The SID Creator: A Visual Approach for Integrating Sensors with the Sensor Web

Arne Bröring[1,2,3], Felix Bache[2,3], Thomas Bartoschek[2], Corné P.J.M. van Elzakker[1]

[1] Faculty ITC, University of Twente, Netherlands
http://www.itc.nl

[2] Institute for Geoinformatics, University of Muenster, Germany
http://www.ifgi.de

[3] 52°North Initiative for Geospatial Open Source Software, Germany
http://www.52north.org

Abstract

This paper describes the *Sensor Interface Descriptor* (SID) model and focuses on presenting and evaluating the *SID creator*, a visual approach to create instances of the SID model. Those SID instances comprise the knowledge required to integrate a sensor with the Sensor Web. This integration is done by an *SID interpreter* which uses an SID instance to translate between a sensor protocol and the Sensor Web protocols. An SID instance, designed for a particular sensor type, can be reused in multiple applications and can be shared among user communities. The SID creator enables users to describe the interface, commands and metadata of their sensors. In a user study, we evaluated the simplification of the sensor integration process through the SID concept. The study incorporated four user groups, ranging from high school students to expert users, who were challenged to integrate weather station sensors with the Sensor Web by utilizing the SID creator. While the common approaches of integrating such sensors with the Sensor Web involve manual coding and extensive adaptation efforts, this new visual approach significantly simplifies the integration process.

1 Introduction

The aim of the Sensor Web is to enable discovery, access, exchange, and processing of sensor data, sensor metadata, and sensor task planning across different applications. The Sensor Web Enablement (SWE) initiative of the Open Geospatial Consortium (OGC) standardizes Web Service interfaces and data encodings which can be utilized to build such a Sensor Web [1]. The interoperable SWE services make sensors available by hiding the sensor communication details and the heterogeneous sensor protocols from applications.

In recent years, the SWE standards have been applied in various projects (e.g. [2, 3]) showing their practicability and suitability in real world scenarios. However, a central challenge still remains to be tackled. Since SWE services are defined from an *application-oriented* and not from a *sensor-oriented* perspective, the integration of sensors and services is not sufficiently defined, yet. In fact, a gap of interoperability between SWE services and sensors has been identified [4].

By looking at two SWE services (Section 2), the Sensor Observation Service (SOS) and Sensor Planning Service (SPS), this interoperability gap becomes clear. The SOS offers operations for registering sensors and uploading their data. Due to limitations in bandwidth and processing power, sensors are usually not able to transform their measured data to the SWE protocols and to upload them to the SOS. The SPS offers operations for an interoperable tasking of sensors. However, it is not defined by the specifications how an SPS server transforms a retrieved sensor task to a command of the sensor protocol.

Today, sensors are integrated with the Sensor Web by manually building proprietary bridges for each pair of SWE service implementation and sensor type. This approach is cumbersome and leads to extensive adaptation efforts - the key cost factor in developing large-scale sensor network systems [5]. Relevant concepts which facilitate the sensor integration have not been realized yet.

Minimizing those sensor integration efforts can significantly support applications such as disaster management where an ad-hoc densification of an existing sensor network is demanded. Examples range from flooding scenarios, in which the affected river courses are not covered densely enough with water gages, to incidents in nuclear plants, which require ad-hoc deployments of radiation detectors. Assuming a Sensor Web is already in place and used by disaster relief organizations as a coherent infrastructure to access sensors, an integration of new sensors in the most efficient way becomes necessary.

This work focuses on presenting and evaluating a visual creator for Sensor Interface Descriptors (SID). This SID creator facilitates the integration of sensors with the Sensor Web by enabling a semi-automatic generation of their interface and protocol descriptions. After describing the SWE initiative and related work (Section 2), an overview of the SID model[1] is presented (Section 3). It enables the declarative description of a sensor's interface. Based on this model, SID interpreters can be built which use the knowledge contained in an instance of the SID model to translate between sensor protocol and Sensor Web protocols. Such interpreters for SID instances can be built independently of particular sensor technology. Hence, the SID model, together with generic SID interpreters, closes the interoperability gap between sensors and the Sensor Web as described above.

However, the manual creation of SID instances is not straightforward. Therefore, the visual SID creator has been developed (Section 4) which supports users in describing the sensor interface to integrate the sensor with the Sensor Web. We evaluated this SID creator and the SID concept by conducting a user study (Section 5). The participants of the user study, nine senior high school students and eleven people with at least a Bachelor of Science degree in Computer Sciences were challenged to integrate the sensors of a weather station with the Sensor Web by utilizing the SID creator. In Section 6, the results of the user study are analyzed. The paper ends with a conclusion and gives an outlook to future work.

2 Background & Related Work

The main Web Services of OGC's SWE framework are the Sensor Observation Service (SOS) and the Sensor Planning Service (SPS). The SOS [6] provides interoperable access to sensor data as well as sensor metadata. To control and task sensors the SPS [7] can be used. A common application of SPS is to define simple sensor parameters such as the sampling rate but also more complex tasks such as mission planning of satellite systems.

Apart from these Web Service specifications, SWE incorporates information models for observed sensor data, the Observations & Measurements (O&M) [8] standard, as well as for the description of sensors, the Sensor Model Language (SensorML) [9].

SensorML specifies a model and encoding for sensor related processes such as measuring or post processing procedures. Physical as well as logi-

[1] A detailed description of the SID model can be found in [18] and [19].

cal sensors are modeled as *processes*. The functional model of a process can be described in detail, including its identification, classification, inputs, outputs, parameters, and characteristics such as a spatial or temporal description. Processes can be composed by process chains.
O&M defines a model and encoding for *observations*. An observation has a result (e.g. 0.7 mSv/a) which is an estimated value of an *observed property* (e.g. radiation), a particular characteristic of a *feature of interest* (e.g. the city of Muenster). The result value is generated by a *procedure*, e.g. a sensor such as a radiation detector described in SensorML. These four central components are linked within SWE.

So, while the connection between the Sensor Web layer, consisting of those SWE components, and the application layer is well-defined, the connection between Sensor Web layer and sensor layer is not yet sufficiently defined. This interoperability gap between the two layers can be addressed from two directions: By following a bottom-up approach the interoperable access on the sensor layer is improved. Top-down approaches introduce mechanisms on the Sensor Web layer to abstract from the variety of sensor protocols (Figure 1).

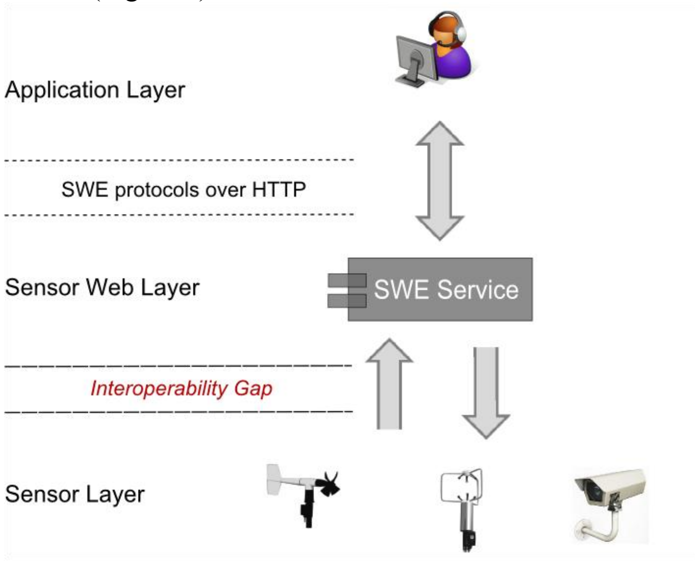


Fig. 1. - The Sensor Web layer stack

The bottom-up direction is addressed by several standardization approaches. Promising is the IEEE 1451 family of standards[2] which is a uni-

[2] http://ieee1451.nist.gov/

versal approach to connect sensors to diverse networks and systems. An important feature of this standards family is the definition of a Transducer Electronic Data Sheet (TEDS) which is a small memory device attached to the transducer describing for example its identification, calibration, correction data, measurement range, and manufacturer related information. However, the expressiveness of TEDS is limited and it cannot capture all metadata of a sensor. For example, higher level processing of sensor data cannot be described in TEDS. This requirement is addressed by SensorML. Therefore, Hu et al. [10] convert TEDS to SensorML by creating a knowledge base which maps each TEDS property to an appropriate SensorML description. It would be promising to extend this approach and to combine it with our work to automatically generate SIDs for IEEE 1451 sensors so that an SID interpreter can connect IEEE 1451 sensors on-the-fly with SWE services.

However, in today's real world applications not only IEEE 1451 but in fact a huge variety of sensor interfaces (standardized or proprietary) are utilized. Hence, different projects are approaching the interoperability gap in a top-down manner, from the upper Sensor Web layer.

The application AnySen [11] is capable of reading and interpreting data from sensor nodes by abstracting the sensor protocols and reading the sensor description from an external file. The authors do not detail but claim that AnySen allows the formatting of these sensor descriptions compliant to the SensorML standard. While AnySen supports the provision of sensor data by connecting to an SOS, other SWE services, in particular tasking of sensors through an SPS, are not supported.

Walter & Nash [4] identify the interoperability gap and analyze different system models which may lower the implementation barrier for coupling sensor systems and SWE services. The authors suggest lightweight SWE connectors which can be adapted to different raw sensor formats to convert them to SWE-based data models. They state that such SWE connectors could be implemented for a wide range of different sensor types. They come up with design approaches, but do not detail them.

The Sensor Abstraction Layer (SAL) [12] is most similar to the SID concept. SAL makes use of SensorML to describe sensor interfaces. As a library, it offers high-level functions to access sensors by hiding their specific technological details. The architecture follows a split design consisting of lightweight SAL agents running on the sensor gateways to handle the communication with the hardware and SAL clients usable by application developers to invoke specific actions on sensors managed by an agent. Missing are mechanisms for the final connection to SWE services and the integration of sensors with the Sensor Web.

None of the approaches above is leveraged by a visual component which supports the creation of a connector or adapter between sensor and Sensor Web. Focus of this work is such a component which enables the visual creation of instances of the SID model.

3 Sensor Interface Descriptors

The architectural principle of a Sensor Web infrastructure including the usage of SIDs is shown in Figure 2. A sensor communicates with a data acquisition system in its specific sensor protocol over a transmission technology such as RS232 or Ethernet. This sensor can also act as a sensor gateway (network sink) so that other nodes of a (possibly mobile) sensor network communicate with it. The SID interpreter runs on the data acquisition system and uses SID instances for the different sensors of the sensor network to translate between the sensor specific protocol and the SWE protocols. The interpreter is responsible to register a sensor at a SWE service and to upload sensor data to an SOS. It is also responsible for the opposite communication direction and forwards tasks received by an SPS to a sensor.

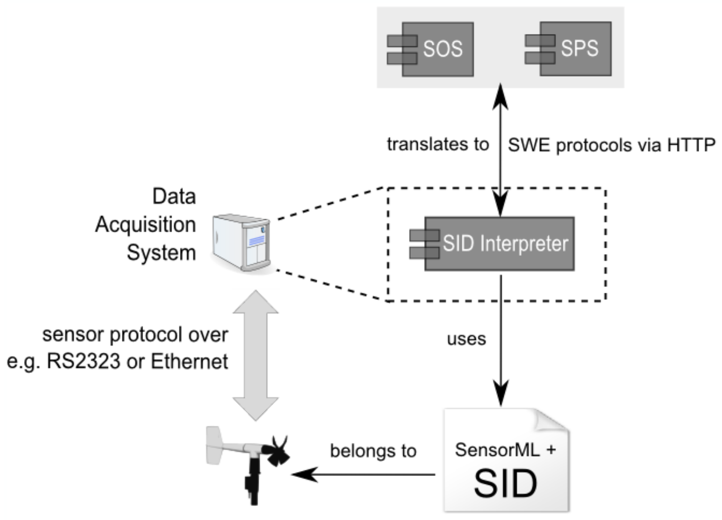


Fig. 2. - Connection of a sensor to SWE services through an SID interpreter.

A strong requirement of the design of the SID model is the strict encapsulation of the SID within the SensorML document. The SID part of the SensorML document is specific for a certain sensor type, not a particular sensor instance. Hence, an encapsulation allows reusing it in the SensorML descriptions of different sensors which are of the same type. The approach developed here, encapsulates the SID within the *InterfaceDefinition* element of a SensorML document.

The *InterfaceDefinition* element contains a stack of layers (Figure 3), aligned with the Open System Interconnection (OSI) reference model. In contrast to the OSI model, SensorML does not further define how to use these layers. The SID model makes use of this layer stack and concretizes its usage to describe the sensor interface.

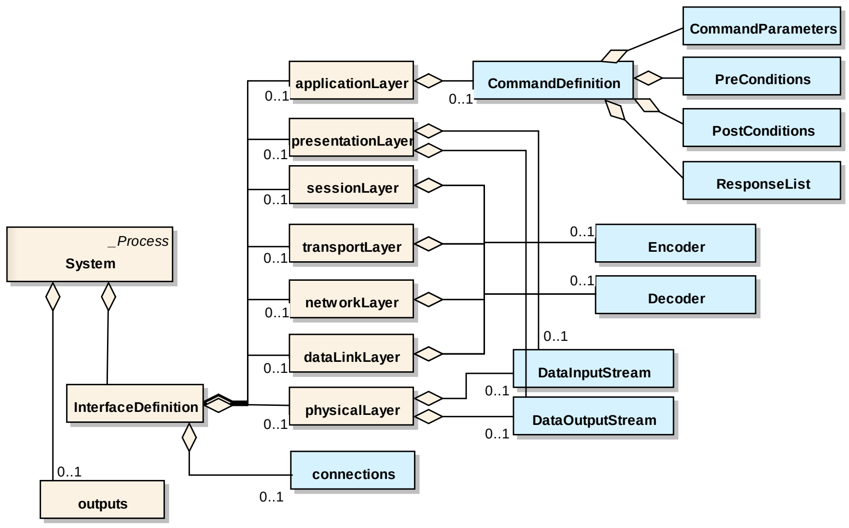


Fig. 3. - Excerpt of SensorML schema (beige colored data types) and an overview of the encapsulated SID extension (blue colored data types).

The addressing parameters (e.g. port and baud rate of a serial connection) are the basis for establishing a physical connection to the sensor. This physical connection is established through the operating system which runs the SID interpreter. The addressing parameters are stored in an external document referenced by the SID, since the SID can be published publicly (e.g. via a SWE service) and the addressing parameters are security relevant.

After establishing the physical connection, a definition of the raw sensor protocol exchanged between sensor and data acquisition system is essential. We describe the structure of these raw data within the lowest, the *physicalLayer* element. As shown in Figure 3, new elements for the data input and data output stream are attached to this element. The two elements are necessary to support duplex communication with sensors.

For enabling the definition of processing steps which are necessary to translate between the sensor protocol and the SWE protocol, the *dataLink-*

Layer, *networkLayer*, *transportLayer*, and *sessionLayer* are utilized. To allow data processing in both directions, from sensor domain to SWE domain and the other way round, elements for data decoding and encoding are added to each layer (Figure 3). Instances of these elements contain descriptions of applied processing steps. Here, the SID model reuses existing SensorML types to define processes with its inputs, outputs, parameters and its computational method.

An example for a typical usage of the layers to process a data stream coming from a sensor and to encode it to SWE protocols can look like this: the data link layer specifies a process for character escaping, the network layer computes a checksum validation, the transport layer transforms the raw data to observations by applying an interpolation, and the session layer computes a date conversion.

The data, resulting from the preceding processing steps, have to be associated with certain metadata, which is part of the O&M model (Section 2), before it can be forwarded to an SOS. The measured data need to be associated with units of measure. Further, the data need to be linked to the elementary SWE components, the observed property and the feature of interest, so that observations of the O&M model can be built and inserted into an SOS.

While the association of the data with a unit of measure is done on the *presentationLayer*, the link to observed property and the feature of interest is established in the outputs element of the SensorML document. This outputs element is not part of the SID, since it is not a sub-element of the *InterfaceDefinition* (Figure 3). The contained information is intentionally kept out of the SID, since the linkage of a sensor to feature of interest and observed property is dependent on the particular use case, not the interface of the sensor type. By not including this information into the SID, a reusing of the SID in different SWE deployments is possible.

The application layer of the OSI model describes interfaces to access the OSI stack. Compliant to this view, the *applicationLayer* is used here to define the commands accepted by the sensor. These command definitions can be used by an SPS so that it can provide information to the clients on how to task the sensor. As shown in Figure 3, the *command* element contains sub-elements to describe possible sensor responses, the pre- and post-conditions for executing the command, as well as the command parameters.

The implementation of our SID interpreter is based on the OSGi framework[3] which is extendible by pluggable and loosely coupled components.

[3] http://www.osgi.org/

An overview of the architectural design of the SID interpreter implementation is depicted in Figure 4.

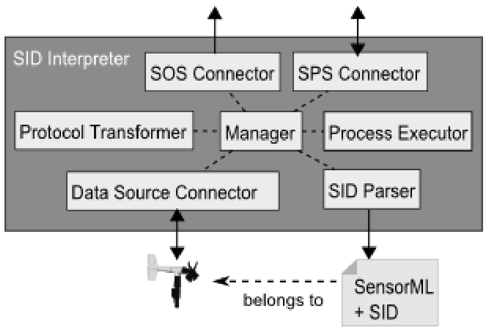


Fig. 4. - Overview of SID interpreter implementation.

A central *Manager* component controls the workflow. First, the *SID Parser* is used to read in the SID document of the sensor. Depending on the specified addressing parameters, a particular *Data Source Connector* implementation (e.g. for USB connections) is chosen to connect to the sensor. Based on the protocol definition of the SID, the *Protocol Transformer* communicates with the sensor in a bi-directional way. The *Process Executor* is able to execute the four native process methods. Also, user-defined MathML processes can be executed by means of the MathML Solver library[4]. The *SOS Connector* triggers the SOS operation *RegisterSensor* to add the new sensor to the Sensor Web and executes the *InsertObservation* operation to upload sensor data as observations to an SOS. The *SPS Connector* forwards the SensorML document and the contained SID to an SPS which uses the sensor command descriptions to provide detailed information on how to task the sensor. Sensor tasks, submitted to the SPS, are forwarded by the SPS to the *SPS Connector*. The tasks are transformed to the sensor protocol, and passed through the *Data Source Connector* to the sensor.

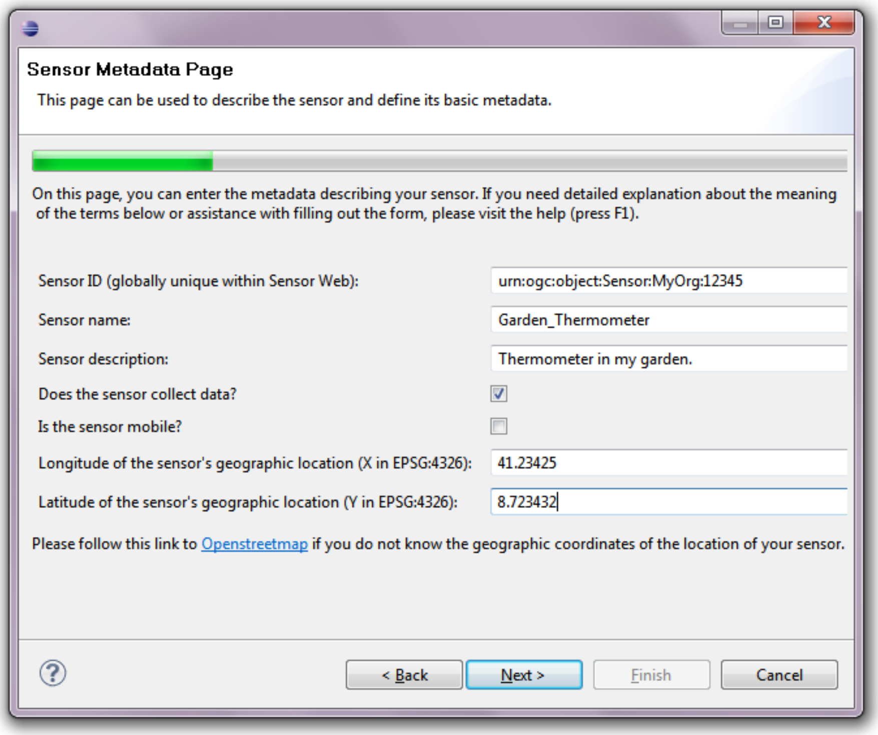
4 A Visual Creator for Sensor Interface Descriptors

The creation of SensorML and contained SID code without tool support is tedious and error-prone, since plain XML has to be written by hand. For this reason, the visual SID creator has been developed which enables a semi-automatic generation of SID instances. This SID creator follows the *wizard* user interface pattern [13] and consists (in the version used in this

[4] http://sourceforge.net/projects/mathmlsolver

work) of four pages for the different aspects of the SID design. Labels and descriptions guide the user in filling out the forms of each wizard page. Additionally, a dynamic context help can be consulted for each page which contains detailed information about all input fields. The syntactic validity of user inputs is directly checked and feedback is given in case of invalid input. The user is only able to go to the next page of the wizard if all fields are completed correctly. A bar on top indicates the overall process of the SID creation.

The first page of the wizard allows the definition of the directory where the generated SID file is saved after creation. The second page (Figure 5) prompts the user to specify basic metadata about the sensor. This includes the globally unique identification within the Sensor Web, a human readable name and description of the sensor, as well as its geographic location. The specified data are pasted in particular SensorML tags when the file is generated. Since SensorML is generic and does not explicitly specify where to put this information, we follow a public profile of SensorML which is optimized for discovery of sensors [14] to encode these data.


Sensor Metadata Page

This page can be used to describe the sensor and define its basic metadata.

On this page, you can enter the metadata describing your sensor. If you need detailed explanation about the meaning of the terms below or assistance with filling out the form, please visit the help (press F1).

Sensor ID (globally unique within Sensor Web):	urn:ogc:object:Sensor:MyOrg:12345
Sensor name:	Garden_Thermometer
Sensor description:	Thermometer in my garden.
Does the sensor collect data?	✓
Is the sensor mobile?	☐
Longitude of the sensor's geographic location (X in EPSG:4326):	41.23425
Latitude of the sensor's geographic location (Y in EPSG:4326):	8.723432

Please follow this link to Openstreetmap if you do not know the geographic coordinates of the location of your sensor.

Fig. 5. - Basic metadata description page of the SID creator.

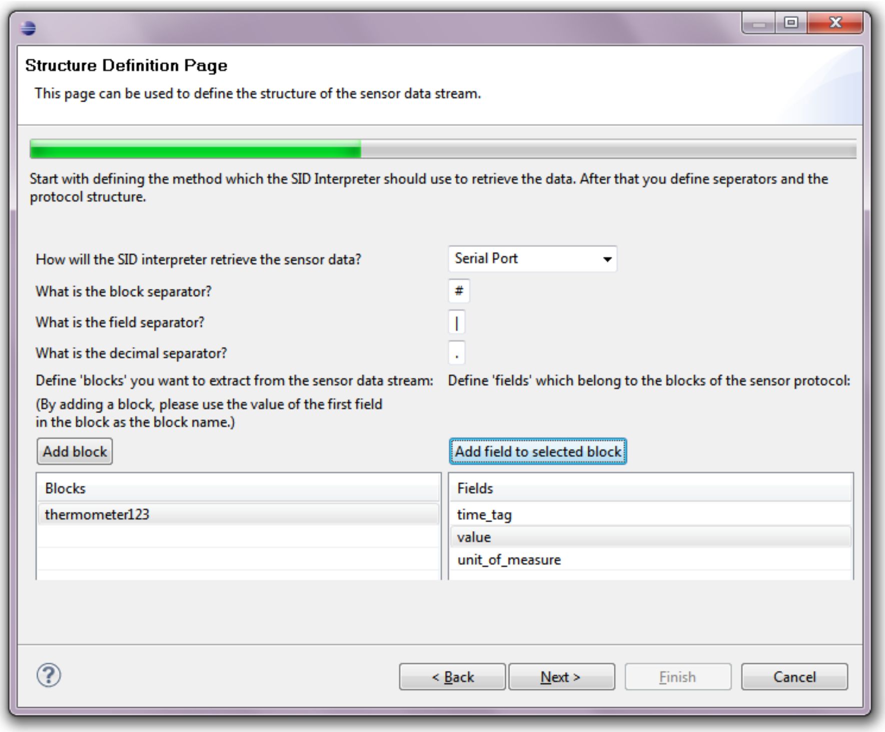
The third page (Figure 6) of the wizard enables the definition of the sensor protocol. First, the user chooses how the SID interpreter retrieves data from the sensor. Alternatives are, for example, the serial port, USB, Ethernet, or a file-based connection where the communication with the sensor takes place through a file on the hard disk of the data acquisition system.

Next, the separator signs of the sensor protocol are being defined. Those signs are utilized by the protocol to separate blocks, fields within a block and decimal numbers. Afterwards the structure of the protocol is defined. The SID creator allows specifying multiple blocks within the data stream coming from the sensor. For those blocks between 1 to n contained fields can be defined. An example of such a block is given in Listing 1 and its description with the SID creator is shown in Figure 6.

```
… # thermometer123 | 2010-09-02T13:05 | 22.34 | °C # …
```

List. 1 - A single block within a sensor data stream.

The block is identified within the sensor data stream by the value of its first field, *thermometer123* in case of Listing 1. This block ID is also specified in the wizard (Figure 6). Further, three fields are added to the block. The second field is the value of the measured data which is of interest and referenced on the next wizard page.



Structure Definition Page

This page can be used to define the structure of the sensor data stream.

Start with defining the method which the SID Interpreter should use to retrieve the data. After that you define seperators and the protocol structure.

How will the SID interpreter retrieve the sensor data? Serial Port

What is the block separator? #

What is the field separator? |

What is the decimal separator? .

Define 'blocks' you want to extract from the sensor data stream:

(By adding a block, please use the value of the first field in the block as the block name.)

Define 'fields' which belong to the blocks of the sensor protocol:

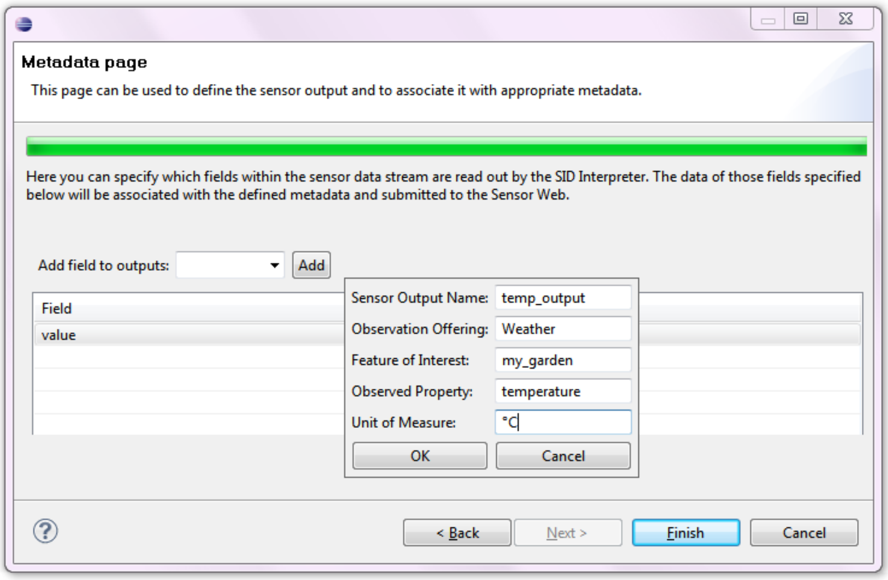
Blocks
thermometer123

Fields
time_tag
value
unit_of_measure

Fig. 6. - Structure definition page of the SID creator.

The fourth page (Figure 7) defines the sensor data output which shall be uploaded to the Sensor Web. Further, the SWE related metadata, such as the observation offering, feature of interest, observed property and unit of measure (Section 2), can be associated with the sensor data output. Those metadata are needed by the SID interpreter to create O&M encoded observations and to call the *InsertObservation* operation of the SOS every time data is coming in from the sensor (in its configured sampling rate).

After finalizing this page, the wizard creates a complete SensorML description for the sensor with a contained SID as defined by the user.


Metadata page

This page can be used to define the sensor output and to associate it with appropriate metadata.

Here you can specify which fields within the sensor data stream are read out by the SID Interpreter. The data of those fields specified below will be associated with the defined metadata and submitted to the Sensor Web.

Add field to outputs: Add

Field
value

Sensor Output Name: temp_output
Observation Offering: Weather
Feature of Interest: my_garden
Observed Property: temperature
Unit of Measure: °C

Fig. 7. - Metadata association page of the SID creator.

5 User Study

The use study was conducted to analyze the usability of the SID creator and to find out whether the developed concepts help to facilitate the integration of sensors with the Sensor Web. The participants were tasked to utilize the SID creator to describe the protocol of a home weather station[5] and to associate the measured sensor data with SWE metadata. By completing the task correctly, the SID creator would output an SID file which can be used by the SID interpreter to automatically register the sensors at an SOS and translate the measured sensor data to O&M and upload it to the SOS server. This setup is depicted in Figure 8. The SID interpreter runs on a computer (data acquisition system) with a USB connected weather station. The weather station writes the measured data continuously every 10 minutes to a data file on the hard drive of the computer.

[5] A common DAVIS weather station (http://www.davisnet.com/) was chosen.

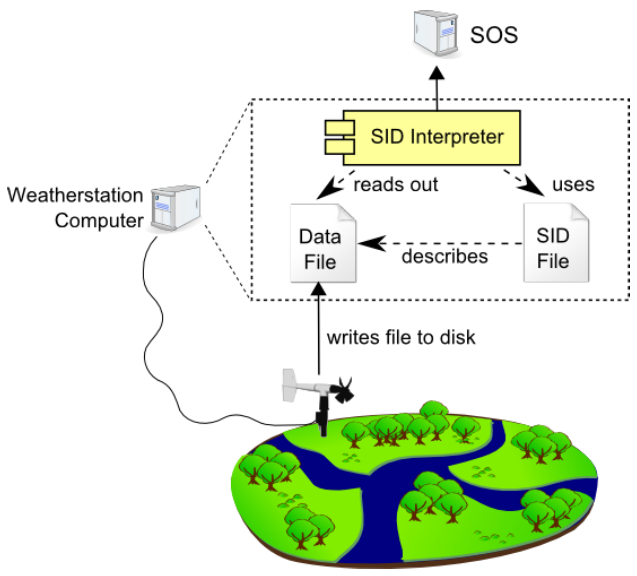


Fig. 8. - Overview of the user study setup.

The structure of the data file written by the weather station represents the sensor protocol which has to be described within an SID. Listing 2 shows an instance of such a data file. Besides basic metadata, the file contains measured data for a particular time stamp from a wind speed sensor, a wind direction sensor as well as a thermometer. In the user study, the participants were tasked to focus on the thermometer and to describe its protocol in an SID.

```
Sensor_Type;DavisWeatherStation#
Coordinate_System;EPSG4326#
Coordinates;52.223;7.544#
Time_Stamp;2010.09.30;12:57:46#
WindSpeedSensor;WindSpeed;34;m/s#
WindDirectionSensor;WindDirection;270;deg#
Thermometer;Temperature;22.34;°C#
```

List. 2 - Weather station data file.

The participants of the study were selected with the intention of having users with varying experience, so that not only the behavior of expert users, but also the behavior of users, who have only little computer knowledge, could be studied. Overall, 20 people took part in the study. Nine participants were high school students[6] aged 17 to 19. Among this group, four had moderate computer experience (e.g. only office programs etc.) and no programming skills. In the following, we refer to those participants as *Group A*. The other five high school students (*Group B*) were attending a computer science course and had good computer experience and basic programming skills in Java and Delphi. Among the other eleven participants, six had at least a Bachelor of Science (BSc) degree in computer sciences (*Group C*). The other five participants (*Group D*) were the most qualified group and had at least a BSc degree in computer sciences and also experience with the SWE specification framework and the SOS in particular. None of the participants had prior knowledge of the SID concept.

All participants were given a 25 minutes introductory presentation explaining the basic idea of the Sensor Web and the relevant standards, i.e. the principle of the SOS as well as the central metadata components of SensorML and O&M (Section 2). A description of the SID concept and the weather station protocol was also part of the presentation. Due to the different levels of user experience, the presentation[7] was kept simple and did not go into encoding details. After this introduction, each participant was given a short written task description and could ask final questions to make sure the task was understood. The test was conducted by applying screen logging in combination with the "Think Aloud" method [15, 16, 17][15, 16, 17], i.e., the participants were supposed to talk about what they were doing and thinking what difficulties they had while utilizing the SID creator. The voice, the screen as well as the duration of each test were recorded. During the test, the interaction between experimenter and participant was minimized. If advice or help was given by the experimenter it was taken note of and such interferences are reflected in the evaluation of the study (Section 6). After finishing the test, the participants were also asked to complete a questionnaire.

6 Analysis and Evaluation of the User Study

As expected, the experienced Group D was most successful in creating valid and working SID instances. Four of five members produced a working SID for the weather station. Two of the six participants with a BSc de-

[6] The high school students took part in a one week school project which aimed at making the school's weather station available on the Sensor Web.

[7] The interested reader can download the presentation here: http://ifgi.uni-muenster.de/~arneb/SID_Creator.pdf. Please be aware that the participants were all German native speakers, hence, the presentation as well as the text of the SID creator pages were kept in German.

gree in computer science but without SWE knowledge (Group C) were also successful. In each of the two groups of high school students (Group A and B) one person created a working SID.

From analyzing the user study recordings, it is noticeable that mistakes made by the participants happened repeatedly and can be classified. Overall, the 20 participants made 45 mistakes. Thereby, it has also been counted as a mistake, if a participant requested advice for a particular problem and the experimenter interfered. Figure 9 shows the average number of such mistakes per person, separated for each participant group and wizard page[8]. The diagram shows that the average number of mistakes per person decreases with increasing level of experience. The high school students (Group A and B) as well as Group C made most of their mistakes on page 3, where the structure of the sensor protocol needs to be defined. On page 4 (Figure 7), each group made almost the same number of around 0.5 mistakes per person. On page 2, very few mistakes were made (only two mistakes by members of Group B and C) showing that this page is rather easy to complete.

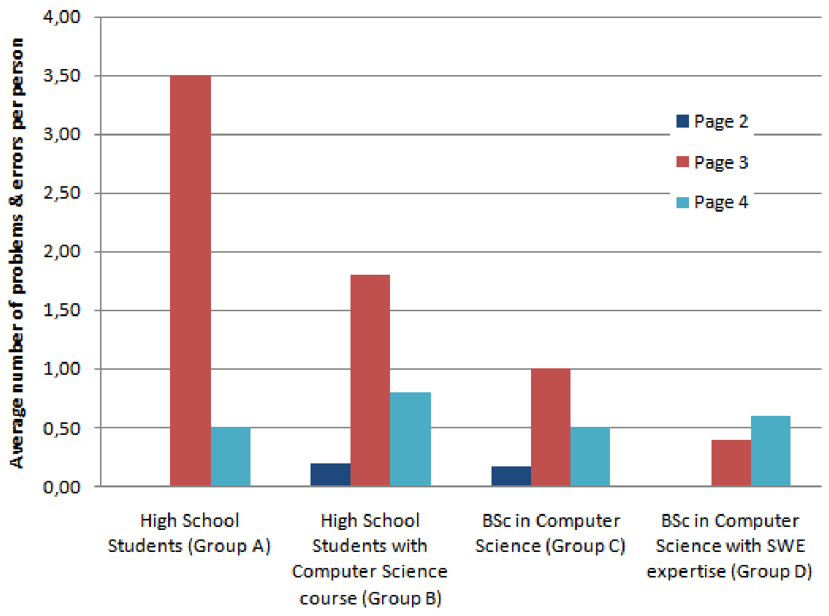


Fig. 9. - Average number of mistakes per person for each wizard page.

[8] Since no participant made a mistake on page 1 of the wizard, it is not considered here.

Figure 10 shows the relative frequency of the kinds of errors that occurred. Most often, namely 20 percent of all mistakes, a wrong sensor output field was chosen on page 4 (Figure 7). In this particular user study, the third field of the thermometer block must have been specified as the output of the sensor. Instead, five participants chose a different field and four participants needed advice to choose the correct one. Also on page 4, a wrong metadata association (e.g., unit of measure was set to "*22.34*") happened in three cases.

On page 3 (Figure 6), a wrong connector specification (e.g., a USB connector was chosen instead of a file-based connection) and mistaken block definition (e.g., the chosen block identifier did not match the block name in the protocol) were each counted eight times. Also on page 3, mistakes in the field definition were made (e.g., not all fields of the block defined). Another mistake on page 3 was the naming of the third field in the protocol as "*22.34*" instead of "*temperature_value*". This does not lead to an invalid SID but shows a misunderstanding of the concept. It happened four times among the nine high school students and once in Group C. Four high school students needed help from the experimenter to specify the separator signs (e.g., the experimenter had to recapitulate what separator signs are).

On page 2 (Figure 5), only one kind of mistake was made by two participants: the checkbox whether the sensor collects data was not checked.

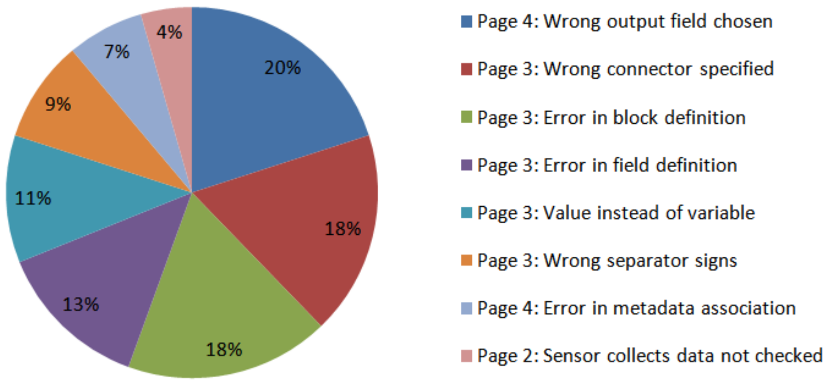


Fig. 10. - Relative frequency of occurred kinds of errors.

Figure 11 shows the average time per person for editing the different SID creator pages. The diagram shows that the editing of page 2 took almost the same amount of time for all groups. In Group A, B and C the editing of page 3 took the longest time which indicates that the definition of the sensor protocol structure was most difficult for them (Group A needed the most time, namely 6 minutes and 52 seconds in average).

Group D put most of the time (3:47) in the definition of the sensor output and its metadata. On page 4, a common misunderstanding was that not only the actual data values but also other fields of the sensor protocol need to be defined as sensor output and uploaded to the SOS. This is not the case since the information contained in the other fields is static and specified as associated metadata by the user of the SID creator.

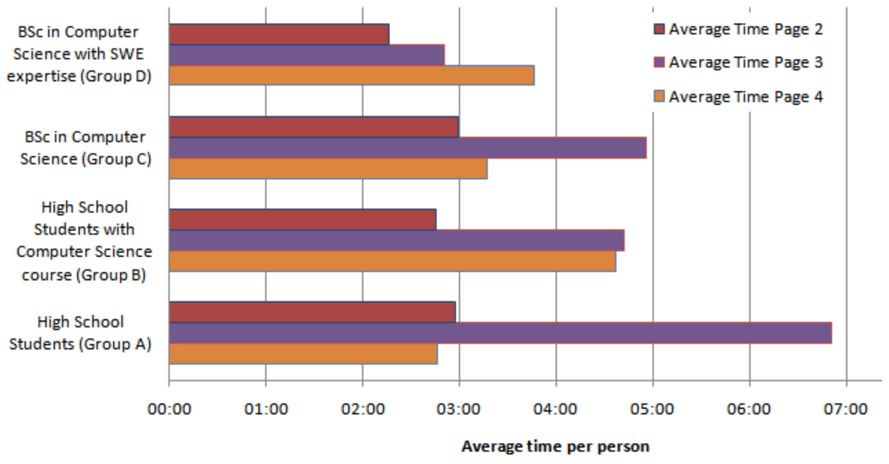


Fig. 11 - Average time per person needed for editing a wizard page.

After each test, the participant was asked to complete a questionnaire. For example, the participants were asked whether they think they have fully/partly/not understood the principle of the SID concept and whether it can replace manual implementation of adapters between sensor and SOS. The answer to this question should indicate a self-report measure about how sure the participant is of what he/she just did. In Group D, all five members stated to have "fully" understood the SID concept. In Group C, one person answered the question with "partly", the other five members answered with "fully". In Group B two persons answered with "fully" and three with "partly". In the most inexperienced Group A, two people stated to have "fully" understood the SID concept and the other two did "not" understand.

Additionally, the five members of Group D, who had already connected a sensor manually to an SOS, were asked whether it is easier to use the visual SID creator instead of implementing an adapter. All of them perceived the SID creator as a helpful tool for the given task and answered with yes. However, three raised the question whether the SID creator in its current design has enough functionality to support all kinds of sensor types. It was assumed that complex sensor interfaces still require a manual implementation of adapters.

7 Conclusion and Outlook

To close the interoperability gap between sensors and the Sensor Web the SID model has been developed based on the SensorML standard. It enables the declarative description of a sensor protocol. An SID interpreter is able to translate the sensor protocol to Sensor Web protocols based on the knowledge contained in an instance of the SID model. This paper presents the SID creator which enables users to visually generate SID instances for their sensors. The SID creator prevents users from manually implementing adapters for each sensor type which shall be integrated with the Sensor Web. Instead, an additional interface description, the SID, can be created which enables the integration of the sensor with the Sensor Web. Once a sensor interface is described by an SID, it can be used in multiple applications by different user communities. Together, the SID model and SID creator are beneficial for sensor manufacturers and sensor data providers who do not have to change their sensor's protocols.

The usability and usefulness of the SID creator was evaluated by conducting a user study. The participants, ranging from high school students to SWE experts, were tasked to create an SID for the sensors of a weather station. The analysis of the user study showed that the SID creator was very useful for the group of SWE experts. They stated that it is easier to use the SID creator to integrate a sensor with the Sensor Web than implementing an adapter manually. Four of five members of that group created a working SID. In the group of people with a BSc degree in computer sciences, who did not have experience in SWE, one third created a working SID and over 80% of those users stated to have fully understood the SID concept. Significantly higher error rates and average time consumption showed that the task was most difficult for the high school students. However, also here two out of nine people were able to create a working SID without any prior experience in integrating sensors with the Sensor Web. Finally, the results of the user study lead to the conclusion that, with increasing level of user experience, the SID creator is a helpful tool and considerably facilitates the process of sensor integration.

For the future, in particular the protocol definition page of the SID creator needs to be improved, since it caused most of the problems as the results of the user study have shown. This can be done, e.g., by enhancing the help texts and including descriptive examples, or by enhancing the current form-based user input to a more graphical design. Also, the SID creator used in this work does not yet allow configuring all details of an SID and not all sensor protocols can be defined. Further extending the SID creator to fully support the SID model will broaden the range of sensor types for which SIDs can be created.

References

[1] M. Botts, G. Percivall, C. Reed, and J. Davidson, "OGC Sensor Web Enablement: Overview and High Level Architecture," *Lecture Notes In Computer Science*, vol. 4540, pp. 175–190, 2008.

[2] S. Jirka, A. Bröring, and C. Stasch, "Applying OGC Sensor Web Enablement to Risk Monitoring and Disaster Management," in *GSDI 11 World Conference, Rotterdam, Netherlands*, June 2009.

[3] C. Stasch, A. C. Walkowski, and S. Jirka, "A Geosensor Network Architecture for Disaster Management based on Open Standards." in *Digital Earth Summit on Geoinformatics 2008: Tools for Climate Change Research.*, M. Ehlers, K. Behncke, F. W. Gerstengabe, F. Hillen, L. Koppers, L. Stroink, and J. Wächter, Eds., 2008, pp. 54–59.

[4] K. Walter and E. Nash, "Coupling Wireless Sensor Networks and the Sensor Observation Service - Bridging the Interoperability Gap," in *12th AGILE International Conference on Geographic Information Science 2009*, Hannover, Germany, 2009.

[5] K. Aberer, M. Hauswirth, and A. Salehi, "A Middleware for Fast and Flexible Sensor Network Deployment," in *32nd International Conference on Very Large Data Bases*, 2006.

[6] A. Na and M. Priest, *OGC Implementation Specification 06-009r6: OpenGIS Sensor Observation Service (SOS)*. Open Geospatial Consortium, 2007.

[7] I. Simonis, *OGC Implementation Specification 07-014r3: OpenGIS Sensor Planning Service*. Open Geospatial Consortium, 2007.

[8] S. Cox, *OGC Implementation Specification 07-022r1: Observations and Measurements - Part 1 - Observation schema*. Open Geospatial Consortium, 2007.

[9] M. Botts, *OGC Implementation Specification 07-000: OpenGIS Sensor Model Language (SensorML)*. Open Geospatial Consortium, 2007.

[10] P. Hu, R. Robinson, and J. Indulska, "Sensor Standards: Overview and Experiences," in *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing ISSNIP'07*, Melbourne, Australia, 3-6 December 2007.

[11] T. Bleier, B. Bozic, R. Bumerl-Lexa, A. Da Costa, S. Costes, I. Iosifescu, O. Martin, S. Frysinger, D. Havlik, D. Hilbring, P. Jacques,

M. Klopfer, S. Kunz, P. Kutschera, M. Lidstone, S. Middleton, Z. Roberts, Z. Sabeur, J. Schabauer, S. Schlobinski, T. Shu, I. Simonis, B. Stevenot, T. Usländer, K. Watson, and K. Wittamore, *SANY - An Open Service Architecture for Sensor Networks*, M. Klopfer and I. Simonis, Eds. SANY Consortium, 2009.

[12] G. Gigan and I. Atkinson, "Sensor Abstraction Layer: A Unique Software Interface to Effectively Manage Sensor Networks," in *3rd International Conference on Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007*, 3-6 2007, pp. 479–484.

[13] J. Tidwell, *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly, 2006.

[14] S. Jirka and A. Bröring, *OGC Discussion Paper 09-033 - SensorML Profile for Discovery*. Open Geospatial Consortium, 2009.

[15] J. Nielsen, *Usability Engineering*. Morgan Kaufmann, 1993.

[16] M. van Someren, Y. Barnard, and J. Sandberg, *The Think Aloud Method - A Practical Guide to Modelling Cognitive Processes*. London: Academic Press, 1994.

[17] C. van Elzakker, I. Delikostidis, and P. van Oosterom, "Field-Based Usability Evaluation Methodology for Mobile Geo-Applications," *The Cartographic Journal*, vol. 45, no. 2, pp. 139–149, 2008.

[18] A. Bröring, S. Below, and T. Foerster, "Declarative Sensor Interface Descriptors for the Sensor Web," in *WebMGS 2010: 1st International Workshop on Pervasive Web Mapping, Geoprocessing and Services*, Como, Italy, 26.-27. August 2010.

[19] A. Bröring and S. Below, *OGC Discussion Paper 10-134: Sensor Interface Descriptors*. Open Geospatial Consortium, 2010.