StarFL – A new Metadata Language for Sensor Descriptions

Or: Why do we need yet another metadata language for sensor descriptions?

Christian Malewski[1], Ingo Simonis[2], Andrew Terhorst[3], Arne Bröring[1,4,5]
[1]Institute for Geoinformatics, University of Münster, Münster, Germany
[2]International Geospatial Services Institute, Emden, Germany
[3]Commonwealth Scientific and Industrial Research Organisation, Australia
[4]ITC Faculty, University of Twente, Enschede, The Netherlands
[5]52° North Initiative for Geospatial Open Source Software, Münster, Germany
{c.malewski | arneb}@uni-muenster.de, ingo.simonis@igsi.eu, andrew.terhorst@csiro.au

Abstract - An ever-increasing number of sensor resources are being exposed via the World Wide Web. Discovery, selection and use of these sensors and their observations require a robust sensor information model, but the homogenous description of sensor metadata is a complex and difficult task. Currently, the only available robust model is SensorML, which is intentionally designed in a very generic way. Due to this genericness, interoperability can hardly be achieved without the definition of application profiles that further constrain the use and expressiveness of the root language. So far, such SensorML profiles have only been developed up to a limited extent. This work describes a new approach for defining sensor metadata, the StarFL model. This language follows a more restrictive approach and incorporates concepts from the recently published Semantic Sensor Network Ontology to overcome the key issues users are experiencing with SensorML. StarFL defines a restricted vocabulary and model for sensor metadata to achieve a high level of interoperability and a straightforward reusability of sensor descriptions.

(key words) SWE; SensorML; Sensor Metadata; Sensor Web; Sensor Web standards and standard implementations; Sensor Web applications, deployments, and best practices; Usage Techniques

I. INTRODUCTION

An ever-increasing number of sensor resources are being exposed via the World Wide Web today. To enable uniform discovery and usage of sensors and associated observations, a robust sensor information model that describes both the sensor and its deployment in the real world is required. Due to the plethora of sensor manufacturers, sensor types, sensor access protocols, and sensor deployment situations, development of such a model is not straightforward. Further, user requirements on sensor description vary across communities and scientific disciplines.

The Sensor Web Enablement (SWE) initiative of the Open Geospatial Consortium (OGC) standardizes web service interfaces and data encodings as building blocks for the Sensor Web [1]. For modelling and encoding sensor metadata the SensorML standard [2] is proposed by SWE. SensorML is very flexible and allows modelling virtually any type of sensor in multiple ways. However, this flexibility comes at some considerable cost. Both new and experienced users struggle with creating sensor description instances, as there is not always a single unambiguous way to encode certain sensor characteristics. The results are interoperability issues at the encoding level if SensorML parsers do not support the full language spectrum, or usability issues, as SensorML receivers have to adapt their applications to various flavours of SensorML. Hence, it is crucial to develop constraining application profiles to ensure the interoperability of SensorML instances. Profiles mandate the presence of certain elements and thus restrict the genericness of SensorML. However, today, after four years of the SensorML standard approval, established profiles are still missing; to the best knowledge of the authors only one has been defined, so far (see [3, 4]).

Meanwhile new applications, such as provenance information systems and domain comprehensive sensor discovery systems, have raised the need for detailed and interoperable sensor descriptions. Further on, concepts of the semantic web, such as ontologies and ontology brokers have been developed to either complement or replace the concept of SensorML. The Semantic Sensor Network Ontology (SSNO) [5], developed by a W3C incubator group, identifies distinct elements of a sensor network and provides definitions of sensor concepts with focus on sensor type organization and classification. The SSNO is grounded in the Dolce Ultra-Light (DUL) upper ontology. That makes it hard to combine SSNO with the Observations & Measurements (O&M) [6] standard, the primary way of encoding measured sensor data as proposed by SWE. O&M models how real world features are observed in the physical and natural sciences. Integrating SSNO into SWE and mapping to SensorML is a challenging task.

We present a modularized sensor mark-up language called the Starfish Fungus Language (StarFL) that relates both languages, SensorML and SSNO. It re-uses concepts of SSNO to restructure and restrict elements of SensorML and also re-interprets their semantics to fit the SWE standards. StarFL might work as a meta-language enabling generic mapping between SSNO and a profiled SensorML. It is aligned to O&M and due to its restrictiveness aims to be highly interoperable, and user friendly. It has originally been developed for in-situ sensors but also supports remote sensors. StarFL applies the

Pareto principle (also known as the 80:20 rule) and aims to describe most sensors with the least complexity. Intentionally, it does not support every detail of a sensor or its constellation in space, but concentrates on the common features shared by a wide range of users and communities. StarFL does provide extension points for domain specific encodings that can be added by communities with special needs.

The remaining paper is structured as follows. Section 2 acquires an overview of available sensor metadata description languages. Section 3 introduces the StarFL concept; Section 4 compares elements of StarFL to SensorML and SSNO in detail. Section 5 summarizes the paper and gives an outlook on future work.

II. Backround and Related Work

SensorML was originally developed with a focus on satellite systems. It specifies a model and XML encoding to describe complete sensing processes. Physical sensors, ranging from simple sensors such as thermometers to composite instruments consisting of multiple single sensing components, as well as virtual sensors can be described. SensorML defines and models physical sensor devices as processes which provide conversions of typically some physical phenomenon (e.g., temperature, conductivity, or pressure) as input and the quantification of another phenomenon as output (e.g., water temperature, salinity, or depth). The common root is the abstract type *Process*. Additionally various metadata about the process can be specified, including its identification, classification, or contact information of the responsible provider among many others. To model physical sensors, the *System* type can be used which adds spatial (e.g., the definition of a geographic position) and temporal (e.g., the definition of a sampling time) attributes. A measurement station, which is a collection of sensing processes, can be modelled using a hierarchic combination of those system types, where the root node represents the platform. Its attached sensors are modelled as child nodes representing a *component of* relationship. Measurement stations can be either fixed or mobile Fig 1.

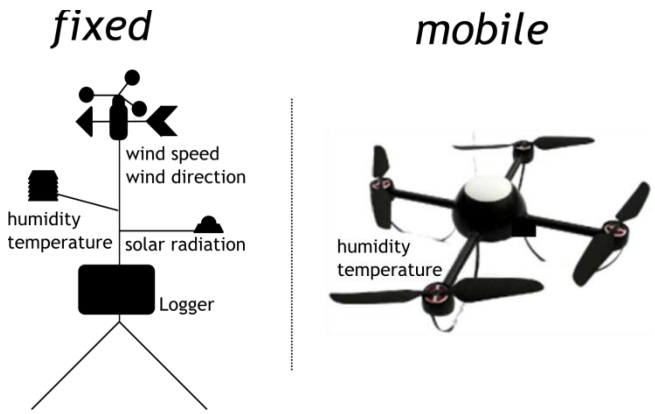

Fig 1

fixed and mobile measurement station

However, there are also cases where modelling the platform is not necessary. For an interoperable and domain comprehensive usage, SensorML allows the development of profiles. Those profiles define the core characteristics of any given type of sensor or platform. This is necessary, due to:

1. The interpretation of a certain system type, as it can adopt several functions (platform, sensor device, sensor sub-device).
2. The omnipresence of optional elements, which allow valid but otherwise meaningless SensorML instances.

In a first attempt, Jirka and Bröring [3] define a basic SensorML profile orientating on a weather measurement station. They suggest partitioning the sensor devices according to their observed property. Additionally, solutions for spatial sensor discovery are presented. However, the issues of dynamic sensor descriptions and redundancy of data and discovery for measurement capabilities are not addressed.

Besides such profiles, SensorML uses semantic annotation techniques to support interpretation of elements. Sensor ontologies provide a vocabulary that can be integrated with SensorML to enable semantic interoperability, or used apart from SensorML as an alternative approach to express sensor metadata. Avancha [7], Eid [8] introduce ontologies that mainly focus on measurements with little capacity to describe sensor systems, or how measurements are taken. Other ontologies highlight the role of stimuli, observed properties, or processes [9, 10]. The SWAMO [10] and MMI [11] ontologies extend the analysis along a third dimension, from measurements and sensor types to systems. Each ontology focuses on systems, the components of a system, and how those components are organized. OntoSensor [12] covers a wider range and is able to describe most of the spectrum of sensor concepts including definitions of SensorML and extensions to IEEE SUMO [13]. It describes sensors, their capabilities and measurands as main types. Compton et al. [14] perform a detailed survey over the further mentioned ontologies and compare them to the CSIRO sensor ontology [15], which influenced together with the Stimulus-Sensor-Observation ontology design pattern [16] the work of the W3C Semantic Sensor Networks Incubator Group. The W3C Incubator Group introduced the Semantic Sensor Network Ontology (SSNO), which is aligned with classes of DOLCE Ultra Lite (DUL) foundational ontology to facilitate reuse and interoperability. It provides concepts to align devices in a sensor context by differentiating between a platform and a sensor. A sensor is defined by its observed property qualified by measurement capabilities and inherits from a system concept that itself has capabilities such as operational range, survival range or deployment. The Sensor Interface Descriptor approach [17, 18] which is based on SensorML goes beyond the metadata description of the sensor instance, and includes the description of the sensor communication interface, similar to IEEE 1451 [19]. StarFL focuses on modelling sensor capabilities, leaving aside the description of the actual sensor interface.

III. The StarFL concept

StarFL makes use of concepts from both SensorML and SSNO. It is described using UML and implemented as XML

schema using the *Hollow World*[1] environment, which provides a framework to develop GML models. StarFL is modularized and consists of two main modules, the *Static Module* and the *Dynamic Module* plus an additional *Common Module*.

1) Static Module

Measuring devices of a certain model offered by sensor vendors typically have the same measuring and physical capabilities summarized in a sensor datasheet. To compact that information and avoid redundancies, StarFL separates parts of the sensor description in the *Static Module* (Fig 2). It consists of three classes. The device-focused part is modelled within the *SensorCharacteristic* class and describes physical attributes (e.g. height, weight), operational attributes (e.g. survival range) and model identification attributes such as the manufacturer or the model number.

A sensor model typically implements one or many procedures to compute measurement results for a phenomenon. The StarFL sensor model encapsulates each procedure for a distinct observed property as a *SensingProcedure*. Besides the observed property, the measurement units and capabilities, which are aggregated by the *MeasurementCapability* class, can be expressed.

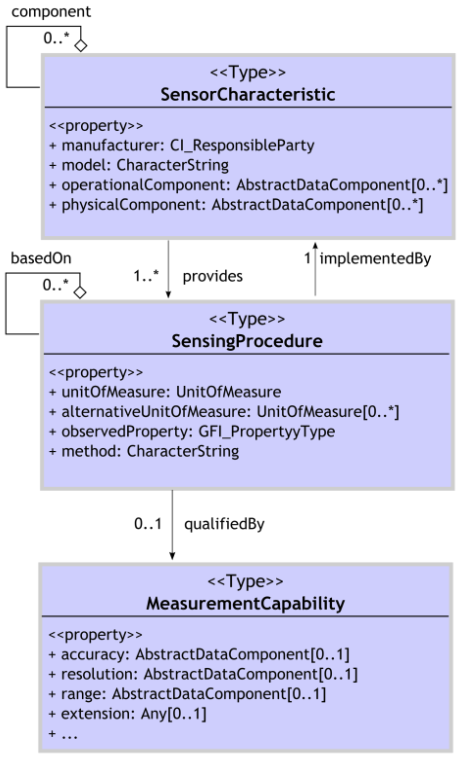

Fig 2

Static Module

Both, *SensorCharacteristic* and *SensingProcedure* perform a self-aggregation pattern, which allows the modeller to express a sensor component hierarchy on the device side and a sensing chain on the operational side. Fig 3 shows a sensor characteristic tree with the root as the main device (in our case, a Vaisala weather station) consisting out of subcomponents each responsible for a collection of sensing procedures.

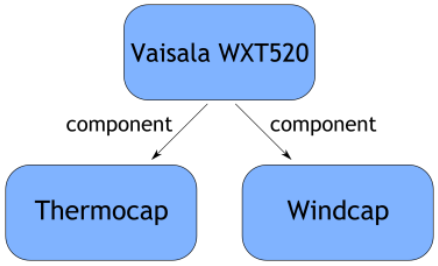

Fig 3

SensorCharacteristic Component hierarchy

A *SensingProcedure* can be based on several other procedures implemented in the same *SensorCharacteristic* hierarchy tree. The combination of the sub-procedures with the succeeding procedure is described via its *method* attribute. As a convention, the *SensingProcedure* is described only once in detail at the most atomic level of a *SensorCharacteristic* component. The upper *SensorCharacteristic* elements refer to the detailed descriptions using the W3C recommended XLink technology [20]. Fig 4 shows the *basedOn* pattern in combination with the relation between the SensorCharacteristic tree and the procedure chain. The procedures generating outputs for temperature and wind speed are described in detail within the *Thermocap* and *Windcap* sensor characteristics. The root component either uses an *XLink* reference to its particular SensingProcedures, if they are described in detail in one of its components or a detailed description if it is not provided by any of its components. In Fig 4 the wind chill computation is based on temperature and wind speed procedures.

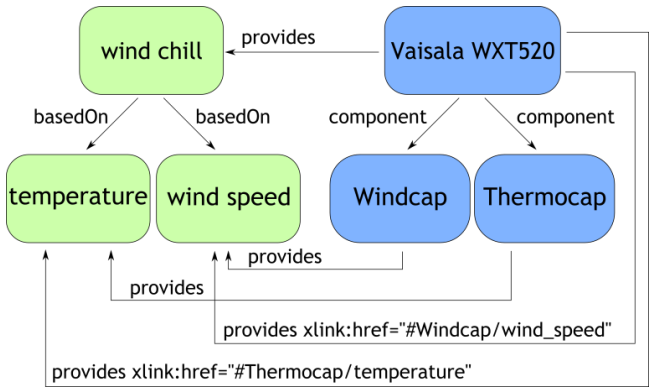

Fig 4

Sensing chain and linkage of operational and device part

StarFL narrows the definition of SensorML and SSNO, which interpret sensors to be physical objects performing observations and transform an incoming stimulus to another representation. In StarFL sensors are physical devices with several implemented processes that perform observations.

[1] https://seegrid.csiro.au/wiki/AppSchemas/HollowWorld

2) Dynamic Module

The *Dynamic Module* models the second common characteristic of sensors understood as physical devices: They are aligned to time and space. These are dependent on the usage of a certain physical sensor instance and non-static, such as operating location and time, deployment or calibration. The representations for the devices deployed in the real world are instances of the class *Sensor*.

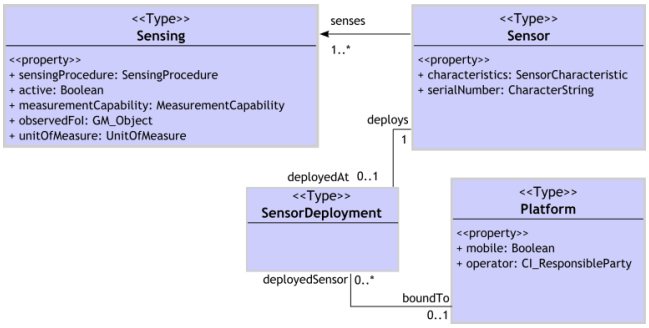

Fig 5

Dynamic Module (Part)

Usually a physical sensor device deployed in the field is of a certain type and model. Once the static description of that explicit sensor model is accessible at a URI, the dynamic part may include references to this static description using the already mentioned XLink technique. In that case the *Sensor* element, which represents the Dynamic Module device part, refers to its belonging SensorCharacteristic. As the deployed sensor is a real world instance, an identifying serial number is provided additionally. According to the static description it aggregates one or many *Sensing* elements, which themselves refer to their specific SensingProcedure instances (Fig 6). The *Sensing* class stores information about its status, the observed property or the used unit of measure for that concrete instance. It fits into the process element of an O&M observation, but narrows the interpretation to a sensor device.

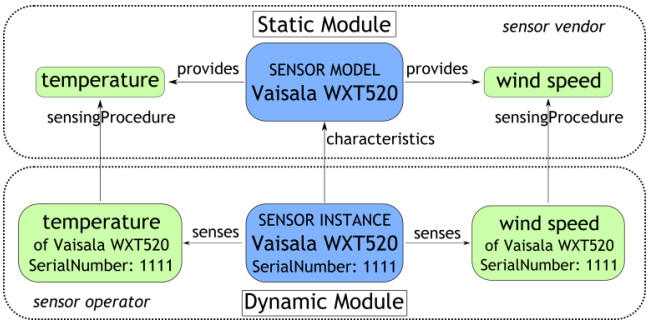

Fig 6

Connection between Static and Dynamic Module

A sensor is either mobile or deployed at a fixed location. In addition, a sensor is optionally attached to a platform, which is a physical structure that binds one or many sensors and itself is either mobile or deployed at a fixed location. The linkage happens through a *SensorDeployment* element. The analogue part for a platform is the *PlatformDeployment*. Both infer from a class *Deployment* which stores attributes such as the responsible person, the deployment location or whether the device is deployed mobile or fixed. Note that if a platform or a sensor is mobile the actual position might be detected using an integrated position sensing procedure (e.g. based on GPS). The original deployment location is not updatable. Following the approach of SensorML, StarFL supports low level encoding for calibration events. A *Calibration* is linked to a *Sensing* element. The quality of both, deployment and calibration is expressed with a *ConformanceTest* (Fig 7), which identifies tested characteristics. Organizations such as the World Meteorological Organization (WMO) or manufacturers set specifications and/or protocols on sensor deployment and calibration. For example a wind vane must be deployed in a certain distance to buildings or trees or a tipping bucket shall not leave a certain accuracy interval in a test. These requirements are described within the *ConformanceCharacteristic* element. A *ConformanceTest* refers to a *ConformanceCharacteristic* with an additional attribute saying whether a specific requirement is passed or not.

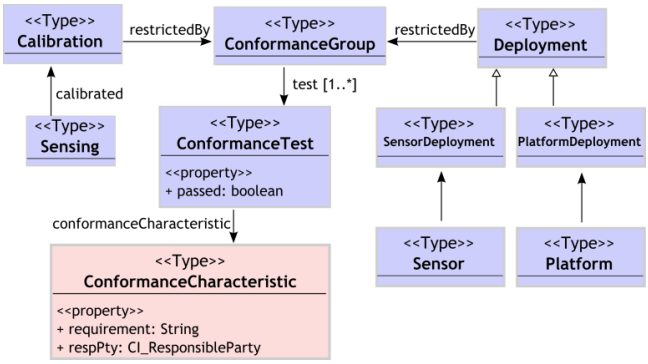

Fig 7

Quality of Calibration and Deployment

In practice both the Static and the Dynamic Module could be managed in separated catalogues. A catalogue for the Static Module could be searched for attributes such as sensor model, observed property or measurement capabilities. The instances could either be stored within the catalogue or on the manufactures website in addition to the already available sensor data sheets. A management system for the Dynamic Module must provide techniques to update attributes, such as the actual position of a sensor or deployment and calibration events. A distinct section in the static catalogue could list *ConformanceCharacteristics*. Once a static description of a sensor model is available, the creation of the application specific dynamic description of a sensor is straightforward and simple. It contains the following steps:

1. Description of all deployed sensors and sensings with a reference to their according static descriptions.
2. Description of the platform and its deployment (if any).
3. Description of linkage (SensorDeployment) between sensors and platform.
4. Description of Calibration with linkage to ConformanceCharacteristics.

Within the concept of using different catalogues the idea of linked data [21] is followed. One sensor description document is split into several. That allows reusing already available information and keeping instance information to a bare minimum. A problem of linked data is its maintenance management. If a resource is updated all referencing resources need to be checked on alignment. In our case only two links are suggested to use and both refer to static and unchangeable instances: The one from the dynamic description to the static description and the one to the ConformanceCharacteristics.

IV. Comparison to Current Sensor Description Languages

The following section compares StarFL to the earlier mentioned sensor metadata concepts SensorML and SSNO. The advantages and disadvantages to SensorML are listed in six subsections discussing semantics, the atomic process pattern, observed property and measurement capabilities, snippet management, the dynamics of metadata documents and the tool support. Regarding SSNO the alignment to O&M, the role of Sensor and Sensing, usage of additional ontologies and the sensor categorization are pointed out.

A. SensorML

StarFL and SensorML were developed from different points of view. SensorML follows the approach to provide syntactic interoperability by providing a definition language that supports every detail of sensors and sensor-to-platform constellations and therefore requires application-specific profiles from SensorML [2]. Otherwise the various levels of freedom will almost naturally lead to inconsistent sensor descriptions and raise interoperability issues eventually. If a profile should be domain comprehensive, all members have to reach agreement on that profile, which is often difficult to realize. However, the more detailed a profile should be, the more effort has to be put in and the more characteristics must be considered. That is the reason why StarFL has been designed the other way round, with a restricted collection of identifiers, properties and capabilities as a basis that captures the majority of sensors. Extension points can support every imaginable extension required by specific communities. Typically a sensor's data sheet contains all public information about a sensor model. In a direct comparison of StarFL with SensorML the following advantages can be pointed out:

1) Semantics

In SensorML the concepts *System*, *Component*, *ProcessModel* and *ProcessChain* inherit from the abstract class *Process*. Unfortunately those element values are not self-describing and their semantics can differ from application to application. To achieve interoperability, a vocabulary stating the role of an element or a profile has to be developed.

StarFL leaves that inheriting relation of measurement process elements out and instead defines clear semantic meaning of disjoint core concepts for platform, sensor (as device) and sensing (sensing process) is defined. A platform, such as a satellite, a plane or a measurement station mast is not understood to be a measurement process as in SensorML but furthermore a physical instance that aggregates sensor devices. A sensor is a device deployed in the field using a *Deployment* element and an active sensing process produces measurement results for an observed phenomenon.

2) Atomic process pattern

In terms of interoperability the concept of atomic physical components as performed in SensorML is disadvantageous. There are two cases where it does make sense to use atomic components. First, a sensor should be described in the most possible level of detail, which is a very tenacious task to perform for each single sensor instance. Herewith, the full amount of a sensor's metadata must be available. Thus, the level of detail in a sensor description is determined via the applicable data provided by the manufacturer with the assumption that a sensor could indeed be divided further. However, the end user might not be interested in the atomic process. Second, if in a specific situation a given sensor needs to be described up to a certain level of detail, all modellers shall agree on what the atomic level of detail is. This for example could be either a wind speed sensor itself or its subcomponents: the chronometer and the device that counts the rotations of a cup anemometer. Thus, in many scenarios the same element can be a Component or a System, which leads to interoperability problems and general confusion.

For that reason the atomic element pattern is left out of StarFL and hierarchic component relations and measuring procedure chains are modelled each with only one class type of node (*SensorCharacteristic* respectively *SensingProcedure*). Two assumptions underpin that decision:

a. A sensor may be divided into more detailed components than given in the manufacturer's published data sheet.

b. All dynamic sensor descriptions reference the same encapsulated hierarchic description of a certain sensor model. The end user then decides while querying metadata of a certain level of detail which information is necessary for his/her application.

3) Observed Property and Measurement Capabilities

One sensor device can observe one or many physical properties. The SensorML representation of that sensor device may model that behaviour with sensor outputs and their measurement capabilities at one hierarchic level in the document. An O&M observation instance references exactly one measurement process, which for instance might be a SensorML System element. The linkage between the observation and the metadata of its producing process therefore is complicated. The observed properties of both, observation and producing process must be compared to each other; afterwards, the particular measurement capabilities must be identified. Hence, logical encapsulation of a sensing process with only one observed property and its measurement capabilities is advantageous. A common profiled approach of process encapsulation is missing. One solution is to define a sensor to observe exactly one property in SensorML. This solution is not intuitive as it does not correlate to the physical device's dimension and leads to several problems. Physical properties of the sensor have to be repeated for every sub-

sensor in the model or one device has to be divided into more than one in the model.

In contrast, StarFL implements a different pattern that is transferable to a SensorML profile. A Sensing element logically encapsulates its observed property, additionally covers measurement capabilities and therewith implements the procedure interface of O&M. The device related data is encoded in Sensor or the SensorCharacteristic element. A mapping from StarFL to SensorML would re-interpret an sml:System element to be a physical sensor. Within its <sml:component> aggregation it would reference to its several sensing processes, which does observe only one property.

4) Snippet Management

Modellers often complain that the description of a sensor is an interminable task. The capture of metadata should not be the most time-consuming job while setting up a sensor network. The approach herewith is to reuse already described data. A common approach to manage snippets in SensorML document is hardly available. Differentiating between a static and a dynamic module in StarFL presents an optimal strategy to deal with code snippets. Once a static description is available, the user of a sensor only has to add sensor device specific spatial and temporal deployment and calibration data, if needed. It can be anticipated that static sensor data will be provided by the manufacturers, as it only requires reformatting already available data sheets

5) Dynamic of metadata documents

In practice sensors may get removed from platform, re-deployed in the environment or just change their position. In parallel to the changes of the real world device the sensor description need to be updated accordingly. SensorML provides the *history* section to document deployment or calibration related events. It does not encapsulate the appropriate attributes, though. Hence, all elements that might have changed, for example the deployment location or relative position to the platform are distributed all over the document. StarFL stores all attributes containing the deployment and redeployment in the distinct class *Deployment*. Linking sensors and modelling redeployments to a platform needs less effort to maintain and adjust descriptions. If a mobile platform might be relocated and redeployed only the *PlatformDeployment* description has to be updated. With removing a sensor device from the real world hosting platform the reference to the particular *SensorDeployment* us deleted. Attaching a new sensor is modelled by just adding a new *SensorDeployment* element. The static module does not need to be changed.

6) Tool support

An advantage of SensorML is that there are a few tools already available supporting version 1.0 of the OGC standard. However, most of them work on a very basic description level and support only specific parts of the language. SensorML generators such as SmlMor[2] often remain on a very basic level. For StarFL, though just published a little while ago, RESTful catalogues are under development and will be available soon.

In summary: StarFL is hard-typed in many ways; hence, on first sight, SensorML can serve more cases and can include a richer variety of sensor characteristics. However, StarFL is intentionally designed by providing a first set of metadata elements, which are sufficient for most cases of sensors (this even includes complex scientific scenarios). If a particular attribute is missing, StarFL supports the usage of extension elements. SensorML tackles this from a different angle by providing general attribute sections, wherein all imaginable characteristics of a sensor can be aligned. Thus to reach interoperability, the creation of profiles becomes mandatory. In this regard, StarFL can be seen as an optimal restricted profile; however, the proposal for code snippet management and the cooperation of sensor catalogues cannot be expressed using SensorML syntax.

B. Semantic Sensor Network Ontology

The following section summarizes differences between StarFL and SSNO. SSNO aims to apply semantic web technologies to the Sensor Web, i.e., to align SensorML, O&M, and lately even foundational ontologies. It describes how parts of a sensing system fit in a sensing domain. The alignment of the SSN ontology with the ultra light version of the DOLCE foundational ontology makes it hard to reconcile SSNO with O&M, which follows a more conventional approach observation descriptions in physical and natural sciences. Though the SSNO could be seen as the main ancestor of StarFL, changes have been performed in both semantic interpretation and conceptual design to foster ease of use, tool support, and general usability.

1) Interface implementation to O&M

The OGC and ISO standard O&M, currently published in version 2.0, represents the observation data model and encoding in the SWE suite of standards. SWE standards such as SOS, SPS make use of it. To fit into SWE, a sensor metadata language shall ensure two aspects: First the seamless implementation of the OM:Observation interface, and second the consistent usage of terms and concepts to facilitate semantic reasoning. SSNO interprets an Observation contrary to O&M by aligning it as a sub-concept of a DUL:Situation. O&M defines an Observation as an *act* of observing a property or phenomenon, a specialized event whose result is a data value [6]. Though the working group outlines that these two concepts perform a close match, a DUL:Situation and a DUL:Event are modelled as disjoint entities. Merged by force, problems regarding the semantic interpretations of elements can result. Not only the core concept of the observation defined differently in the SSNO, but other elements can only be linked through *closeMatch* relations:

SSNO expression		**O&M expression**
Sensor	⇔	observationProcedure
SensorOutput	⇔	observationResult
ObservationResult	⇔	result
ObservationValue	⇔	observationResult

[2] http://mmisw.org/smlmor

Thus a seamless implementation of O&M is not given. A mapping between O&M and SSNO has to be performed before semantic tools can operate on any concept. As opposed to this, StarFL performs a clean alignment to O&M. Its definition of SensingProcess is a sub concept of the observation procedure described in O&M. A SensingProcess produces observations and with an according observation result. These concepts are not modelled in StarFL but adopted from O&M. Like in SensorML, a *Sensing* element, which represents a procedure/process, can be linked embedding the XLink technique in the procedure tag of an O&M observation instance.

2) *Sensor and Sensing*

The core concept of SSNO, the relation between Sensor, Sensing and the linkage to the real world phenomena (observed property), is not solved satisfactory from a practical point of view. Thus, relations between core elements have been changed in StarFL. StarFL limits the definition of a sensor to a physical device, because they are the primary source in (semi-) automated Sensor Webs. To provide a solution for really all types of sensors, SSNO is far more abstract and describes sensors to be physical devices, computational methods, a laboratory setup with a person following a method, or any other thing that can follow a sensing method to observe a property.

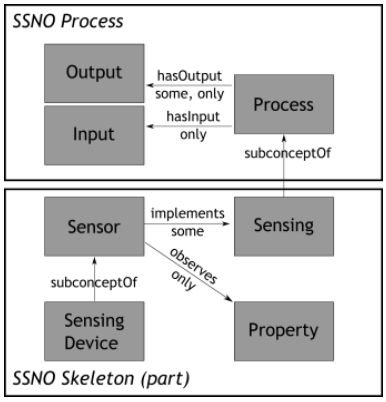

Fig 8
SSNO Sensor and Sensing

In SSNO, the *observes*-relation between sensor and property effects the definition of a sensing device (Fig 8). Following that approach, components of a device, which are responsible to derive a certain property, are summarized to a single sensor/physical device. For each property that a system observes the components are rearranged and thus represent a different sensor. As outlined before in section IV.A.3), an easier and more intuitive way might be the one-to-one relation between a physical device and its sensor model and to encapsulate the sensing process in an additional class sensing. A sensor model orientated approach as performed in StarFL is not expressible in SSNO but more intuitive and allows for searching for certain instruments and not for sensor types. Due to their dependencies on observed properties a StarFL *Sensing* element has a close match to an SSNO *Sensor*. Hence, for a mapping between both models the StarFL Sensing element would accordingly be annotated.

The StarFL *SensingProcedure* is interpreted similar to SSNO:Sensing, which is a sub-concept of DUL:Process and has one input and several outputs. Changes were necessary due to the different interpretation of process and device part in StarFL and to the support of sensing chains. The self-aggregation pattern of StarFL:SensingProcedure makes it possible to describe procedure chains. The *SensingProcedures* can be seen as pre-procedures of other *SensingProcedures* that perform a *basedOn* relation to them. In difference to SSNO the StarFL SensingProcedure has just one output but might have more than one input, which themselves are the outputs of its pre-procedure. The output of a procedure is a result for its observed property. A StarFL procedure therefore can reuse one or more outputs from previous procedures. A process that computes rain duration might base on a process for time measurement and water volume measurement.

3) *Additional Ontologies*

By reusing ideas from SensorML, StarFL covers more elements, such as the direct integration of spatial and temporal values or calibration information, whereas SSNO must include secondary vocabularies, which involves the risk of interoperability issues due to misaligned or contradicting external secondary ontologies.

In summary StarFL has adopted some concepts of SSNO such as measurement capabilities or the interpretation of platform. However, it uses a more straightforward and intuitive method to describe physical sensor devices. A modeller does not have to know exactly which physical part of a sensor is responsible for the derivation of a certain property. That information might be not accessible sometimes. However, SSNO has the goal to align components in a sensor domain. StarFL follows a more practical approach, with the primary goal to derive and qualify metadata for observation interpretation and sensor discovery processes.

V. APPLICATION

To demonstrate the usage of StarFL we apply it to the *ifgicopter* sensor platform [22]. The ifgicopter is an unmanned aerial vehicle (UAV) enhanced with a standalone computing board to manage the interaction with additional sensors. In our use case the ifgicopter is equipped with a Sensirion SHT75 sensor to observe temperature and humidity. A request for that specific sensor model in a catalogue would return a StarFL document (Listing 1). The root element SensorCharacteristic especially contains or references its provided two SensingProcedures.

```
<sfl:SensorCharacteristic gml:id="SensorChar_Sensirion_SHT75">
  <sfl:manufacturer><…>Sensirion AG<…> </sfl:manufacturer>
  <sfl:model>SHT75</sfl:model>
  <sfl:provides>
   <sfl:SensingProcedure
        xlink:href=”http://my.catalogue.net/SHT75/relHum/”>
  <sfl:SensingProcedure
        xlink:href=”http://my.catalogue.net/SHT75/temp/”>
  </sfl:provides>
</sfl:SensorCharacteristic>
```

Listing 1

The *SensingProcedures* humidity and temperature derive measurement results for their specific phenomena. They are described further by attributes such as their qualifying measurement capabilities. Listing 2 exemplarily demonstrates the encoding of accuracy.

```
<!—Humidity sensing procedure of Sensirion SHT75 -->
<sfl:SensingProcedure gml:id=" SHT75_relHum ">
  <sfl:observedProperty
            xlink:href="http://prop.ont.net#relativeHumidity"/>
  <sfl:unitOfMeasure uom="%"/>
  <sfl:qualifiedBy>
    <sfl:MeasurementCapability gml:id="humidityMeasurementCaps">
      <sfl:accuracy>
        <sfl:ComplexConditionalValue> <!-- Humidity  0 - 10 % RH -->
          <sfl:condition>
            <swe:QuantityRange>
              <swe:uom code="%"/> <swe:value>0 10</swe:value>
            </swe:QuantityRange>
          </sfl:condition>
          <sfl:value>
            <swe:QuantityRange>
              <swe:uom code="%"/> <swe:value>-4 4</swe:value>
            </swe:QuantityRange>
          </sfl:value>
        </sfl:accuracy><…>
      </sfl:MeasurementCapability>
    </sfl:qualifiedBy>
  <sfl:implementedBy  xlink:href =”http://my.catalogue.net/SHT75"/>
</sfl:SensingProcedure>
```

Listing 2

If the real humidity is between 0 and 10%, the accuracy of the provided values has a maximal derivation of ±4%. Other conditions and measurement capabilities, such as resolution or drift are left out for formatting reasons but work similar to the given example. SweCommon 2.0 does not provide a clear syntax for encoding conditions. Therefore StarFL defines a class *ComplexCodtitionalValue* to express value dependencies.

```
<sfl:Platform gml:id="ifgicopter">
  <sfl:operator><…>SWSL ifgi<…></sfl:operator>
  <sfl:mobile>true</sfl:mobile>
  <sfl:deployedAt>
    <sfl:PlatformDeployment gml:id="platformDeployment">
      <sfl:validTime><…>
        <swe:value>2011-03-25T10:10 2011-03-25T10:20</swe:value>
      </…></sfl:validTime>
      <sfl:deployer> <…>SWSL ifgi<…></sfl:deployer>
      <sfl:deploymentLocation>  <gml:Point>
          <gml:coordinates cs="WGS84">52.0 7.0 30 </gml:coordinates>
        </gml:Point> </sfl:deploymentLocation>
      <sfl:operationArea>   <gml:Polygon><…>
          <gml:coordinates cs="WGS84">51.9 6.9 52.1 7.1</gml:coordinates>
        </…> </gml:Polygon> </sfl:operationArea>
      <sfl:siteCharacteristic>
        <swe:Text>
          <swe:value>Located in moderate climate zone. </swe:value>
        </swe:Text>
      </sfl:siteCharacteristic>
    </sfl:PlatformDeployment>
  </sfl:deployedAt>
  <!-- ============ deployed sensors ========== -->
  <sfl:deployedSensor xlink:href=”http://my.catalogue.net/Sens_Depl01/>
</sfl:Platform>
```

Listing 3

StarFL interprets the ifgicopter to be a *Platform.* The platform is set up with a *PlatformDeployment* and references all attached sensors. In this particular use case the mobile ifgicopter platform is deployed at a point location for ten minutes, as illustrated in Listing 3. The operational area is set up in context of a WGS84 system. The valid time attribute expresses the time interval wherein the platform is operating. In a next step the sensor deployment respectively its sensor is described. Note that only the most basic attributes such as temporal and spatial data have to be considered. A detailed StarFL document therefore needs much less lines of code in comparison to SensorML to describe a sensor platform. The core connections are shown in Listing 4, where the sfl:characteristics attribute and the sfl:sensingProcedure link to their corresponding Static descriptions. The encapsulation of these attributes provides the opportunity to easily re-mount sensors on platforms and therewith decreases the effort for updating a sensor document.

```
<sfl:SensorDeployment gml:id="sd01">
  <sfl:validTime>
    <swe:TimeRange>
      <swe:uom code="ISO8601"/>
      <swe:value>2010-03-01T10:10 2011-06-30T10:00</swe:value>
    </swe:TimeRange>
  </sfl:validTime>
  <sfl:deployer xlink:href="#SWSL"/>
  <sfl:mobile>false</sfl:mobile>
  <sfl:deploys>
    <sfl:Sensor gml:id="SHT75_1111">
      <sfl:serialNumber>1111</sfl:serialNumber>
      <sfl:characteristics xlink:href =”http://my.catalogue.net/SHT75"/>
      <sfl:senses>
        <sfl:Sensing gml:id="hum_SHT75_1111">
        <sfl:sensingProcedure
                    xlink:href=”http://my.catalogue.net/SHT75/relHum/”>
        <sfl:unitOfMeasure uom="%"/>
        <sfl:active>true</sfl:active>
      </sfl:Sensing>
    </sfl:senses>
    <sfl:senses …><!--Temperature analogue to relative humidity -->
  </sfl:deploys>
</sfl:SensorDeployment>
```

Listing 4

VI. Conclusion and Outlook

In this paper the concept of StarFL – an alternative sensor metadata language that reuses concepts of both SensorML and the Semantic Sensor Network Ontology – has been described. Elements and patterns of SensorML and SSNO are applied where possible, adjusted and restricted, where necessary. StarFL focuses on sensor devices and does aim to support the majority of all sensor models. These restrictions lead to a more interoperable model, which may act as a proposal for a detailed SensorML profile leading to a straightforward creation of sensor metadata descriptions. Limitations of freedom may be overcome by using the extension point pattern of StarFL. Further, through its Static Module, StarFL introduces a mechanism for reusability that can easily be applied. Sensors are designed from an intuitive point of view without shifting the borders from real world sensing devices to software components or models. This allows users to store metadata of

their sensors easily and facilitates the creation of applications that make use of semantically aware sensor metadata descriptions, such as our mediation framework for sensor plug & play [17] or the RESTful web service for providing linked sensor data [23].

Further modifications may be necessary to optimize usability and interoperability aspects. The work on StarFL is still ongoing as minor issues have been reported by the user community and are as yet not satisfactorily resolved. For example the question of relative spatial connection of a sensor to a platform with an according definition of a relative reference system remains open. At the time tools are implemented to manage both the Static and the Dynamic Module. As the development of SensorML 2.0 is still in progress at the OGC, we will bring the concepts of StarFL into the discussion by aiming at influencing the development of the new SensorML version or even demonstrating an alternative solution for describing sensor metadata.

ACKNOWLEDGEMENTS

This research was financially supported by the project Flexible and Efficient Integration of Sensors and Sensor Web Services funded by the ERDF program for NRW, contract no. N 114/2008 and the European Commission as part of the integrated project 'Emergency Support System' (ESS), contract number no. 217951. Initial development of StarFL was funded by the Australian Government through the Intelligent Island Program and CSIRO. The Intelligent Island Program is administered by the Tasmania Department of Economic Development, Tourism and the Arts.

REFERENCES

[1] A. Bröring, J. Echterhoff, S. Jirka, I. Simonis, T. Everding, C. Stasch, S. Liang, and R. Lemmens, "New Generation Sensor Web Enablement," *sensors*, vol. 11, no. 3, pp. 2652–2699, 2011.

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

[3] S. Jirka and A. Bröring, *OGC Discussion Paper 09-033 - SensorML Profile for Discovery*. Open Geospatial Consortium, 2009. [Online]. Available: https://-portal.opengeospatial.org/files/?artifact_id=33284&version=2

[4] S. Jirka, A. Bröring, and C. Stasch, "Discovery Mechanisms for the Sensor Web," *Sensors*, vol. 9, 2009.

[5] L. Lefort, C. Henson, and K. Taylor, Eds., *Semantic Sensor Network XG Final Report*, ser. W3C Incubator Group Report. W3C, 2011.

[6] S. Cox, *OGC Implementation Standard 10-025: Observations and Measurements - XML Implementation, Version 2.0.0*. Open Geospatial Consortium, 2010.

[7] S. Avancha, C. Patel, and A. Joshi, "Ontology-driven adaptive sensor networks," in *First Annual International Conference on Mobile and Ubiquitous Systems, Networking and Services*. Citeseer, 2004, pp. 194–202.

[8] M. Eid, R. Liscano, and A. El Saddik, "A universal ontology for sensor networks data," in *Computational Intelligence for Measurement Systems and Applications, 2007. CIMSA 2007. IEEE International Conference on*. Ieee, 2007, pp. 59–62.

[9] C. Stasch, K. Janowicz, A. Bröring, I. Reis, and W. Kuhn, "A Stimulus-Centric Algebraic Approach to Sensors and Observations," in *3rd International Conference on Geosensor Networks, GSN 2009*, ser. Lecture Notes in Computer Science, N. Trigoni, A. Markham, and S. Nawaz, Eds., vol. 5659. Oxford, UK: Springer, July 2009, pp. 169–179.

[10] A. Devaraju, H. Neuhaus, K. Janowicz, and M. Compton, "Combining Process and Sensor Ontologies to Support Geo-Sensor Data Retrieval," in *6th International Conference on Geographic Information Science (GIScience 2010), Zurich, CH 14-17th September, 2010*, 2010; forthcoming.

[11] L. Bermudez, J. Graybeal, and R. Arko, "A Marine Platforms Ontology: Experiences and Lessons," in *Workshop on Semantic Sensor Networks (SSN 2006), In conjunction with 5th International Semantic Web Conference (ISWC 2006)*. Athens, GA, USA: http://www.ict.csiro.au/ssn06/, November 2006.

[12] D. Russomanno, C. Kothari, and O. Thomas, "Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models," in *The 2005 International Conference on Artificial Intelligence (IC-AI 2005)*. Las Vegas, NV, USA: CSREA Press, June 2005, pp. 637–643.

[13] A. Pease, I. Niles, and J. Li, "The suggested upper merged ontology: A large ontology for the semantic web and its applications," in *Working Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web*, vol. 28. Ed2 monton, Canada, 2002.

[14] M. Compton, C. Henson, H. Neuhaus, L. Lefort, and A. Sheth, "A survey of the semantic specification of sensors," in *Proceedings of the 2nd International Workshop on Semantic Sensor Networks (SSN09)*, T. K., A. Ayyagari, and D. De Roure, Eds., vol. Vol-522. CEUR, 2009, pp. 17–32.

[15] H. Neuhaus and M. Compton, "The semantic sensor network ontology," in *AGILE Workshop on Challenges in Geospatial Data Harmonisation, Hannover, Germany*, 2009, p. 33.

[16] K. Janowicz and M. Compton, "The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration into the Semantic Sensor Network Ontology," in *3rd International Workshop on Semantic Sensor Networks 2010 (SSN10), In conjunction with the 9th International Semantic Web Conference (ISWC 2010)*, K. Taylor, A. Ayyagari, and D. D. Roure, Eds., vol. 668. Shanghai,China: CEUR, November 2010.

[17] A. Bröring, P. Maué, K. Janowicz, D. Nüst, and C. Malewski, "Semantically-enabled sensor plug & play for the sensor web," *Sensors*, vol. 11, no. 8, pp. 7568–7605, 2011. [Online]. Available: http://www.mdpi.com/1424-8220/11/8/-7568/

[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: http://webmgs2010.como.polimi.it, August 2010.

[19] K. Lee, “IEEE 1451: A Standard in Support of Smart Transducer Networking,” in *17th Instrumentation and Measurement Technology Conference*, vol. 2. IEEE, 2000, pp. 525 – 528.

[20] S. DeRose, E. Maler, D. Orchard, and B. Trafford, “Xml linking language (xlink),” *Working Draft WD-xlink-20000221, World Wide Web Consortium (W3C), Feb*, 2000.

[21] C. Bizer, T. Heath, and T. Berners-Lee, “Linked Data - The Story so far,” *Journal on Semantic Web and Information Systems*, vol. 5, no. 3, pp. 1–22, 2009.

[22] M. Rieke, T. Foerster, and A. Bröring, “Unmanned Aerial Vehicles as Mobile Multi-sensor Platforms,” in *The 14th AGILE International Conference on Geographic Information Science. 18.-21. April 2011. Utrecht, Netherlands.*, 2011.

[23] K. Janowicz, A. Bröring, C. Stasch, S. Schade, T. Everding, and A. Llaves, “A restful proxy and data model for linked sensor data,” *International Journal of Digital Earth (IJDE)*, 2011.