Sensor metadata blueprints and computer-aided editing for disciplined SensorML

Paolo Tagliolato¹ ² ³, Alessandro Oggioni¹, Cristiano Fugazza¹, Monica Pepe¹, and Paola Carrara¹

¹ Institute for Electromagnetic Sensing of the Environment, National Research Council (IREA-CNR), via Corti 12, 20133 Milan, Italy
² Institute of Marine Science, National Research Council (ISMAR-CNR), Tesa 104 Arsenale, Castello 2737/F, 30122 Venice, Italy
³ LifeWatch Italy
E-mail: {tagliolato.p, oggioni.a, fugazza.c, pepe.m, carrara.p}@irea.cnr.it

Abstract. The need for continuous, accurate, and comprehensive environmental knowledge has led to an increase in sensor observation systems and networks. The Sensor Web Enablement (SWE) initiative has been promoted by the Open Geospatial Consortium (OGC) to foster interoperability among sensor systems. The provision of metadata according to the prescribed SensorML schema is a key component for achieving this and nevertheless availability of correct and exhaustive metadata cannot be taken for granted. On the one hand, it is awkward for users to provide sensor metadata because of the lack in user-oriented, dedicated tools. On the other, the specification of invariant information for a given sensor category or model (e.g., observed properties and units of measurement, manufacturer information, etc.), can be labor- and time-consuming. Moreover, the provision of these details is error prone and subjective, i.e., may differ greatly across distinct descriptions for the same system.

We provide a user-friendly, template-driven metadata authoring tool composed of a backend web service and an HTML5/javascript client. This results in a form-based user interface that conceals the high complexity of the underlying format. This tool also allows for plugging in external data sources providing authoritative definitions for the aforementioned invariant information. Leveraging these functionalities, we compiled a set of SensorML profiles, that is, sensor metadata blueprints allowing end users to focus only on the metadata items that are related to their specific deployment. The natural extension of this scenario is the involvement of end users and sensor manufacturers in the crowd-sourced evolution of this collection of prototypes. We describe the components and workflow of our framework for computer-aided management of sensor metadata.

1. Introduction
The need for continuous and comprehensive environmental knowledge has led to an increase in sensor resources, deployed both in space and on earth. The Variety of these systems and the Velocity of their technical evolution impose accurate knowledge on their characteristics, that is, sensor metadata. The accumulation of sensor data, collecting observations continuously in time, provides the third “V” characterizing Big Data (Volume). In order to discover, integrate, and exploit sensor data (and also to preserve them for future needs), we need to accurately record sensor information: Otherwise, overall usability of sensor networks is hampered.
To cope with these issues and to foster interoperability among sensor systems, the Open Geospatial Consortium (OGC) promoted the Sensor Web Enablement (SWE) initiative. The provision of metadata according to the prescribed SensorML schema is a key component for achieving this and nevertheless availability of correct and exhaustive metadata cannot be taken for granted. The reason why sensor description through SensorML has not yet reached widespread adoption in the Earth Observation community is manifold. As an example, the inherent complexity of SensorML, together with the scarce availability of dedicated tools for the editing of metadata, makes the provision of accurate sensor descriptions a challenging task. Moreover, the flexibility of this data format allows for multiple, equivalent descriptions of a sensor. This inhibits the emergence of reference descriptions that can be shared by the maintainers of a given sensor model (we are going to further elaborate on this in the next Section).

RITMARE[^1], a Flagship Project by the Italian Ministry of University and Research, aims at the integration of the widespread resources of Italian marine research, comprising both traditional geographic information and sensor data. Sub-project 7 (SP7), coordinated by us, is charged of building the Spatial Data Infrastructure (SDI) to be employed by the data providers involved in RITMARE - public research bodies and inter-university consortia - as well as by variety of stakeholders - public administrations, private companies, and citizens. In order to meet the project’s requirements as regards metadata creation, SP7 developed EDI[^2], a template-driven metadata authoring tool that can be easily customized to any XML-based metadata format and to a specific workgroup, institute, or project. The tool is made available as Free Open Source Software (FOSS). For establishing the RITMARE SDI, we customized the tool to ease authoring of SensorML v1.0.1 and v2.0.0 and integrated it with the software used by data providers to populate the SDI. As by-product of these activities, we created blueprints for specific sensor models, encoding all invariant information (e.g., the details on the manufacturer) in easy-to-use web forms that can be customized by individual maintainers.

2. Related work
The Sensor Model Language (SensorML)[^3] has been adopted by OGC for developing the SWE framework. SensorML provides a definition language that supports all details of sensors and sensor-to-platform constellations. We already mentioned that authoring SensorML descriptions is typically regarded as tedious and error-prone. This is partially due to the scarcity of authoring tools expressively tailored to SensorML. The SWE Software webpage[^4] on the OGC website provides some links to SensorML editors. The active ones are: Pines SensorML Editor[^5] and SensorML Process Editor[^6]. Both are freely-available and written in Java; unfortunately, both tools are currently not being developed and, anyway, are based on SensorML v1.0.1. Consequently, current practice entails the usage of general purpose XML editing tools which can be awkward to users unfamiliar with XML coding.

Other issues have been highlighted in literature as possible limitations to widespread adoption of SensorML, even by experienced users. Firstly, SensorML is, on purpose, an extremely flexible schema: While this characteristic allows for representing a very broad range of sensors, it elicits multiple descriptions for the same sensor[^7]. This relates not only to the concept of “soft-typing”[^8] but also to the XML structure of documents[^9] (i.e., same sensor, different XML
trees). In order to address these issues, the Earth Observation community recurs to SensorML profiles.

Profiles specify additional constraints on the schema they refer to, thus providing less flexibility to be a mandatory (structure constraint) or provide ranges for parameter values (narrowing soft-typing). Profiles may be associated with tools, like validators (e.g. schematron rules) as in [7], but their definition does not require such kind of facilities. In this case, the conformance of documents to the profile must be checked manually. A system, to ease metadata creation, should allow providers to annotate, for a sensor instance, solely the parts of their pertinence (e.g. just the serial number, not the model number; just the responsible party, not the manufacturer name; just the calibration events, not the physical characteristics).

In order to clearly exemplify the extent to which, for the aforementioned issues, SensorML complexity is hampering development of sensor networks, we pinpoint the OGC Sensor Observation Service 2.0 Hydrology Profile, a standard adopted as a Best Practice by OGC in 2014 to enhance interoperability among hydrological data providers. This profile regards the provision of SensorML descriptions, a key component in Sensor Observation Service standard [8], as an optional feature, thus expressing algorithms, sensor types, and time series types only, but not the physical device, the sensor instance. The solution we present in the next Section, other than overcoming complexity of SensorML editing to metadata authors, allows for distinguishing between sensor types and sensor instances, a missing feature in SensorML that partly motivated the emergence of alternative formats.

Specifically, Malewski et al. discusses a different conceptual schema for sensor descriptions, namely the Starfish Fungus Modeling Language (StarFL) [4], which “aims to describe more sensors with the least complexity”. StarFL features the structural distinction between types (SensorCharacteristic defined in the StarFL static module) and instances (Sensor defined in the dynamic module) of sensors. In principle, the distinct modules could be managed as separate catalogs. Also, in the context of the W3C Sensor Semantic Network Ontology (SSNO), in [10], Compton and coauthors highlight, besides other aspects linked to leveraging the Web Ontology Language (OWL) as a schema definition language (following the definitions in [12]), that the clear distinction between classes and instances in SSNO should be considered an advantage over SensorML. The authors insightfully suggest the possible integration of StarFL with SensorML (v2.0.0) as a semantically annotated profile, functional to a translation of SensorML into SSNO data structures.

We, among others [13, 5], believe that there is a strong need for a corpus of best practices for SensorML encoding, together with appropriate schema-specific tools for authoring metadata. This is even more important for SensorML because of its role in OGC standards and SDIs.

3. A framework for computer-aided management of sensor metadata

The challenge tackled by project RITMARE includes providing user interfaces to easily and directly manage, publish, and share marine data and metadata with no specific technical skills. As regards metadata editing, SP7 developed the tool described in the next Section, we then implemented the necessary customizations to assist users in the management of sensor life cycle, comprising an editor for disciplined authoring of SensorML.

3.1. Sensor metadata editor

In establishing the SDI for project RITMARE, we had to ease provision of metadata for datasets (according to the Italian profile to INSPIRE metadata) and for sensors (as SensorML descriptions). Thus we developed an open-source, user-friendly editor, EDI[14], which is

---

It should be noted that, since version 2.0.0, SensorML implements some mechanisms in this direction.
composed of a backend web service[8] and an HTML5/Javascript client[9] EDI is a template-driven metadata authoring tool that is capable of abstracting from the specific XML schema a given metadata format is complying with. The only input required by EDI to assist composition of metadata records in a given XML format is an appropriate template. The template language introduces the primitives that are necessary for the definition of metadata elements, intended as high-level abstractions of the information that shall be provided by the user. For the time being, we already authored templates for the several metadata formats.[10]

**Listing 1.** Portion of the EDI template for OGC Lightweight SOS Profile written in SensorML v2.0.0

```xml
<template>
  <settings>
    <metadataEndpoint>http://...</metadataEndpoint>
    <sparqlEndpoint>http://...</sparqlEndpoint>
  </settings>
  <datasources>
    <datasource>
      <id>keywords</id>
      <query>'!<![CDATA[SELECT ?c ?l ?a ...]]></query>
    </datasource>
  </datasources>
  <group>
    <id>output</id>
    <label xml:lang="en">Output signals</label>
    <element>
      <id>output_param</id>
      <label xml:lang="en">Outputs</label>
      <help xml:lang="en">The outputs ... (ogc:11-169r1)</help>
      <isMandatory>true</isMandatory>
      <isMultiple>true</isMultiple>
      <hasRoot>/sml:PhysicalSystem/sml:outputs/sml:OutputList</hasRoot>
      <produces>
        <item>
          <hasPath>sml:output/@name</hasPath>
          <hasDatatype>autoCompletion</hasDatatype>
        </item>
        <item>
          <hasPath>sml:output/swe:Quantity/@definition</hasPath>
          <hasDatatype>autoCompletion</hasDatatype>
        </item>
      </produces>
      <help xml:lang="en">Start typing the first...</help>
    </element>
  </group>
</template>
```

[8] EDI-NG Server: [https://github.com/SP7-RITMARE/EDI-NG_SERVER](https://github.com/SP7-RITMARE/EDI-NG_SERVER)
[9] EDI-NG Client: [https://github.com/SP7-Ritmare/EDI-NG_client](https://github.com/SP7-Ritmare/EDI-NG_client)
[10] At the moment on our GitHub repositories the following templates are available: the INSPIRE profile to ISO 19115; RNDT (Repertorio Nazionale dei Dati Territoriali - national repository of territorial data: [http://www.rndt.gov.it](http://www.rndt.gov.it)), the Italian profile to the former; SensorML (v1.0.1 and v2.0.0); the NcML format for annotating NetCDF deployments on THREDDS servers.
Listing 1 shows an excerpt of the template for SensorML v2.0.0 and allows to pinpoint the main characteristics of this (meta)language. The outer template tag contains a settings section that defines the general parameters that are taken into account for creating the HTML editing frontend: Particularly, the metadataEndpoint and sparqlEndpoint tags contain, respectively, the URL of the web service processing the client input and the default SPARQL [15]. The next essential component of a template is the definition of datasources, that is, the specification of where to look up when the editor fills drop-down lists, provides alternatives in autocompletion features, etc.

The template then contains a sequence of group tags that allow the developer, as the name suggests, to group metadata elements together, a feature that can be employed to divide the editing interface into sections or tabs. Group tags contain a number of element tags that represent metadata fields, even when these are composite entities made up by a number of distinct items. Each of these three contracts can be given multilingual labels in order to tailor the interface to multiple languages (and automatically switch between them).

Each element tag specifies whether the metadata element is Mandatory or is Multiple. Also, tag hasRoot specifies at which level of the XML node tree multiple instances of the same element shall be rooted. Finally, item tags represent all the nodes that are required in order to fully define the metadata item (ISO metadata is particularly redundant in the specification of these). A key component in the specification of items is the associated data type, specified by means of the hasDatatype tag.

For the time being, it is possible to represent traditional data types ("string", "text", "int", "real", and "date") and two string data types complying with a specific format ("URI", "URN"). Then, three data types draw values from context information stored as RDF data [10]. The “code” data type creates a dropdown list (in the language chosen by the user for metadata) from the terms defined in a SKOS [17] thesaurus. The “autoCompletion” data type also takes values from a SKOS thesaurus but generates a text input with autocompletion capabilities.

The above description of the capabilities of EDI is necessarily partial for the sake of conciseness. Still it should be apparent that the expressiveness of the underlying templating language is sufficient to rendering most state-of-the-art metadata encodings. More specifically, the typical requirements imposed by a profile can be easily programmed: Line 21 in listing 1 shows, as an example, the code for implementing one of the conformance requirements defined in [6] ("sml:outputs (mandatory)") and Figure 1 shows the corresponding portion of the generated interface. This should cover the main desideratum in Section 2, that is, the availability of dedicated, user-friendly tools for authoring profiled SensorML.

3.2. Sensor metadata blueprints

As discussed earlier in section 2, the possibility of separating the descriptions of sensor types and instances has been pointed out as a way to enhance operative work. Both the aforementioned alternatives to SensorML, StarFL and SSNO, realise such separation by defining classes and

---

11 We use here the popular version of "tag" to indicate an XML element, in order to avoid confusion with the element element defined in our template language.
instances (StarFL with its static and dynamic modules [4], SSNO by using A-Box and T-Box of a knowledge base [10]). Let us discuss this point to better clarify our solution.

In Object Oriented languages the class/instance separation is the main stream approach, but not the only one. Object prototypes have been proposed in literature[12] as an alternative paradigm. Henry Lieberman discussed the two approaches, relating them to the philosophical controversy between “representing general concepts as abstract sets or classes and representing concepts as concrete prototypes”, the second being an approach that “corresponds more closely to the way people seem to acquire knowledge from concrete situations” [18]. Learning by example is the “prototype” strategy, and even an actual practice in the community of SensorML users. Borning propose [19] different flavours of prototype-based languages, considering various mechanisms for creating new objects from existing ones. Duplication with no reference is the simplest possibility, but constraining new objects to their “prototype” with well-defined rules is another. He even contemplates the possibility to have classes in this system, to group objects with shared characteristics. Advantages of this point of view in object oriented programming is out of the scope of our discussion, but the consideration of prototypes is, in our opinion, appropriate.

As mentioned, the current practice in defining sensor metadata with SensorML is by examples, or, as StarFL authors suggest, by “snippets” of code. StarFL design, they say, could better support such a practice, allowing references to specific documents in the static module instead of copying snippets. Something similar to this idea, even if not affecting the structure of the conceptual model, has been incorporated within SensorML v2.0.0, where inheritance mechanisms (sml:typeOf) have been implemented. There is perhaps some difference, because it is not possible

Moreover, in current practice, in Javascript/ECMAScript, one of the key technologies of the world wide web, the prototype concept is a central characteristic.
to define a SensorML document as the representative of a type of sensor. 52°North, in order to support SensorML v2.0.0 inheritance in its current SOS implementation, extended SensorML introducing an element for marking a sensor document when it must be considered a “type” (i.e. a sensor in an abstract way: not a physical instance). Theoretically speaking, this distinction could be avoided even in an Object Oriented Programming paradigm, considering the broader realm of prototype-orientation. We can interpret the “typeOf” of SensorML v2.0.0 as a sort of derivation of a new object by reference to an existing one. The originating object can be thought of as an object and not as a class. In particular, tracing a parallelism with the static module of StarFL, this prototypical object can have the peculiarity of being invariant (or with some invariant part, in the case of SensorML). We could call such objects “blueprints”. And, in effect, this kind of objects are of great usefulness. They can be defined by a community of expert users, or better the manufacturers themselves, reporting all fine grained characteristics of a model.

Lieberman [18] contrasts inheritance and delegation as the mechanisms supporting sharing knowledge in object-oriented systems, respectively, in the class-oriented and in the prototype-oriented approaches. To share knowledge through inheritance, a class is needed for containing commonalities among its instances. Delegation doesn’t need the definition of a class to share knowledge among objects. Any object can be a prototype for others. Derived objects can simply refer to their prototypes when in need for some information they do not directly own. We should consider, in this sense, the SensorML “typeOf” and StarFL reference among static and dynamic modules, examples of delegation mechanism, with the difference that in StarFL delegation can only be applied to invariant parts of Sensors.

SensorML v1.0.1 does not present such inheritance, or delegation. Sharing knowledge among objects, that is, sharing metadata among distinct SensorML documents, is just not contemplated, but it happen by the act of copying a document and using it as a prototype. No more reference, perhaps, can be encoded within the copied document. A management system could eventually store this kind of meta-information. Even in this case, prototyping and copying can be exploited to enhance interoperability. Once a collection of SensorML documents is considered as reference for a community developing on the same examples, the reference knowledge is shared, and this enhance omogeneity. Moreover, if derivation from prototypes is constrained (as proposed in [19]) by “blueprinting” some parts of the original documents, with mechanisms shared by the community (i.e. management systems), a better support to consistence among information is given.

In this line we developed a web-based system, involving the EDI software, able to provide users with a catalog of prototypes to choose from, when in need for creating metadata of a new sensor instance. After the choice of a sensor model (its prototype), the user is redirected to the editor SensorML 1.0.1 web-form, pre-compiled with all the available prototype information and, according to the defined derivation constraints, blocking the blueprinted parts. We developed an initial catalog of 10 prototypes of SensorML documents for the most used sensor models deployed by the marine community involved in the RITMARE project. We profiled SensorML, requiring the identifier element and constraining its range to URIs following a naming convention that let such identifiers to be dereferenced. This way each SensorML document created following this profile rule in RITMARE can be retrieved in the distributed infrastructure by its identifier/URL. Each prototype, following itself this profile rule, can be uniquely identified and retrieved. This means that it is technically straightforward to use any SensorML document within the infrastructure as a prototype. The strategy for the enrichment of the catalog is to use the forthcoming SensorML documents of instances of new sensor models, not yet represented.

13 We discuss this point and some technical aspects of the URI dereferencing strategy in a discussion in the INSPIRE thematic cluster, available at https://themes.jrc.ec.europa.eu/discussion/view/21204/inspire-and-referencing
developed by users of the infrastructure by the use of the web-interface provided by EDI. Currently the catalog of prototypes is managed by administrators. An idea for a natural future evolution of the system, is the introduction of a consensus mechanism, fostering a bottom-up approach to the definition of reference SensorML documents.

4. Conclusions and future work

In this paper, we propose our solution to the issues hampering widespread adoption of SensorML as a sensor description metadata schema. Specifically, we developed a template-driven, form-based editing tool, EDI, that is capable of easing editing of records in any XML-based schema. By itself, such tool allows for a number of user-oriented functionalities that speed up metadata editing and assure compliance with the specific schema that is considered. Moreover, the tool can be used to encode specific profiles for a given schema; a feature that is advisable for SensorML descriptions of individual sensors. Consequently, we built a number of sensor metadata blueprints that relieve developers of the burden of providing invariant information about a specific sensor, thus allowing her to focus on the aspects concerning the specific deployment and dynamic information. This software is made available as FOSS. The solution we presented is yielding an increase in quality and quantity of the available sensor descriptions in the RITMARE community, encompassing most of Italian marine research, and representing most of Italian observation systems in this domain. For this reason the metadata blueprints created by now are mainly related to sensor in the marine environment. Nonetheless the easiness to create new blueprints of SensorML prototype, by editing existing ones, or simply by using the authoring tool (EDI) could hopefully trigger a crowd-sourced provision and validation of blueprints. The collection of crowd-sourced blueprints may straightforwardly build consensus on a broad range of sensor types, belonging to a variety of application domains.

Acknowledgments

The activities described in this paper have been funded by the Italian Flagship Project RITMARE and LifeWatch Italy.

References

[1] Fugazza C, Basoni A, Menegon S and Oggioni A 2014 Procedia computer science 33 261–265 ISSN 1877-0509 URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84904462808&partnerID=q2rcBxpzhttp://dspacecris.eurocris.org/jspui/handle/123456789/213
[2] Botts M and Robin A 2007 OpenGIS Sensor Model Language (SensorML) Implementation Specification OGC 07-000. Open Geospatial Consortium URL http://portal.opengeospatial.org/files/?artifact_id=21273
[3] Simonis I and Echterhoff J 2008 GEOSS and the sensor web - GEOSS Sensor Web Workshop Report GEOSS task DA 07-04 (Geneva, Switzerland)
[4] Malewski C, Simonis I, Terhorst A and Bröring A 2012 International Journal of Digital Earth 7 450–469 ISSN 1753-8947 URL http://dx.doi.org/10.1080/17538947.2012.739663
[5] Hu C, Guan Q, Chen N, Li J, Zhong X and Han Y 2014 Remote Sensing 6 10546–10570 ISSN 2072-4292 URL http://www.mdpi.com/2072-4292/6/11/10546/
[6] Jirka S, Stasch C and Bröring A 2014 OGC® Best Practice for Sensor Web Enablement Lightweight SOS Profile for Stationary In-Situ Sensors - OGC 11-169r1. Open Geospatial Consortium
[7] Jirka S and Broering A 2009 OGC® Discussion Paper 09-033 - SensorML Profile for Discovery. Open Geospatial Consortium
[8] Bröring A, Stasch C and Echterhoff J 2012 Sensor Observation Service Interface Standard, Version 2.0. - OGC 12-006
[9] Volker A, Jirka S and Utech M 2014 OGC® Sensor Observation Service 2.0 Hydrology Profile URL http://docs.opengeospatial.org/bp/14-004r1/14-004r1.html#27
[10] Compton M, Barnaghi P, Bermúdez L, García-Castro H, Corcho O, Cox S, Graybeal J, Hauswirth M, Henson C, Herzog A, Huang V, Janowicz K, Kelsey W D, Le Phuoc D, Lefort L, Leggieri M, Neuhaus H, Nikolov A, Page K, Passant A, Sheth A and Taylor K 2012 Web Semantics: Science, Services and Agents on
the World Wide Web 17 25–32 ISSN 15708268 URL http://www.sciencedirect.com/science/article/pii/S1570826812000571

[11] McGuinness D L and Van Harmelen F 2004 OWL Web Ontology Language Overview - W3C Recommendation 10 February 2004 URL http://www.w3.org/TR/owl-features/

[12] Haslhofer B and Klas W 2010 ACM Computing Surveys 42 1–37 ISSN 03600300 URL http://dl.acm.org/citation.cfm?id=1667062.1667064

[13] Sutter R D, Wainscott S B, Boetsch J R, Palmer C J and Rugg D J 2015 Wildlife Society Bulletin 1–13 ISSN 19385463 URL http://doi.wiley.com/10.1002/wsb.548

[14] Basoni A, Bastianini M, Fugazza C, Menegon S, Minuzzo T, Oggioni A, Pavesi F, Pepe M, Sarretta A, Tagiolato P and Carrara P 2014 Fostering bottom-up capacity in managing and sharing marine observations: the RITMARE StarterKit Proceedings of EuroGOOS 2014

[15] Harris S and Seaborne A 2013 SPARQL 1.1 Query Language - W3C Recommendation 21 March 2013 URL http://www.w3.org/TR/sparql11-query/

[16] Cyganiak R, Wood D and Lanthaler M 2014 RDF 1.1 Concepts and Abstract Syntax - W3C Recommendation 25 February 2014 URL http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

[17] Miles A and Bechhofer S 2009 SKOS Simple Knowledge Organization System Reference - W3C Recommendation 18 August 2009 URL http://www.w3.org/TR/skos-reference/

[18] Lieberman H 1986 SIGPLAN Not. 21 214–223 ISSN 0362-1340 URL http://doi.acm.org/10.1145/960112.28718

[19] Borning A H 1986 Classes versus prototypes in object-oriented languages Proceedings of 1986 ACM Fall Joint Computer Conference ACM ’86 (Los Alamitos, CA, USA: IEEE Computer Society Press) pp 36–40 ISBN 0-8186-4743-4 URL http://dl.acm.org/citation.cfm?id=324493.324538