# Open Sensor Platforms: The Sensor Web Enablement Framework and Beyond

Alexander Funk, Claas Busemann, Christian Kuka, Susanne Boll, Daniela Nicklas

OFFIS - Institute for Information Technology
Oldenburg, Germany
busemann|kuka@offis.de
http://www.offis.de

Carl von Ossietzky Universität
Oldenburg, Germany
alexander.funk|susanne.boll|daniela.nicklas@uni-oldenburg.de

**Abstract:** Over the last years, more and more sensor systems emerged that can not only be accessed but also controlled and configured using the Internet. This trend allows mobile applications that rely on sensor data; however, the access and configuration of sensor systems is a tedious task. As many sensor manufacturers implement their own protocols, there are many communication protocols. Although unifying standards like the Sensor Web Enablement framework (SWE)—developed by the Open Geospatial Consortium (OGC)— exists, they are still not widely used by off-the-shelf sensor systems. As a result, other middleware platforms like the Sensor Configuration and Aggregation Middleware for Multi Platform Interchange (SCAMPI) have been developed. In this paper, we analyze the Sensor Web Enablement (SWE) framework regarding requirements of sensor-based applications, show how it can be integrated in SCAMPI, and highlight the strengths and limitations of SWE based on the experiences we made during the integration process. The implementation and evaluation of the system shows the benefits and disadvantages of the SWE framework from the perspective of a developer who wants to integrate the SWE framework into a middleware as well as from the perspective of a developer who wants to build a system around the SWE framework. The results of this paper can be used to assess whether the SWE standard and its application will be useful for a given scenario.

## 1 Introduction

Many mobile applications rely on sensor data. Some need it to adapt their services to their users current situation (so-called context-aware applications [SAW$^+$94]), e.g., to change the presentation, adapt the presented information, or take actions. Examples are a navigation application that changes the display from a high-level view to a zoomed-in detailed view based on speed, or a tourist information application that displays points of interest depending on the users location. Other mobile applications even use sensor data as their primary application data, to show the current status of real-world objects, e.g., the location of moving objects (like trucks in a logistics scenario), weather information, or the status of other players in a mixed-reality game.

The integration of live sensor data into applications adds complex tasks to the software development, like the communication with the sensor or the transformation of raw sensor signals to meaningful information. For mobile applications that rely on stationary sensor systems, these tasks have to be done multiple times, since due to the inherent mobility of the application, the relevant sensors change when the user is moving.

Today, many low cost and easy to install sensors are available, many more in the future. Often, these sensors can already be accessed over the Internet as they come along with TCP/IP enabled Ethernet or WLAN interfaces. Technologies like sensor middlewares or simple wrapper applications even allow more sensors to be accessed through the Internet. This kind of access is very important for upcoming technologies like the Internet of Things, which demands a high number of sensors that can be accessed through the web. But also companies and even private persons are more and more interested in having simple web based access to sensors. The high number of existing sensors and sensor middlewares leads to a high number of communication protocols. The differences between these proprietary protocols make it hard for developers to implement systems that communicate with sensors from different manufacturers. The OGC tries to solve the problem by defining the Sensor Web Enablement (SWE) framework, which defines a unified communication standard for sensors and sensor services. However, it seems like the SWE framework is not widely used yet, as the number of implementations is pretty low. The existing implementations also seem to be build around the standard and do not integrate it into an existing system.

In this paper we analyze the benefits and disadvantages of the SWE framework by integrating it into the SCAMPI middleware. Thereby not only the compatible features and not implementable functions are considered but also things like the communication overhead. This work can be used by developers to assess whether the integration of the SWE standard into their system is feasible. The results of the evaluation could also be seen as suggestions to improve the SWE standard. The rest of the paper is organized as follows: The related work can be found in Section 2. Section 3 examines the characteristics of open sensor platforms while Section 4 gives an overview of the SWE framework. Section 5 describes our middleware. Finally, the evaluation can be found in Section 6 and the conclusion in Section 7.

## 2 Related Work

The Sensor Web Enablement (SWE) [Con08] is a proposal from the Open Geospatial Consortium (OGC) for a Protocol that describes Sensors and Sensor Observations. It is designed to unify the communication between sensors and sensor services. A detailed description of the standard can be found in Section 4.

To achieve a unified usage and exchange of sensor data between different components and middlewares several implementations of the SWE standard have been proposed. The 52 °North Sensor Web community [NOR] focuses on the development of a broad range of services and encoding implementations related to the SWE framework. The Space Time Toolkit [Spa] of the University of Alabama uses parts of the SWE framework for 4D visualization via an interactive user interface. The GeoCENS [GEO] project at York University uses SWE to visualize geospatial data in the web. The NICTA Open Sensor Web Architecture (NOSA) [KBLK07] has implemented five core specifications of the SWE including the Sensor Model Language (SensorML), the Observation and Measurement (O&M), the Sensor Collection Service (SCS), the Sensor Planning Service (SPS) and the Web Notification Service (WNS). Furthermore the SWE was implemented in the EO-1 SWE [EO1] testing project by the NASA to link together ground and space-based instruments to enable autonomous collaborative observation collections. Other groups target on special parts of the SWE like the Sensor Observation Data Registration [CZC07] to improve the quality of a sensor data registry. However, all of these implementations seem to be build especially for the SWE framework. None of them integrated the standard into an already existing platform. They also do not specify which functions or applications can be realized when integrating the SWE framework. However, existing sensor systems would benefit from an integration of the standard as they thereby would achieve a higher level compatibility to other systems.

A service oriented approach for sensor discovery, access and usage is used by several groups. The Microsoft SenseWeb [Nat06] Project for example allows different users to share their sensor measurements over the Internet. The Sensor.Network [GPU10] developed by Sun Microsystems can be used for sensor data aggregation and visualization. The GSN [ZAST08, AHS07, AHS06] is a middleware for processing and discovery of sensor measurements. All of them are using their proprietary standards and representation of sensors, transducers and sensor data repositories. They also use their own functions to discover, access and use sensors and sensor data. SWE is not supported by one of these. The SCAMPI middleware [BKW+09] which is used in this paper to analyze the SWE standard is also a service oriented sensor system that could be compared to systems mentioned above. The middleware is explained in detail in Section 5.

Service oriented architectures are one of the most promising ways of implementing sensor access and sensor discovery. One of the upcoming framework specifications for such middlewares is the OSGi specification. The OSGi Alliance (formerly Open Services Gateway initiative) defines a hardware independent framework for dynamic administration and modularize development. The OSGi Alliance simply defines the interfaces of the platform and gives no advice about the internal structure of the so called OSGi bundles. OSGi is used in automotive, mobile devices and smart grids.

## 3 Characteristics of Open Sensor Platforms

Open sensor platforms, like the ones mentioned in the related work, often share the same characteristics which we consider typical for this kind of technologies. We conducted an extended analysis of these characteristics. This section shortly introduces the results, which are later used to compare different features of SWE and SCAMPI.

**Sensor Administration** The basic application purpose of an open sensor platform is to allow the administration of sensors. They come along with functions that allow the registration of new sensors, but also the request of sensor information, the removal of sensor information and so on. Some platforms even allow the request for sensor information using an abstract description (so called sensor meta data, see below).

**Sensor Description and Access** A sensor description is usually done by specifying a name or ID and sensor type. The sensor type includes a description of the transmitted data. Static sensors have a fixed position, while mobile sensors transmit their position with every measurement. The description of a sensor often also includes its platform, for example a car which carries multiple sensors. In addition, some systems allow the description of processing chains inside of the sensor which allows applications to understand the quality of the transmitted data.

**Sensor Metadata** Many platforms allow a description using abstract sensor data, like keywords, contact persons, sensor categories and so on. This data can be used by other applications or persons to make proper decisions when using the sensor data. It is also often helpful, if the metadata is available over the Internet for everyone, as it reveals the sensors purpose to the user.

**Processing of Sensor Data** Open sensor platforms are also often used to process sensor data. This can be a simple processing like filtering data using a threshold or a complex system that allows the manipulation, aggregation and filtering of data streams. These system are also often not only able to process but also to provide the processed sensor data to the user or an application. This processed sensor data can then be accessed as if it is coming from a real sensor.

**Sensor Observations** There seem to be two common principles when accessing and proving sensor data. One principle follows the sensor and allows the application to access its data. The interpretation of the measurement has to be done by the application. The other principle follows the asset that may be monitored by several sensors. The application simply specifies an asset in which it is interested and the platform delivers the measurements that may be collected by various sensors.

**Communication Protocol** Open sensor platforms usually also provide an open communication protocol. Even if the basic functions of these protocols are often the same, the representation usually differs notably. XML seems to be a very common technology for modern communication protocols. However, there are also simple text based or binary representations.

# 4 Sensor Web Enablement Framework

The Sensor Web Enablement (SWE) [Con08] is a proposal from the Open Geospatial Consortium (OGC) for a Protocol that describes Sensors and Sensor Observations. It also describes several (Web-)Services to access those data structures. The vision is to simplify the connection and cooperation between different sensor networks, especially using World Wide Web. The SWE therefore is for sensor networks what HTTP is for the internet. The ultimate goal is to connect heterogeneous sensors and sensor networks among each other and allow a unified access to those sensors. This could, in the long term, lead to lower costs and better quality sensor communications because there is only one protocol to focus on instead of several proprietary ones, as the interoperability between different proprietary protocols can be a serious problem. The broad adoption of the SWE is obviously required for this to work.

## 4.1 Sensor Web Enablement Approach

As the key to a broad adoption is flexibility the SWE framework optimally should be usable in all use cases that involve sensor communication. For this reason SWE is not an implementation but a specification. It does not specify a codebase, instead it defines the protocols and web services that have to be used. The implementation is not prescribed by the SWE-Standard. This means that the defined web services can be built in any language on any platform. The only constraint is to be conform to the SWE XML schemas which are defined by the standard. As shown in Fig. 1 the SWE framework basically consists of three XML schemas and four (XML) web service descriptions [Con08].

When dealing with sensor data the SWE framework specifies *FeatureOfInterests*. These are Objects in the real world that are somehow related to a location like a specific geographic area, a lake or a river. FeatureOfInterests have *ObservablePropertys* that can be observed by sensors, like the water temperature of a lake. Sensors can monitor these observable properties and measure an estimate of its value as part of an ***Observation***. An observation is a snapshot of a FeatureOfInterest's properties. ObservablePropertys that have an actual estimate of their value inside an observation are called ***ObservedProperty***.

## 4.2 XML schemas of Sensor Web Enablement

As mentioned before the architecture of the SWE framework basically consists of three XML schemas and four web services [Con08]. These are explained in detail in the following sections.

**Sensor Model Language** The Sensor Model Language [Con07b] is used to describe sensors and sensor platforms. It uses XML to encode all sensor information and is specified using an XML-Schema. Defined attributes include metadata like contact person, key-

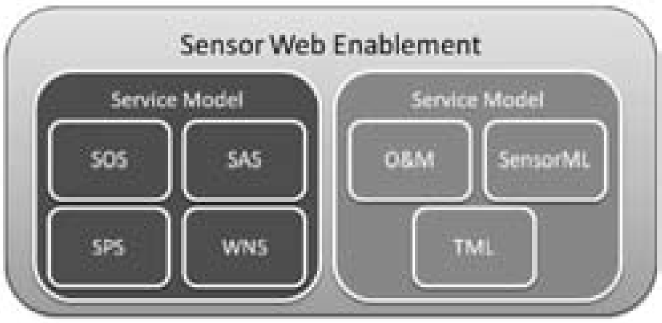

Figure 1: Basic Architecture of the SWE (see [Con08])

words for easy sensor discovery, location of the sensor or an identification. SensorML can also describe the data stream of a sensor and the corresponding data types or units of measurement. Sensors are often part of a sensor platform, for example a satellite. In that case the satellite is the sensor platform and consists of several other sensors. It is possible to describe such relations inside a SensorML document and specify more information about the type of relationship, for example the exact location of a sensor relative to its sensor platform. SensorML is also able to store processes and linked processes, so-called process chains. The idea behind this concept is to save all data involved in the process of measuring a property. Processes can describe the whole process from measuring a low-level electric impulse, converting this impulse to a number and finally sending this measurement from the sensor to a destination. This means the whole process is transparent to every client and can be used to tune historic sensor observations if for example a sensor is known to use a certain formula for its translation steps.

**Observation & Meauserement** Observation & Measurement (O&M) [Con07a] is a model language that is used to describe observations from sensors. Like SensorML, O&M is specified using an XML schema. It covers data like the time of the observation, the observed properties and the feature of interest. It also contains the actual data stream, an estimate of every observed property and an identifier for the sensor that made the observation.

**Transducer Model Language** The Transducer Model Language is an equivalent to the Sensor Model Language. It is also specified as an XML schema and describes transducers using XML. Its purpose is to describe transducers and streaming sensors. Therefore, it provides a more adequate XML structure.

### 4.3 Web Services of Sensor Web Enablement

**Sensor Observation Service** The Sensor Observation Service (SOS) is probably the most important web service in the SWE framework. The SOS provides access to sensors, current sensor observations and historic sensor observations. Clients can also search for sensors using metadata, for example search for all sensors that are observing a specific FeatureOfInterest. Clients receive observations represented as O&M documents and sensor descriptions represented as SensorML document. The registration of new sensors is also possible using the SOS.

**Sensor Planning Service** The Sensor Planning Service (SPS) is responsible for tasking sensors on the basis of client queries. Clients can send their tasks to a SPS which then tries to fulfill them. This means that the SPS tries to reconfigure all necessary sensors to accomplish the task. When finished, the SPS can use the Web Notification Service to inform the client.

**Sensor Alert Service** The Sensor Alert Service (SAS) is used to define and observe alerts. An alert is an expression that defines values for properties of a sensor. The SAS monitors the necessary sensors and sends an alert to all registered clients if an expression applies. Alerts can be defined as public and clients can register to any public alert. Thereby this mechanism is usually more effective than having every client to poll the sensor data by their own.

**Web Notification Service** The Web Notification Service (WNS) is a utility service for asynchronous delivery of data to a client. Basically the WNS is used by any other service to asynchronously send a message to a client that registered at the WNS earlier and therefore has a unique id to be identified. The client itself can specify its desired method of asynchronous communication. The supported methods depend on the actual implementation and are therefore not specified by the SWE framework.

## 5 SCAMPI Middleware

SCAMPI [BKW$^+$09] is a sensor data middleware that lies between the sensors which transmit their measurements and the application that is about to use them. It is able to handle a large amount of sensors and users which can be administrated using simple control mechanisms. The design goal was to develop a sensor middleware that allows providers to realize and host efficient low-cost sensor applications for end-users. Fig. 2 shows a schematic view of the middleware.

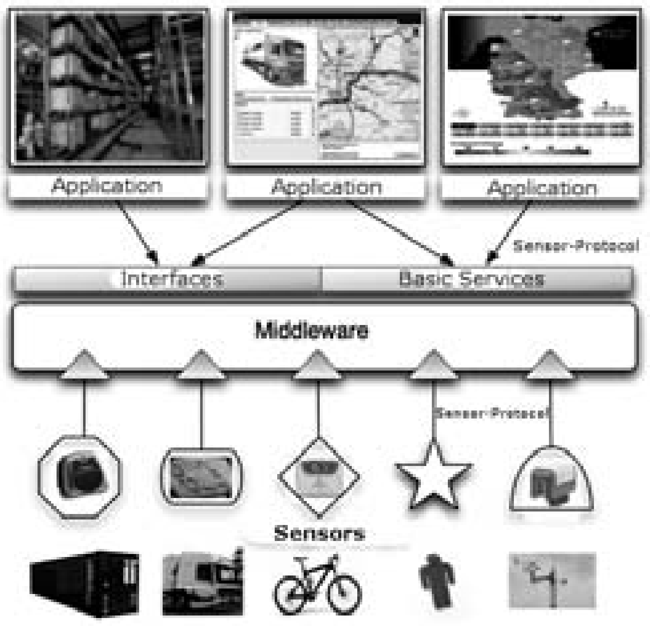

Figure 2: Schematic Representation of our Middleware

## 5.1 SCAMPI Approach

The main feature of the middleware is letting applications have a unified access to the sensor data. As the middleware comes along with a sensor adapter service that allows the integration of various communication protocol drivers almost every kind of sensor can send its data to the middleware. Through the unified access to this data using web services this data can easily be accessed by other applications. The middleware comes along with its own communication protocol (SCAI) which can not only be used to transmit sensor data, but also to administrate the middleware. The main representation of this protocol is realized using XML but there is also a short text version for sources with low resources. It can for example be used to register new sensors or add supplementary metadata to their description.

The system consists of different web services that allow the graphical administration of the middleware but also the user friendly visualization of sensor data. Another important feature is the processing unit, which can be used to process the incoming sensor data before it is made available to the application. Thereby the processing can be quite complex, like for example joining different sensor data streams or aggregating data. The processed sensor data can be accessed by an application in the same way in which it would access a real sensor.

### 5.2 Why not Sensor Web Enablement?

During the design process of the middleware we had to decide whether to build the system based on the SWE standards or not. We finally decided not to base our middleware on the SWE approach as there are several reasons that speak against it. The main reason is that does not support the description of complex processing operations within a middleware. Also the communication overhead that comes along with the XML descriptions of the SWE protocols seemed to be too high for sources with low resources. The SWE framework does describe services that allow the configuration of sensors but it does not specify a protocol for this. Finally, SWE seems to follow an asset based approach when handling observations, in that observations themselves are 'first class objects' that can be addressed, stored, and retrieved individually. SCAMPI on the other side is a sensor based system. However, based on these reasons we initially decided not to build our middleware based on the SWE standards. As the SWE standard more and more evolved and nowadays is use in several projects we decided to integrate the SWE services into our systems as those standards are important to archive compatibility between different systems.

## 6 Comparison of Sensor Web Enablement and SCAMPI Features

In this section, we compare the features of SWE and SCAMPI based on the characteristic in Section 3. The column "effort" in the following tables refers to the effort necessary to implement a specific feature in SCAMPI.

**Sensor Administration** Sensor administration and the configuration of the sensors are supported in different levels on SWE and SCAMPI. SWE provides the Sensor Planning Service for such a use case but does not define a XML protocol to configure a sensor. The SCAI protocol however already has the ability to describe a configuration for a sensor. This means SCAMPI already offers a concept for configuration including a protocol, whereas SWE does not define a protocol but only the responsible web service. Another

| Feature | SCAMPI | SWE | effort |
|---:|:---:|:---:|:---:|
| sensor support required | no | yes | - |
| protocol can configurate sensors | yes | no | - |

Table 1: Sensor Administration Features

advantage of SCAMPI is the ability to use sensors even if those sensors do not provide specific support for SCAMPI. Developers can define so called "sensor adapters" inside the SCAMPI middleware that act as a wrapper between the sensor and SCAMPI. Through this concept every sensor can be used, assumed the sensor adapter is developed once. The SWE on the other side needs explicit support by the sensors because it does not provide any comparable concept. A comparison of these features can also be found in Tab. 1.

**Sensor Description and Access** SCAMPI and SWE both have their own languages for the sensor description and in both cases it is XML, SCAI and SensorML respectively. They both offer a web service for the unified access to sensors too. A closer look to SCAI and SensorML reveals the difference between both and shows that SensorML has many more place holders for the data of a sensor description. In SWE a SensorML document can define whole sensor platforms which consist of several different sensors and can additionally describe the relation between those sensors. Also the SensorML can directly encode the actual position from a sensor in a predefined placeholder. SCAMPI does not provide

| Feature | SCAMPI | SWE | effort |
|---:|:---:|:---:|:---:|
| unique id for sensors | yes | yes (SML) | - |
| sensors are separate entity | yes | yes | - |
| placeholder for actual position in protocol | no | yes (SML) | low |
| actual position through sensors data stream | yes | yes (SML) | - |
| describe sensor platforms | no | yes | high |
| define datatypes for datastream elements | yes | yes | - |
| sensor description through web service | yes | yes (SOS) | - |

Table 2: Sensor Description and Access Features

the ability to define sensor platforms and has no predefined place holder for the sensor position. In SCAMPI the sensor position can be transmitted but has to be transmitted as part of the sensor data stream. The concepts of sensor description and sensor access are largely compatible. Making SCAMPI compatible with SWE on this end should not be a major problem. A comparison of these features can also be found in Tab. 2.

**Sensor Metadata** Sensor metadata provided by a sensor platform does help the client to search for the sensors he actually requires. In case of SCAMPI there is no metadata

| Feature | SCAMPI | SWE | effort |
|---:|:---:|:---:|:---:|
| keywords, descriptions for sensors, contact person(s) | no | yes (SML) | low |
| change history from sensor description | no | yes (SML) | medium |
| references to sensor documentation | no | yes (SML) | low |
| date of expire for sensor description | no | yes (SML) | medium |
| UoM for every data stream element | no | yes (SML) | low |
| Classification using categories | yes | no | medium |

Table 3: Sensor Metadata Features

that could be attached to a sensor description as it does not provide a placeholder for it. The SWE is different, it has placeholders, most of them optional, that can be populated by metadata and then can be used to filter the sensors or to increase discoverability of sensors. This metadata can be keywords, description, contact person(s), references to sensor documentation, change history from the SensorML document, a unit of measurement (UoM) for every data stream element or a date of expiration for the specific SensorML document it is embedded into. However, SCAMPI does allow the classification of sensors into categories which is also a metadata and helps users to identify the correct sensors. Sensor metadata normally does not affect the general compatibility between SWE and SCAMPI because most metadata in SWE is optional and could be discarded. Of course this would neutralize the enhancements through metadata described earlier. Implementing that missing metadata into SCAI should also be possible without major problems. A comparison of these features can also be found in Tab. 3.

**Processing of Sensor Data** The processing of sensor data, especially between sensor data of different sensors, can open up many new possibilities. Through processing of sensor data it is possible to use several sensors and define a specific procedure to generate more meaningful data. The SWE does not provide any web service for sensor data aggregation or processing of sensor data. Nevertheless the SensorML does provide the concept of processes and process chains which can be used to describe such processing steps involved on sensor basis. These data structures can be used to describe sensor data processing steps on a sensor. The SWE on its own does not provide processing on middleware or web service basis. SCAMPI on the other side provides a processing unit for sensor

| Feature | SCAMPI | SWE |
|---:|:-:|:-:|
| virtual sensors | yes | no |
| notify on defined situation | yes | yes (SAS) |
| sensor data processing on middleware basis | yes | no |
| sensor data processing on sensor basis possible | yes | yes |

Table 4: Processing of Sensor Data Features

data processing and does also provide the concept of virtual sensors. A virtual sensor is made up from different processing steps that can be defined by a client. These processing steps can use sensor data from different sensors, join them, filter them or check for specific values to form a new virtual sensor. This virtual sensor then has its own data stream which can be accessed by a client. Since the SWE is a specification rather then an implementation it is possible to implement a web service that is SWE compatible but also provides the processing of sensor data on a middleware basis. It is imaginable that the virtual sensors in SCAMPI can be described as SensorML documents that describe through processes and process chains all of their processing steps. In that case the concept of processes in SWE and the concept of virtual sensors in SCAMPI would play nicely together. A comparison of these features can also be found in Tab. 4.

**Sensor Observations** Sensor observations are referring to the data an actual sensor measured. The SWE and SCAMPI have very different principles in interacting with those observations. The SWE treats a sensor observation as an entity for itself. A sensor observation contains a unique id and some meta data like the observed feature of interest or the observed properties. This enables the possibility to search for specific sensor observations by their metadata. For example a client could request a sensor observation that has a specific feature of interest as target. In SCAMPI there is no sensor observation entity.

| Feature | SCAMPI | SWE | effort |
|---:|:---:|:---:|:---:|
| unique id for sensor observations | no | yes (OM) | high |
| define feature of interest | no | yes (OM) | high |
| define observed properties | no | yes (OM) | high |
| sensor observations are separate entity | no | yes | high |
| get observation by metadata | no | yes (SOS) | high |

Table 5: Sensor Observations Features

The only entity inside SCAMPI is a sensor. Metadata like the feature of interest or the observed properties can not be saved. One could say the SCAMPI approach is very sensor centric whereas the SWE approach shifts the interest more in the direction of the sensor observations. This distinction in principles concerning the sensor observations is a very problematic point when it comes to compatibility between both systems. There is no way to mimic all possibilities of sensor observations from SWE in SCAMPI and therefore it is not possible to achieve a reasonable mapping between the protocols. A comparison of these features can also be found in Tab. 5.

**Communication Protocol** The communication protocol is used between the sensors and the web service and between client and web service. Optimally it has only a low overhead to reduce the amount of data that is necessary to be transmitted. The SWE defines, among others, the SensorML and O&M. These protocols are very verbose and can be called human readable. The downside to this is the size of those protocols. SensorML and O&M tend to have a relative high overhead compared to their payload. This could lead to problems when a sensor does not have the capabilities to handle or parse large XML documents. SCAMPI on the other hand supports several protocols. It supports SCAI, a human

| Feature | SCAMPI | SWE |
|---:|:---:|:---:|
| several protocols | yes | no |
| short-text-protocol for low end sensors | yes | no |

Table 6: Communication Protocol Features

readable XML protocol, but there is also a short-text protocol available that reduces the character count and so reduces the overhead of the protocol. This short-text protocol is not as easy readable as the XML version, at least the meaning will not be clear to anybody at the first glance. However this allows SCAMPI to even support sensors which are low on hardware resources. A comparison of these features can also be found in Tab. 6.

## 7 Conclusion and Future Work

This paper describes how to integrate the SWE framework into an existing sensor middleware. The evaluation identifies the SWE features that can or can not be adapted and also the features of the middleware which could not or only partially be realized using the SWE framework. These results can be used by sensor middleware developers who have to decide whether to use the SWE framework or not. In the future we are going to implement more features of the SWE standard to achieve a higher level of compatibility with the standard. However, these implementations will be realized in a way that will not move the focus of the middleware completely to the one of the SWE approach, as this would change the basic purpose of the middleware.

## Acknowledgment

The authors would like to thank Stefan Behrensen for his superior support for this paper.

## References

[AHS06] K Aberer, M Hauswirth, and A Salehi. Middleware support for the "Internet of Things". In *5th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"*, Germany, 2006. 5th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze".

[AHS07] Karl Aberer, Manfred Hauswirth, and Ali Salehi. Infrastructure for data processing in large-scale interconnected sensor networks. In *Proceedings of the Mobile Data Management (MDM 2007)*, Mannheim, Germany, 2007. IEEE Computer Society.

[BKW$^+$09] Claas Busemann, Christian Kuka, Utz Westermann, Susanne Boll, and Daniela Nicklas. SCAMPI - Sensor Configuration and Aggregation Middleware for Multi Platform Interchange. In *GI Jahrestagung*, pages 2084–2097, 2009.

[Con07a] Open Geospatial Consortium. *Observations and Measurements - Part 1 - Observation schema*, December 2007.

[Con07b] Open Geospatial Consortium. *OpenGIS Sensor Model Language (SensorML) Implementation Specification*, July 2007.

[Con08] Open Geospatial Consortium. *OGC Sensor Web Enablement Architecture*, August 2008.

[CZC07] Nengcheng Chen, Zhong Zheng, and Zeqiang Chen. An Efficient Sensor Observation Data Registration based on Asynchronous Service Middleware. *Network and Parallel Computing Workshops, IFIP International Conference on*, 0:374–379, 2007.

[EO1] GSFC. Sensor Web / Testbed Initiatives. Website. Available online at http://eo1.gsfc.nasa.gov/new/extended/sensorWeb/sensorWeb.html; visited on 2010-11-01.

[GEO] GeoCENS. Website. Available online at http://sensorweb.geomatics.ucalgary.ca/; visited on 2010-11-02.

[GPU10] V. Gupta, A. Poursohi, and P. Udupi. Sensor.Network: An open data exchange for the web of things. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pages 753 –755, 29 2010.

[KBLK07] T. Kobialka, R. Buyya, C. Leckie, and R. Kotagiri. A Sensor Web Middleware with Stateful Services for Heterogeneous Sensor Networks. In *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, pages 491 –496, 2007.

[Nat06] Suman Nath. Challenges in building a portal for sensors world-wide. In *In First Workshop on WorldSensor-Web, Boulder,CO*, pages 3–4. ACM, 2006.

[NOR] 52 °North. Website. Available online at http://52north.org; visited on 2010-11-01.

[SAW$^+$94] B. N Schilit, N. Adams, R. Want, et al. *Context-aware Computing Applications*. Xerox Corp., Palo Alto Research Center, 1994.

[Spa] Space Time Toolkit. Website. Available online at http://code.google.com/p/space-time-toolkit; visited on 2010-11-01.

[ZAST08] Yongluan Zhou, Karl Aberer, Ali Salehi, and Kian-Lee Tan. Rethinking the Design of Distributed Stream Processing Systems. In *Proceedings of the 24th International Conference on Data Engineering Workshops (NetDB 2008)*, pages 182–187, Cancun, Mexico, 2008. IEEE Computer Society.