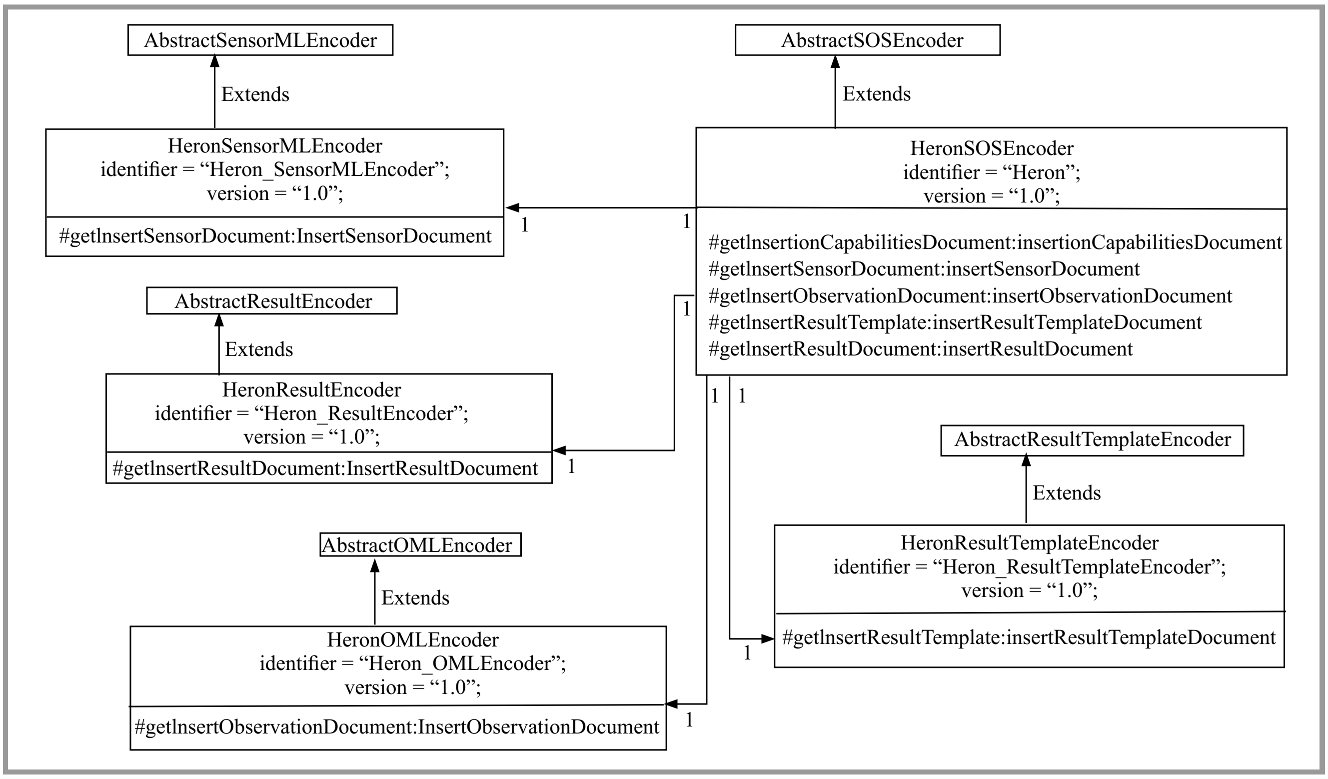

Fig. 3. Class diagram of the SWE API implementation for the SEMAT Heron Island deployment.

Table 1
The integration of the heterogeneous sensor data models/encodings using the SWE encodings

WSN	Sensor Data Model/Encoding	SWE Encoding
Deception Bay, Heron Island and Rowes Bay	**Node model:** name, latitude, longitude, buoy model, power log and sensors	**SensorML:** identifier, position, outputs and components
	Sensor model: serial number, manufacturer, description, type, parameter number and unit of measurement	**SensorML:** identifier, position and outputs
	SensedData model: type, unit of measurement, position, time, raw_Data, calibrated_Data and power log	**O&M:** field and values
Federation Place	**Node model:** name, latitude, longitude and sensors	**SensorML:** identifier, position and components
	Sensor model: name, type, observedProperty, code, altitude and definition	**SensorML:** identifier, position, outputs and observableProperty
	Observation model: definition, code and type	ResultTemplate: field
	Data model: value and timestamp	**Result:** value

querying sensor (system) metadata, and offeringID is used for inserting the sensor's sensed data. This is the same value that must be used within O&M in order to link the observation and sensor metadata/data;

- Capabilities– this attribute is used to describe the feature that the sensor is measuring (e.g. Ocean);
- Location – describes the sensor's (system) geographical location, its format is defined by its "reference-frame" definition;
- Inputs – describes the sensor's (system) process input;
- Output – describes the sensor's (system) process output;
- Components – other systems that is included within the system.

A WSN deployment typically consists of sensor nodes, where sensors are attached/installed. So proposed WSNs are described as a sensor node (identification, capabilities, location, inputs, outputs, and components) that combines

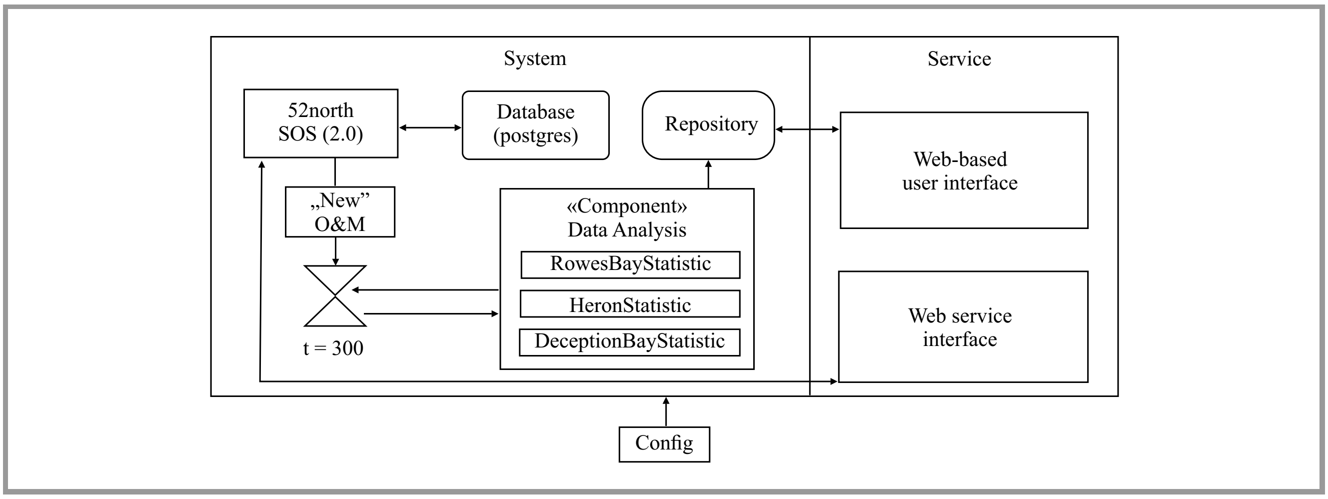

Fig. 4. The enhanced SOS architecture with web-based data visualization and statistical analysis functionality.

all the sensors under the components attributes, where individual sensors are described (identification, capabilities, location, inputs, outputs). The SEMAT Heron Island deployment is described under the sensor node's capabilities attribute and individual sensor's capabilities describe what actual environmental factor the sensor is measuring, e.g. temperature, light. Note that SensorML documents can be quite large. The InsertSensor document for the Heron Island deployment contains over 800 lines of code.

Table 1 shows how the SWE API integrates the heterogeneous sensor data models/encodings using the SWE encodings. The SWE API implementation for the Deception Bay, Heron Island and Rowes Bay deployments converts the Node and Sensor model into SensorML, and the SensedData model into O&M. The API implementation for the Federation Place deployment transforms the Node and Sensor model into SensorML, and the Observation and Data model into InsertResultTemplate and InsertResult. The purpose of this table is to show how each deployments own characteristics can be maintained, while the SWE API maps these characteristics to the appropriate corresponding attributes in SWE.

3.2. SOS Design and Implementation

As previously mentioned, the authors required a generic web-based user interface to operate across all the WSN deployments using JSF. Therefore, an existing SOS implementation (52° North SOS version 2.0) is extended to support web-based data visualization and statistical analysis of the collected data. This was achieved using the following open source technologies and APIs:

- Apache Common Math API – provides mathematics and statistics, i.e. descriptive statistics, simple/multiple regression, rank transformation, covariance, correlation, and statistical tests;
- Weka API – enables Java to support several data mining tasks including data pre-processing, clustering, classification, regression, and feature selection;
- Highstock library – provides general timeline charts with navigation options, e.g. scrolling and date picker;
- Google Maps API – allows for the embedding of Google Maps on a web page.

The system provides additional features to the existing SOS web service interfaces (i.e., InsertObservation, InsertResultTemplate and InsertResult) that analyses and visualizes the data based on the system configuration. The SWE encodings/documents are provided by the OGC (net.opengis.* package). 52° North SOS does not provide support for all the semantic definitions to describe attributes such as FeatureOfInterest, Location, Format, etc. These attributes are described using the 52° North API (ogc.n52.sos.ogc.om.features*).

Figure 4 illustrates the enhanced SOS architecture and how the system operates. At the heart of the architecture is the 52° North SOS. SWE compliant documents are pushed to, or retrieved from the SOS via SWE's web service interface. When a sensor transfers sensed data, the system stores the data in a temporary location. This sensed data is formatted according to the O&M SWE standard.

The system runs a batch-process at predefined intervals to iterate through new data to generate and store a JavaScript Object Notation (JSON) file for each O&M document. The interval is determined by a configuration file. For example, in this instance the configuration file contains $t = 300$ which means that batch process runs every 300 s.

Individual statistical analysis implementations can integrate data from multiple sensors by providing a list of sensor identifiers supplied by the offering parameter. Note that the offering parameter is O&M's equivalent of the sensor identifier in sensorML. This parameter can be accessed by invoking the isOfferingListed function in O&M. The system also checks for any statistical analysis implementations that use the sensed data through the offering property within the O&M document during the iteration. The result of each

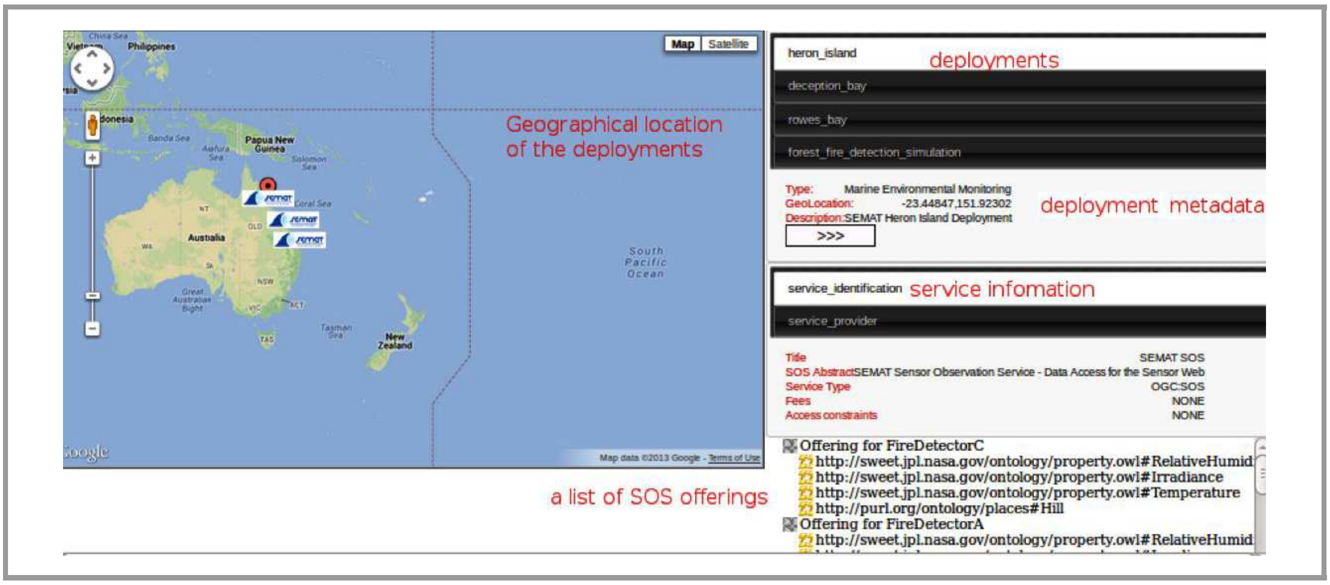

Fig. 5. The main web page shows the geographical location of the WSN deployments on the Google Maps, and sensor/observation data provided by the SOS.

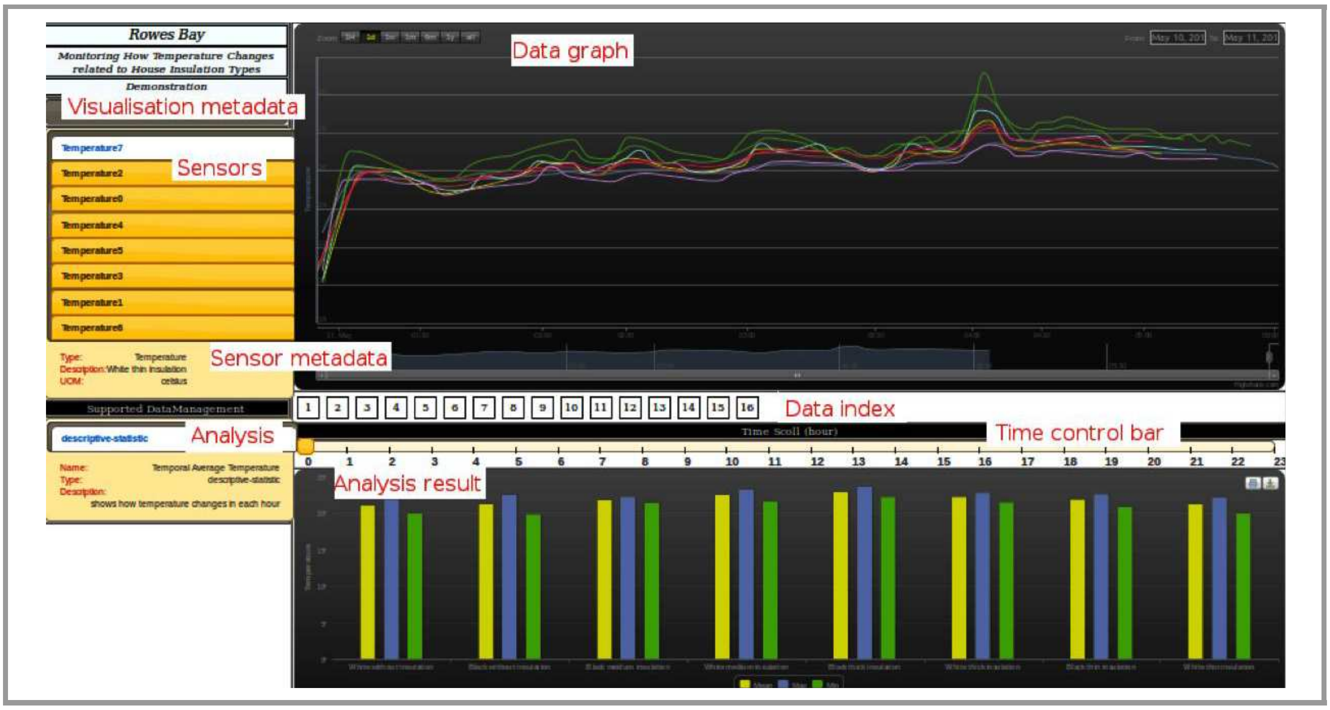

Fig. 6. An example deployment page for the web-based user interface.

analysis process is mapped with the Java Map interface, which is transformed into a JSON file and stored in the repository. The JSON files provide a light-weight data interchange format to facilitate asynchronous browser/server communication. This technology is used for mapping deployment data via the Java Map interface.

The web-based user interface provides graphical visualization of the sensors, observation data, and the analysis results. When the user interface requires information, it can access the data from the 52° North SOS using SWE's web service interface. Alternately, when statistical analysis/aggregation data is required, the user interface can access the repository.

The user interface consists of two primary sections:

- Main web page – provides an overview of registered deployments, presents the deployment locations on a Google Map, and allows the user to select and navigate to a specific deployments web page,
- Deployment web page – provides access to all details specific to an individual deployment, and presents

the user with options to graphically visualize sensed data and analysis results, or to export the data to a file.

Figure 5 shows the user interface's main web page. The main page of the interface consists of four frames:

- Map – illustrates the geographical location of various WSN deployments and sensor nodes as markers on Google Maps. A user can view a deployment's parametric information by clicking on the corresponding marker in the map;
- Deployment – displays a list of WSN deployments associated with a particular project (e.g. SEMAT) and shows each deployment's name, geographical location, type, and description. A user can also view information specific to a WSN deployment by clicking the corresponding deployment button;
- Service – provides the SWE service information;
- SOS offering – shows a list of the sensed data provided by the SOS.

Figure 6 shows an example of what is shown in the user interface when a specific deployment has been clicked on. This page consists of the following components:

- Visualization metadata – provides a brief description about the deployment's purpose;
- Sensors – shows a list of sensors associated with the deployment and allows a user to click on a specific sensor to view its metadata and sensed data;
- Sensor metadata – displays sensor metadata information (e.g., the sensor type, description);
- Analysis – lists any analysis processes being conducted on the collected data, and provides the user with several options regarding the types of statistical analysis that can be performed;
- Analysis result – provides graphical illustration of the analysis results;
- Data graph – provides graphical illustration of the sensed data. Note that numerous sensor data sets can be overlayed on the same graph;
- Time control bar – allows the user to control the analysis result by varying the time span.

Figure 6 shows the deployment page for the Rowes Bay WSN. The interface is showing the data graphs corresponding to the temperature sensors and the sensor metadata, i.e. where each sensor is located in relation to the roofing material. The user can graph each sensor's data individually, or overlay all sensor data. The deployment page also illustrates the statistical analysis results conducted on the temperature data. The analysis section is charting the minimum, maximum, variance, and average temperature of each temperature sensor at hourly intervals. An user has the ability to export the collected (and analyzed) data in a series of formats for use in other software packages.

The web-based user interface is written in JavaScript and HTML to increase flexibility and reusability. The interface also provides the ability to add, modify, or remove a data source (i.e. sensor and analysis results), and to configure the visualization type (i.e. charting type) from the deployment web page using a system configuration file. A new deployment web page can be added by creating a web page and modifying the configuration file (i.e. setting its data sources and visualization types). This is automatically converted into the corresponding JavaScript functions to generate charts/graphs and Google Maps. In this manner, the web-based interface essentially becomes "generic" in that it can display data from any WSN regardless of its application.

4. Performance and Lessons Learned

4.1. Performance of the Enhanced SOS System

To evaluate the integration of the statistical analysis and visualization services with the SOS, the 52° North SOS is compared with presented enhanced SOS. The test environment was setup with a virtual machine and a laptop. Both SOS implementations were installed on the virtual machine (Linux kernel 2.6.32, Red Hat 4.4.7, 2.93 GHz CPU and 1 GB RAM) with Java 1.7 and Tomcat 7. JSF was installed on the laptop (Mac OS X 10.7.5 2.8 GHz CPU, 4 GB RAM) to simulate test data sets (InsertObservationDocument).

The effects of the additional features in the system performance were observed. 100 simulated sensors were registered on the both SOS implementations, and the enhanced SOS was configured to provide descriptive statistics on the sensed data transferred by these sensors every minute. A test function was implemented in the middleware to encode and transfer the InsertObservationDocument with 100 data points for each sensor.

Figure 7 presents the results of the performance comparison between the 52° North SOS and the enhanced SOS. The X-axis and Y-axis gives the number of sensors and the time (in seconds) respectively. The test results show that the InsertObservation, InsertResultTemplate and InsertResult service time of the enhanced SOS are approximately 0.5, 0.1, and 0.3 seconds slower respectively than the original SOS for every transaction. This indicates that the additional features of proposed enhanced SOS have minimal impact on the original service performance of the 52° North SOS.

4.2. Lessons Learned from Implementing SWE

In this section some of the practical challenges the authors faced when using SWE are briefly described and how they managed to overcome these issues.

A major hurdle for implementing a SWE-compliant WSN is that SWE takes time to learn (i.e. a few months). The SWE documentation is complex. An SWE implementer

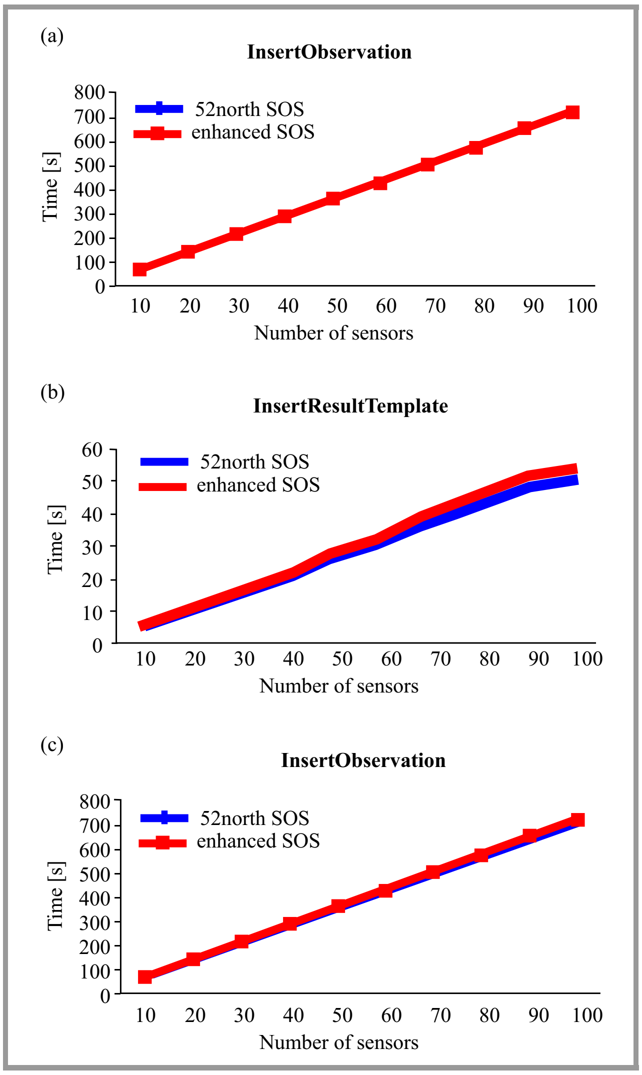

Fig. 7. A performance comparison between the 52° North SOS and the enhanced SOS.

has to go through every part of the documentation or a least the relevant sections in order to successfully get a SWE-compliant WSN operational.

The process with moving a WSN towards SWE is found to first focus on getting a SOS running. The next step is understanding how to push a SensorML document into the SOS using the sensor's identifier (and also usually a semantic web resource). Once a sensor is registered with SOS, the sensor's identifier is used to push a sensor observation into the SOS. SOS will then use the sensor's identifier to automatically link observation data with the corresponding SensorML document.

A major issue that hinders an SWE implementation is that SWE is very pedantic with its expectations about data and message formats. SWE requires specific data to be provided in a particular order with an explicit structure, and also with an exact semantic resource. If any data or message does not strictly adhere to these formatting rules, the information is not recognized and causes errors. This lack of flexibility can make implementing SWE a challenge. In some instances SWE will raise the error to the user's attention. On other occasions it may appear that data has been stored correctly (as no errors are flagged). However, as the format is incorrect, the data is unable to be retrieved. This lack of storage safeguard can cause frustration if a significant quantity of data has been incorrectly stored and can no longer be recovered. Therefore, extensive testing of formatting is required before employing the system for use with real data.

A further frustration with SWE is that SOS asks you to describe a large number of characteristics. None of these characteristics are optional. If you do not have this information, you need to put in dummy values as placeholders. Furthermore, SOS version 2.0 has a semantic web component that requires you to define what the metadata is for (semantic web resource). This may not be relevant in the context of your system, or if you do not desire to use the semantic web components. Additionally, it can be difficult to locate some of these semantic resources.

JSF attempts to alleviate some of these issues as the SWE API transforms the sensor data context of a deployment into the SWE encoding (SensorML, O&M, etc.). As this process is fully automated, it makes it easier to generate the encodings. All you have to do is pass the parameters (system name, process name, type of process) and the SWE API creates the respective SWE-compliant document. This ensures that data is properly formatted, data is not missing (or dummy values are created), and semantic resources are in place. This reduces the amount of work required and the number of potential errors when operating the system.

5. Conclusion

The authors have been involved with multiple real-world WSN projects including the SEMAT deployments at Deception Bay and Heron Island, the Digital Homestead deployment at Rowes Bay, and the Federation Place WSN. Each project differed in size, complexity, and application. Different sensors were involved (i.e., temperature, humidity, light, water pressure, and salinity) which provided different data streams. Furthermore, each project used different technologies to collect, log, transfer, and store the data. A system has been required to speed up the deployment process by automating common tasks, and a way to integrate, visualize, and analyze the data under a common web-based user interface to support the decision making processes of the end user.

This paper presented a system architecture based on the SWE framework that facilitates the integration and interoperability of sensor data from dispersed WSN systems. The authors proposed a middleware integration platform JSF, which provides integration between the WSN systems and the SWE framework. JSF manages adapted SWE specifications of an individual domain application to map, format, and store its sensor and sensed data, and provides the ability to extract sensor data from a domain WSN application. JSF transforms the data into the corresponding request document using its SWE API implementation, formats the data (via a protocol binding), and then transfers the requested document to the SOS. JSF facilitates the discovery and exploitation of sensor data from dispersed WSNs, and reduces the amount of effort needed for developing a new WSN application. As presented solution provides a standardized framework, JSF can be reused for any type of WSN application.

The paper also presented an enhanced SOS implementation for JSF that provides support for web-based data visualization and statistical analysis using open source freely available technologies. Additional features were added to the existing SOS web service interfaces that integrates data from multiple sensors. The enhanced SOS provides functionality for analyzing the data and allows the data and the analysis result to be visualized graphically via a web-based user interface. A performance analysis was conducted to compare the impact that the additional functionality of the enhanced SOS has on the system compared to a regular SOS implementation. The performance analysis showed that the additional features have minimal impact on the system performance, where each additional transaction with 100 sensed data points increased its service response time by approximately 0.5 s. Also some lessons learned for implementing a WSN systems using SWE are briefly described. Some impediments/difficulties for integrating SWE are the initial learning curve, having to strictly adhere to SWE document formats, having to supply extraneous or unnecessary information, dealing with subtle errors in SOS due to incorrect message formats or missing data, and the tedious process of generating lengthy SWE-compliant documents.

Future work involves extending the SOS server with a semantically-enabled SOS server [22]. A significant issue with the SWE architecture is the lack of semantically rich discovery mechanisms. This makes it hard to explore related concepts, subgroups of sensor types, or other dependencies between the sensors and the data they collect. Integrating SOS with semantic technologies will enable the SOS server to query high-level knowledge of the environment as well as the raw sensor data. This can facilitate knowledge sharing and exchange, and automated processing of web resources.

References

[1] F. L. Lewis, "Wireless sensor networks", in *Smart Environments: Technology, Protocols and Applications*, D. Cook and S. Das, Eds. New York: Wiley, 2004, pp. 11–46.

[2] C. Srimathi, C. Eunmi, H. A. Jemal, and N. Rajesh, "Sensor grid middleware metamodeling and analysis", *Int. J. Distrib. Sensor Netw.*, pp. 1–12, 2014.

[3] M. Botts, G. Percivall, C. Reed, and J. Davidson, "OGC Sensor Web Enablement: Overview and High Level Architecture", in *GeoSensor Networks – 2nd Int. Conf. GSN 2006, Boston, MA, USA*, S. Nittel, A. Labrinidis, and A. Stefanidis, Eds. *LNCS*, vol. 4540, pp. 175–190. Springer, 2008.

[4] M. Botts and A. Robin, "Bringing the sensor web together", *Geosciences*, pp. 46–53, 2007.

[5] K. Delin and S. Jackson, "The sensor web: a new instrument concept", in *SPIE's Symposium on Integrated Optics*, San Jose, CA, USA, 2001.

[6] K. Delin, "Sensor webs in the wild", in *Wireless Sensor Networks: A Systems Perspective*, N. Bulusu and S. Jha, Eds. London: Artech House, 2005.

[7] P. Gibbons, "Irisnet: An architecture for a worldwide sensor web", *IEEE Pervasive Comput.*, vol. 2, no. 4, pp. 22–33, 2003.

[8] N. Markovic, A. Stanimirovic, and L. Stoimenov, "Sensor web for river water pollution monitoring and alert system", in *Proc. 12th AGILE Int. Conf. on Geographic Inform. Sci. AGILE 2009*, Hannover, Germany, 2009.

[9] D. Moodley and S. Ingo, "A new architecture for the sensor web: the SWAP framework", in *Semantic Sensor Netw. Worksh. A workshop of the 5th Int. Semantic Web Conf. ISWC 2006*, Athens, Georgia, USA, 2006.

[10] E. Torres-Martinez, P. Granville, M. Schoeberl, and M. Kalb, "A web of sensors: enabling the earth science vision", *Acta Astronautica*, vol. 53, no. 4–10, pp. 423–428, 2003.

[11] S. Schade *et al.*, "Citizen-based sensing of crisis events: sensor web enablement for volunteered geographic information", *Appl. Geomat.*, vol. 5, no. 1, pp. 3–18, 2013.

[12] S. Shafi, A. A. Reshi and A. Kumaravel, "Wireless sensor network based early warning and alert system for radioactive radiation leakage", *Middle-East J. Scient. Res.*, vol. 19, no. 12, pp. 1602–1608, 2014.

[13] S. M. Guru, P. Taylor, H. Neuhaus, Y. Shu, D. Smith, and A. Terhorst, "Hydrological sensor web for the South Esk catchment in the Tasmanian state of Australia", in *Proc. 4th Int. Conf. on eScience, eScience'08*, Indianapolis, IN, USA, 2008, pp. 432–433.

[14] C. Hu, Q. Guan, N. Chen, J. Li, X. Zhong, and Y. Han, "An observation capability metadata model for EO sensor discovery in sensor web enablement environments", *Remote Sens*, vol. 6, no. 11, pp. 10546–10570, 2014.

[15] S. Back, S. B. Kranzer, T. J. Heistracher, and T. J. Lampoltshammer, "Bridging SCADA systems and GI systems", in *Proc. IEEE World Forum on Internet of Things WF-IoT 2014*, Seoul, Korea (South), 2014, pp. 41–44.

[16] G. E. Churcher, G. Bilchev, J. Foley, R. Gedge, and T. Mizutani, "Experiences applying sensor web enablement to a practical telecare application", in *Proc. 3rd Int. Symp. on Wirel. Pervasive Comput. ISWPC 2008*, Santorini, Greece, 2008, pp. 138–142.

[17] G. E. Churcher and J. Foley, "Applying and extending sensor web enablement to a telecare sensor network architecture", in *Proc. 4th International ICST Conf. on COMmun. Syst. softWAre and middlewaRE*, Dublin, Ireland, 2009, p. 6, 2009.

[18] K. B. Lee and M. Reichardt, "Open standards for homeland security sensor networks", *Instrument. & Measur. Mag.*, vol. 8, no. 5, pp. 14–21, 2005.

[19] F. Samadzadegan, H. Zahmatkesh, M. Saber, and H. J. Ghazi khanlou, "An interoperable architecture for air pollution early warning system based on sensor web", *ISPRS – Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 1, no. 3, pp. 459–462, 2013.

[20] J. Trevathan, R. Johnstone, T. Chiffings, I. Atkinson, N. Bergmann, W. Read, S. Theiss, and T. Stevens, "SEMAT – The next generation of inexpensive marine environmental monitoring and measurement systems", *Sensors*, vol. 12, no. 7, pp. 9711–9748, 2012.

[21] Y. J. Lee, J. Trevathan, I. Atkinson, W. Read, and R. Johnstone, "The evolution of the SEMAT sensor network management system", in *Proc. 7th Int. Conf. on Intell. Sensors, Sensor Netw. and Inform. Process. ISSNIP 2011*, Adelaide, Australia, 2011, pp. 229–234.

[22] C. A. Henson, J. K. Pschorr, A. P. Sheth, and K. Thirunarayan, "SemSOS: Semantic Sensor Observation Service", in *Int. Symp. on Collabor. Technol. and Syst.*, Baltimore, MA, USA, 2009.

Yong Jin Lee completed his Ph.D. in Information Technology in 2015 at James Cook University. His research interests include wireless sensor networks, datamining, ecommerce security and fraud, and big data. He worked on the Smart Environmental Monitoring and Analysis Technologies project in conjunction with the University of Queensland. Dr. Lee now works in industry in South Korea as an analyst/programmer. His current projects involves big data analysis of sensor data.

E-mail: yong.lee1@jcu.edu.au
eResearch Centre
James Cook University
Townsville, Queensland, Australia

Jarrod Trevathan is a lecturer/researcher for the School of Information and Communication Technology at Griffith University. His research interests include ecommerce security and fraud, dynamic handwritten signature verification, phishing scams, and wireless sensor networks. He has worked in industry as an analyst/programmer and has an extensive history working with James Cook University and the University of Queensland. He took a lead role in the Smart Environmental Monitoring and Analysis Technologies project. Dr. Trevathan has published over 70 papers in conferences, journals and book chapters.

E-mail: j.trevathan@griffith.edu.au
School of Information and Communication Technology
Griffith University
Nathan, Queensland, Australia

Ian Atkinson is a Professor and the director of the eResearch Centre at James Cook University. His research interests include environmental monitoring, sensors, data management and supramolecular chemistry. He has worked in various professional/corporate IT roles and is the recipient of several major research grants, including the Smart Environmental Monitoring and Analysis Technologies and CSIRO Digital Homestead projects.

E-mail: ian.atkinson@jcu.edu.au
eResearch Centre
James Cook University
Townsville, Queensland, Australia

Wayne Read is an applied mathematician Associate Professor. He is a lecturer/researcher for the School of Engineering and Physical Sciences at James Cook University. His research interests include numerical and computational mathematics, mathematical modeling (particularly for ground water), and ecommerce security and fraud. He has worked in various roles at James Cook University, including Head of School for over five years. He has a long standing relationship with IT/Computer Science researchers in mentoring and providing applied mathematical solutions for algorithms and data structures.

E-mail: wayne.read@jcu.edu.au
School of Engineering and Physical Sciences
James Cook University
Townsville, Queensland, Australia