Pavesi, F et al 2016 EDI – A template-driven metadata editor for research data. *Journal of Open Research Software,* 4: e40, DOI: http://dx.doi.org/10.5334/jors.106

SOFTWARE METAPAPER

EDI – A Template-Driven Metadata Editor for Research Data

Fabio Pavesi[1], Anna Basoni[1], Cristiano Fugazza[1], Stefano Menegon[2], Alessandro Oggioni[1], Monica Pepe[1], Paolo Tagliolato[2] and Paola Carrara[1]

[1] IREA-CNR, IT

[2] ISMAR-CNR, IT

Corresponding author: Fabio Pavesi (fabio@adamassoft.it)

EDI is a general purpose, template-driven metadata editor for creating XML-based descriptions. Originally aimed at defining rich and standard metadata for geospatial resources, It can be easily customised in order to comply with a broad range of schemata and domains.

EDI creates HTML5 [9] metadata forms with advanced assisted editing capabilities and compiles them into XML files. The examples included in the distribution implement profiles of the ISO 19139 standard for geographic information [14], such as core INSPIRE metadata [10], as well as the OGC [8] standard for sensor description, SensorML [11].

Templates (the blueprints for a specific metadata format) drive form behaviour by element data types and provide advanced features like codelists[1] underlying combo boxes or autocompletion functionalities.

Virtually, the editing of any metadata format can be supported by creating a specific template.

EDI is stored on GitHub at https://github.com/SP7-Ritmare/EDI-NG_client and https://github.com/SP7-Ritmare/EDI-NG_server.

Keywords: form generator, metadata editing, INSPIRE, OGC, semantic enrichment, XML

Funding Statement: EDI has been developed within the Italian flagship Project RITMARE funded by the Italian Ministry of University and Research (http://www.ritmare.it), in the activities of the SubProject 7 (creation of the interoperable spatial data infrastructure of Italian marine research).

(1) Overview

Introduction

The effective provisioning of spatial resources on the Internet has always been hampered by the intrinsic characteristics of this category of data: They are either of non-textual nature or, even when utilizing a text-based format for their encoding, generally convey little information on their semantics, purpose, and associated principals. For this reason, metadata is typically associated with resources in order to ground search, retrieval, and ultimately reuse of spatial information. Also, once instances of this category of resources are successfully "discovered" through the ad-hoc search engines generally referred to as geoportals, compliance with a number of access protocols is required by user agents in order to actually exploit the associated data.

The management of geospatial information in Europe is regulated by the INSPIRE (INfrastructure for SPatial InfoRmation in Europe) Directive [2], which itself is based on the groundwork set by the ISO 19115 ("Geographic information – Metadata) and ISO 19119 (Geographic information – Services) standards. Unfortunately, these standards can easily become inadequate because of the emergence of categories of data not previously considered. As an example, spatial data (in the broadest sense) have been recently enriched by the new category of data sources constituted by real-time/near-real-time data pulled from sensors. Consequently, these new data sources require specific data and metadata representations for management and enactment.

In this context, metadata editing plays a pivotal role that requires a flexible strategy. EDI, is a template-based metadata editor that is capable of abstracting from the specific XML schema a given metadata format is complying with. Also, the service-based deployment strategy allows users to retain full control on the data structures that are produced. Further details on EDI's role can be found in [1,3]

EDI's main goals are depicted in **Figure 1** and listed here:

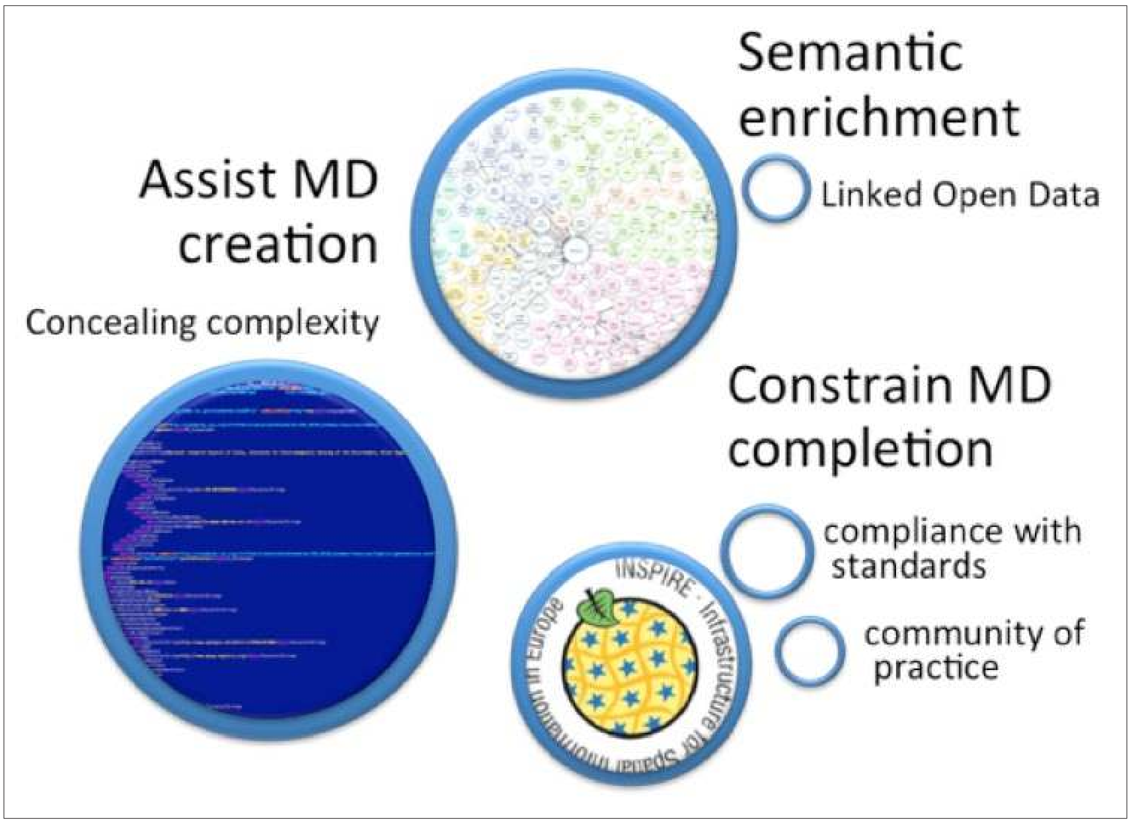

Figure 1: EDI's goals.

- Assisting metadata creation
  - by concealing complexity, e.g. storing mandatory default data inside hidden fields, driving users in fields filling, etc.
  - by helping avoid typos, e.g. by providing combo boxes and autocompletion functionalities
- Fostering semantic enrichment
  - With an eye to the Semantic Web, EDI allows plugging in external data sources that are made available as SPARQL[2] endpoints. On the basis of these, EDI can semantically annotate metadata by providing, beside text descriptions of the entities that are referred to in the metadata (e.g., keywords, points of contact, toponyms, etc.), unique identifiers for these, in the form of URIs[3]
- Constraining metadata completion
  - by validating field types and mandatory field presence on the client-side
  - by facilitating input with dedicated widgets; As an example, the geographic extent of a dataset (the "bounding box") can be entered either by drawing a rectangle on a map or by filling in the coordinates

EDI allows system administrators to easily associate codelists, controlled vocabularies, and any kind of context information that are specific to their domain with the application. This capability is supported by allowing the editing interface to draw information from generic RDF data structures [5] made available as SPARQL endpoints.

EDI was developed as part of sub-project 7 (http://www.ritmare.it/en/articolazione/sottoprogetto-7.html) of the Italian Flagship Project RITMARE (http://www.ritmare.it/).

Creating a form based on a specific template is as easy as inserting the following line inside a script tag:

```
edi.loadLocalTemplate("INSPIRE_dataset", "1.00",
onTemplateLoaded);
```

Templates are described by XML files, in terms of elements (groups of controls somehow related to one another) and items (single controls).

Items

- are associated with data types to enable validation,
- can have a display directive (a URL can be displayed as a single text box or as a text box with underlying picture preview)
- can have a datasource

Implementation and architecture

EDI consists of a client- and a server-side components.

The relationship between them is shown in **Figure 2**.

The EDI client gathers input from the user.

It then packs the information provided by the user into an intermediate data structure (referred to as EDIML) and sends it to the EDI server to be compiled into the specific metadata (XML) format the template refers to.

The server sends the XML metadata back to the client for further usage (e.g. storage, CSW[4] sharing, ...).

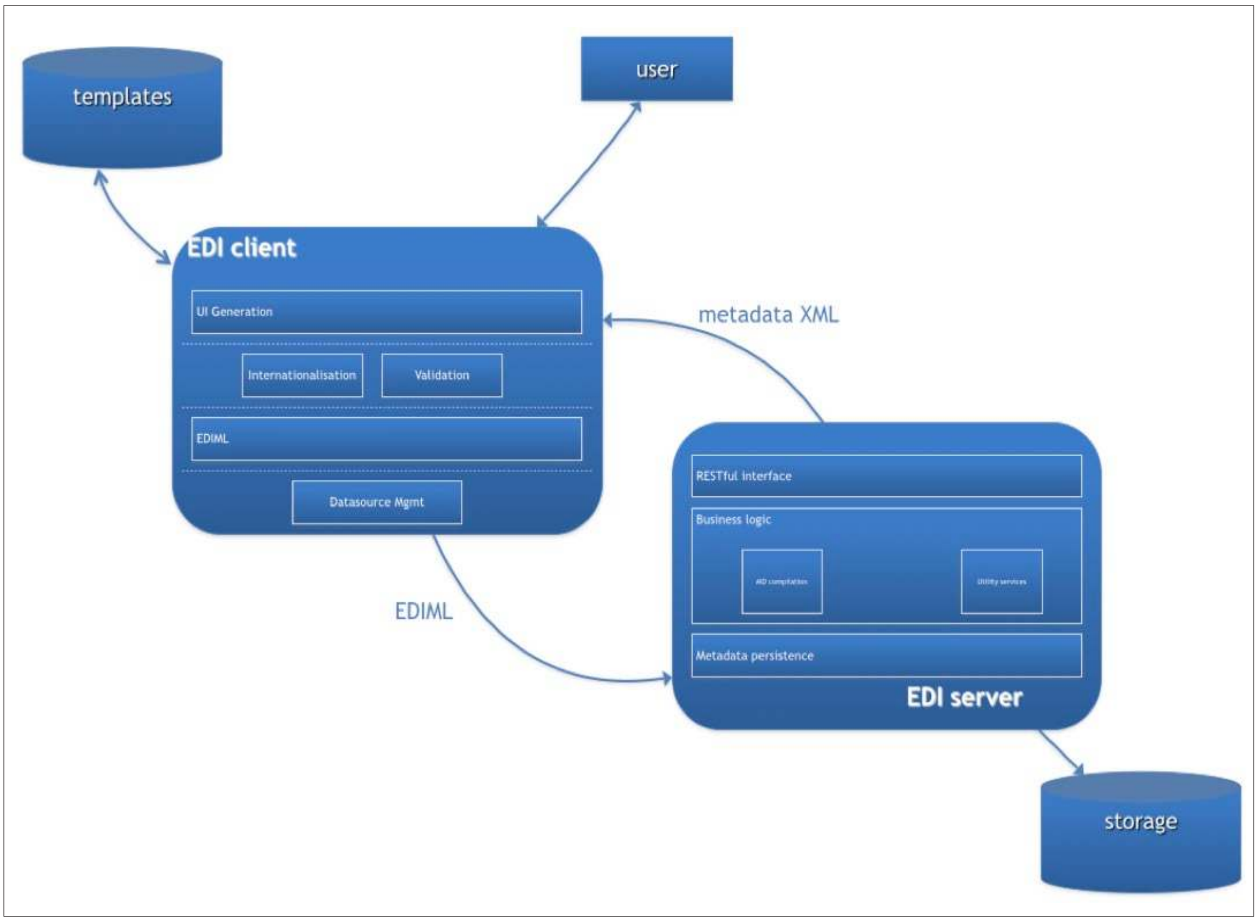

Figure 2: Architecture.

The client

The client is written in Javascript and is based on the jQuery (https://jquery.com) and Bootstrap 3 (http://getbootstrap.com) frameworks.

Its main components are:

- **UI Generation module** – it renders HTML5 controls according to the data type and constraints defined in the template.
- **Internationalisation module** – in templates, all labels and help boxes can be localised by specifying them in an arbitrary number of languages, each being declared by an xml:lang attribute.
- **Validation module** – on the basis of the data types and constraints specified in the template, form content is validated upon submission, returning a "pass", "warning" or "error" state. Validation can be disabled at template-level for testing purposes.
- **EDIML module** – this component hosts the main data structure, i.e. *ediml.content*, and methods to maintain it. This is also the module managing conversations with the EDI server.
- **Datasource Mgmt module** – manages the data sources underlying codelists or autocompletion functionalities.

The server

The server is written in Java as a Spring[5] Boot Application. The server consists of:

- **A RESTful[7] interface** – that the client uses to compile metadata and access utility services.
- **Business logic** – it consists of two main blocks:
  - **Metadata compilation** – this is the actual core of the server: it transforms an EDIML XML document containing the data and structure of the user filled-in form into the target XML metadata. Compilation can be followed by a template-defined chain of XSLT transformations if further processing is required.
  - **Utility services** – various services, e.g. XSLT transformations, EDIML retrieval, etc.
- **Metadata persistence** – metadata generated by the server is stored in a PostgreSQL relational database.

Listings 1–4 show excerpts of the template for SensorML v2.0.0 and allow to pinpoint the main characteristics of the EDI 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 front-end: Particularly, the ***metadataEndpoint*** and ***sparqlEndpoint*** tags contain, respectively, the URL of the web service

```
<settings>
  <userInterfaceLanguage xml:lang="it"/>
  <metadataLanguage selectionItem="ling_md_1"/>
  <metadataEndpoint>http://sp7.irea.cnr.it/tomcat2/edi/</metadataEndpoint>
  <sparqlEndpoint>http://sparql.get-it.it</sparqlEndpoint>
  <requiresValidation>false</requiresValidation>
  <baseDocument><![CDATA[
    <sml:PhysicalSystem xmlns:gml="http://www.opengis.net/gml/3.2"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:gco="http://www.isotc211.org/2005/gco"
    xmlns:gmd="http://www.isotc211.org/2005/gmd"
    xmlns:sml="http://www.opengis.net/sensorml/2.0"
    xmlns:swe="http://www.opengis.net/swe/2.0"
    xmlns:gsr="http://www.isotc211.org/2005/gsr"
    xmlns:gts="http://www.isotc211.org/2005/gts"
    xmlns:gss="http://www.isotc211.org/2005/gss"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:swes="http://www.opengis.net/swes/2.0"
    xsi:schemaLocation="http://www.opengis.net/sensorml/2.0
http://www.utm.csic.es/SensorWeb/Schemas/sensorML/2.0/sensorML.xsd">
      </sml:PhysicalSystem>
    ]]>
    </baseDocument>
    <xsltChain>
      <xslt>http://sp7.irea.cnr.it/jboss/MDService/rest/SensorML20.xsl?version=1.00</xslt>
    </xsltChain>
</settings>
```

Listing 1: SensorML template excerpt – (Settings).

```
<endpointTypes>
  <endpointType xml:id="virtuoso" method="GET" queryParameter="query">
    <parameters>
      <parameter name="format" value="application/sparql-results+json"/>
      <parameter name="save" value="display"/>
      <parameter name="fname" value="undefined"/>
    </parameters>
  </endpointType>
</endpointTypes>
```

Listing 2: Sensorml template excerpt – (endpointtypes).

processing the client input and the default SPARQL endpoint for the ***datasources*** below. The ***baseDocument*** tag contains the outermost levels of the document's XML hierarchy that is shared by all descriptions that are created with a given template. 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 for the autocompletion features, etc. For convenience, these can also be clustered into ***endpointTypes*** in order to avoid duplication of parameters that are shared by all instances of a specific triple store (i.e., the data base for RDF).

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 ***item***s. Each of these three constructs 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 ***isMandatory*** or ***isMultiple***. 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*** attribute.

EDI templates define the primitives necessary for the definition of metadata elements.

```
<datasources>
  <codelistxml:id="languages"endpointType="virtuoso">
    <uri>http://inspire-registry.jrc.ec.europa.eu/registers/Languages/items</uri>
    <url>http://sp7.irea.cnr.it:8891/sparql</url>
  </codelist>
  <sparqlxml:id="keywords"endpointType="virtuoso">
    <query><![CDATA[
      PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
      SELECT ?c ?l ?a
      FROM <http://ritmare.it/rdfdata/parameters>
      WHERE {
      ?c rdf:typeskos:Concept.
      OPTIONAL {
      ?c skos:prefLabel ?l.
      FILTER( LANG(?l) = "en")
      }
      OPTIONAL {
      ?c skos:prefLabel ?a.
      FILTER( LANG(?a) = "it")
      }
      FILTER( REGEX( STR(?l), "$search_param", "i") || REGEX( STR(?a), "$search_param", "i") )
      }
      ORDER BY ASC(?l) ASC(?a)
]]></query>
    </sparql>
    <sparqlxml:id="keywordsS"endpointType="virtuoso">
      <query><![CDATA[
        PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
        SELECT ?c ?l ?a
        FROM <http://ritmare.it/rdfdata/parameters>
        WHERE {
        ?c rdf:typeskos:Concept.
        OPTIONAL {
        ?c skos:prefLabel ?l.
        FILTER( LANG(?l) = "en")
        }
        OPTIONAL {
        ?c skos:prefLabel ?a.
        FILTER( LANG(?a) = "it")
        }
        FILTER( REGEX( STR(?c), "$search_param", "i") || REGEX( STR(?a), "$search_param", "i") )
        }
        ORDER BY ASC(?l) ASC(?a)
      ]]>
      </query>
    </sparql>
    <singletonxml:id="manufacturers"endpointType="virtuoso"triggerItem="">
      <query><![CDATA[
        PREFIX ns: <http://www.w3.org/2006/vcard/ns#>
        SELECT ?c ?l
        FROM <http://ritmare.it/rdfdata/manufacturers>
        WHERE {
        ?c rdf:typefoaf:Organization.
        ?c foaf:name ?l.
        FILTER( REGEX( STR(?l), "$search_param", "i") )
        }
        ORDER BY ASC(?l)
      ]]>
      </query>
    </singleton>
  </datasources>
```

Listing 3: Sensorml template excerpt – (datasources).

```
    <group xml:id="system_ids">
      <label xml:lang="it">Identificazione</label>
      <label xml:lang="en">Identification of the system</label>
      <element xml:id="manuf_name" isMandatory="false" isMultiple="false">
        <label xml:lang="it">Denominazione del produttore</label>
        <label xml:lang="en">Manufacturer name</label>
        <help xml:lang="it">Indicare la denominazione del produttore del sensore, del sistema
di sensori o della piattaforma. Iniziare a digitare tre lettere per ottenere dei suggerimenti
da un elenco precompilato di produttori. </help>
        <help xml:lang="en">Name of the manufacturer of the sensor, system or platform. Start
typing the first three letters to obtain suggested autocompletion from a manufacturer list.</help>
        <hasRoot>/sml:PhysicalSystem/sml:identification/sml:IdentifierList</hasRoot>
        <produces>
          <item hasIndex="1" outIndex="" isFixed="true" hasDatatype="string">
             <hasPath>sml:identifier/sml:Term/sml:label</hasPath>
             <hasValue>Manufacturer Name</hasValue>
          </item>
          <item hasIndex="2" outIndex="" isFixed="true" hasDatatype="string">
             <hasPath>sml:identifier/sml:Term/@definition</hasPath>
             <hasValue>urn:ogc:def:identifier:OGC:manufacturerName</hasValue>
          </item>
          <item hasIndex="3" outIndex="" isFixed="false" hasDatatype="autoCompletion"
             datasource="manufacturers">
             <hasPath>sml:identifier/sml:Term/sml:value</hasPath>
          </item>
        </produces>
      </element>
    </group>
```

Listing 4: Sensorml template excerpt – (groups, elements and items).

```
       <element xml:id="nome_std" isMandatory="true" isMultiple="false">
        <label xml:lang="en">Name of standard</label>
        <label xml:lang="it">Nome dello standard</label>
        <hasRoot>/gmd:MD_Metadata/gmd:metadataStandardName</hasRoot>
        <produces>
         <item hasIndex="1" xml:id="nome_std_1" isFixed="true" hasDatatype="string">

<hasPath>/gmd:MD_Metadata/gmd:metadataStandardName/gco:CharacterString</hasPath>
          <hasValue>DM - Regole tecniche RNDT</hasValue>
         </item>
        </produces>
       </element>
```

Listing 5: Hidden predefined item example.

```
       <element xml:id="data_md" isMandatory="true" isMultiple="false">
        <label xml:lang="en">Metadata date</label>
        <label xml:lang="it">Data dei metadati</label>
        <help xml:lang="en">The date metadata were created, in the yyyy-mm-dd format.</help>
        <help xml:lang="it">Data di compilazione dei metadati, assume come valore di default la
data corrente nel formato yyyy-mm-dd.</help>
        <hasRoot>/gmd:MD_Metadata/gmd:dateStamp</hasRoot>
        <produces>
         <item hasIndex="1" xml:id="data_md_1" isFixed="false" hasDatatype="date">
           <hasPath>/gmd:MD_Metadata/gmd:dateStamp/gco:Date</hasPath>
           <defaultValue>$TODAY$</defaultValue>
         </item>
        </produces>
       </element>
```

Listing 6: Default value example.

Simple	Composite	External sources	Internal references
string	dateRange	code (alias: codelist)	copy
text	boundingBox	autoCompletion	ref
date	URL	select	function
int			
real			
autonumber			
hidden			
URN			
URI			
sensorID (URL)			

Table 1: Item data types.

```
<elements>
   <ediVersion>2</ediVersion>
   <version>1.00</version>
   <template>SensorML20_lightweight</template>
   <xsltChain/>
   ...
   <element>
      <id>manuf_name</id>
      <root>&#x2F;sml:PhysicalSystem&#x2F;sml:identification&#x2F;sml:IdentifierList</root>
      <mandatory>NA</mandatory>
      <represents_element>manuf_name</represents_element>
      <items>

         <item>
           <id>manuf_name_3</id>
           <elementId>manuf_name</elementId>
           <path>sml:identifier&#x2F;sml:Term&#x2F;sml:value</path>
           <datatype>autoCompletion</datatype>
           <fixed>false</fixed>
           <useCode/>
           <useURN/>
           <outIndex/>
           <value>DeepSea Power &amp;amp; Light</value>
           <labelValue>DeepSea Power &amp;amp; Light</labelValue>

<codeValue>http:&#x2F;&#x2F;sp7.irea.cnr.it&#x2F;rdfdata&#x2F;sensors&#x2F;manufacturer&#x2F
;31</codeValue>
           <urnValue/>

<languageNeutral>http:&#x2F;&#x2F;sp7.irea.cnr.it&#x2F;rdfdata&#x2F;sensors&#x2F;manufacturer
&#x2F;31</languageNeutral>
           <listeningFor>#manuf_name_3</listeningFor>
           <isLanguageNeutral/>
           <datasource>manufacturers</datasource>
           <hasIndex>3</hasIndex>
           <field/>
           <itemId/>
           <show/>
           <defaultValue/>
           <query/>
         </item>
       </items>
       <alternativeTo/>
     </element>
     ...
</elements>
```

Listing 7: Sample ediml.

Each element defines one or more items that correspond to the individual XML nodes that are populated in the metadata record.

The essential information defining an element is:

- the element must or must not show in the interface (***isFixed***, see **Listing 5**)
- textual explanation in the interface (***label*** and ***help***)
- compulsoriness (***isMandatory***)
- the XPath defining the position where the XML node shall be placed (***hasRoot***, ***hasPath***)
- its value (***hasValue, defaultValue***, see **Listing 6**)
- data types (***hasDatatype***, see **Listing 6**)

Elements contain items, which have a data type declared for validation purposes.

A list of available data types is shown in **Table 1**.

The interchange format – EDIML

EDIML is an XML format we devised to pass on user-entered metadata from the EDI client to the EDI server and backwards, without losing the semantic enrichment EDI is capable of when the destination format is semantics-unaware, as ISO standards, unfortunately, are.

This is the reason why EDI constantly keeps two versions of each metadata: one in EDIML, e.g. for subsequent editing, and one in the destination format (e.g. SensorML 2.00).

Listing 7 shows an example of the EDIML format. Please note that, for item manuf_name_3, the value is "DeepSea Power & Light", but the URI (i.e. "http://sp7.irea.cnr.it/rdfdata/sensors/manufacturer/31") is brought along.

Quality control

Functional testing was conducted by manually creating a collection of metadata, used as a test bench for validation.

Both Sensor ML (ver 1.0.0 and 2.0.0) and INSPIRE metadata are created by different experts of different disciplines (physical and chemical oceanography, marine geology, geophysics, coastal systems, marine ecology, fishery and aquaculture, marine biomolecular science, human impacts, climatology, biogeochemistry, remote sensing, agriculture) and for different type of data resources, in order to account for the huge variability of contents, lexicons, outlooks, as well as for the subjectivity, in the metadata filling. Several experts participating to research projects, mainly RITMARE, are involved in the testing; metadata are manually compiled using the template available in the standard distribution that refers to one specific metadata (XML) format. More than 60 samples of Sensor ML and more than 50 samples of INSPIRE metadata are compiled for the purpose.

The consistency was carried out for the whole metadata collection against the corresponding XML Schemas[6] (as declared in the ***baseDocument*** tag), by means of SensorML and ISO XSDs.

Quality has been further tested specifically for the INSPIRE profile by means of the official INSPIRE validator (available at http://inspire-geoportal.ec.europa.eu/validator2/), manually uploading the samples, and getting positive returns.

(2) Availability

Operating system

Any OS that can run Java 1.7 for the back end, any OS for the client

Programming language

Javascript[7] / JavaEE 1.7[8]

Additional system requirements

EDI Client is tested on Mozilla Firefox

Dependencies

Java 1.7 (EDI Server only)

PostgreSQL[9] 9.0+ (EDI Server only, tested on 9.3.4)

List of contributors

Pepe Monica, Oggioni Alessandro, Fugazza Cristiano, Tagliolato Paolo – template design, functional testing

Pavesi Fabio – software design and development

Menegon Stefano – functional testing, integration with the GET-IT platform

Software location

Archive (e.g. institutional repository, general repository) (required)

Name: GitHub

Persistent identifiers:

EDI Server: https://github.com/SP7-Ritmare/EDI-NG_server.git

Distribution (v1.2, commit ID b351a9d): https://github.com/SP7-Ritmare/EDI-NG_server/releases/download/v1.2/edi.zip

EDI Client: https://github.com/SP7-Ritmare/EDI-NG_client.gitDistribution (v 1.2, commit ID 2f4f05c): https://github.com/SP7-Ritmare/EDI-NG_client/releases/tag/v1.2

Licence: GPL v3

Publisher: Fabio Pavesi

Date published: 15/12/14

Language

English

(3) Reuse potential

EDI actually has great reuse potential.

In fact, it can generate documents according to any XML schema and then any data formats with an XML serialisation can be produced.

Plenty of templates are available in the EDI-NG_templates repository[10], which we have been using in several research projects, also serve as samples of what can be attained with templates.

For instance, we provide both SensorML 1.0.1 and SensorML 2.0

As an example, an EDI template[11] producing an RDF/XML output has been developed.

On the other hand, as EDI allows for specifying an arbitrary chain of XSL Transformations for post-processing the XML output, it can generate any text-based output format as well.

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

Competing Interests
The authors declare that they have no competing interests.

Notes

[1] a codelist is a flat terminology (as opposed to hierarchical ones, such as taxonomies) that is defined to indicate the possible values of a specific data item; as an example, the ISO standards for geographic information define a number of codelists for specifying the resource type (whose possible values are dataset, series, and service), the role of a point of contact (creator, custodian, distributor, etc.).
[2] SPARQL 1.1 Overview: http://www.w3.org/TR/sparql11-overview/ [4]
[3] Uniform Resource Identifier (URI): https://tools.ietf.org/html/rfc3986
[4] Catalog Service for the Web – http://www.opengeospatial.org/standards/cat
[5] Spring – http://spring.io
[6] XML Schema – http://www.w3.org/standards/xml/schema
[7] Javascript – http://www.w3.org/standards/webdesign/script
[8] JavaEE – http://www.oracle.com/technetwork/java/javaee/overview/index.html
[9] PostgreSQL – http://www.postgresql.org
[10] EDI-NG_templates repository – https://github.com/SP7-Ritmare/EDI-NG_templates.git
[11] RDF/XML example – https://github.com/SP7-Ritmare/EDI-NG_templates/blob/master/templates/FOAF_v2.00.xml

References

1. **Basoni, B, Bastianini, M, Fugazza, C, Menegon, S, Minuzzo, T, Oggioni, A, Pavesi, F, Sarretta, A, Tagliolato, P, Vianello, A,** and **Carrara, P,** 2014. EDI: Software per la metadatazione di risorse geografiche, dati osservativi e documenti, conformi RNDT e INSPIRE. Available at: https://github.com/SP7-Ritmare?query=EDI.
2. **European Commission** 2008. Commission Regulation (EC) No 1205/2008 of 3 December 2008 implementing Directive 2007/2/EC of the European Parliament and of the Council as regards metadata. *Official Journal of the European Union,* L 326(1205), pp.12–30. Available at: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2008:326:0012:0030:EN:PDF.
3. **Fugazza, C** et al., 2014. A Holistic, Semantics-aware Approach to Spatial Data Infrastructures. In Proceedings of 3rd International Conference on Data Management Technologies and Applications. SCITEPRESS – Science and Technology Publications, pp. 349–356. DOI: http://dx.doi.org/10.5220/0004997603490356. Available at: http://www.scitepress.org/DigitalLibrary/Link.aspx?.
4. **Harris, S** and **Seaborne, A** 2013 SPARQL 1.1 Overview. W3C recommendation, 21 March 2013. Available at: http://www.w3.org/TR/sparql11-query/
5. **W3C** 2014 RDF 1.1 Semantics – W3C Recommendation 25 February 2014. W3C\. Available at: http://www.w3.org/TR/rdf11-mt/.
6. **Fugazza**, **C** et al, 2014 The RITMARE Starter Kit: Bottom-up Capacity Building for Geospatial Data Providers. In Proceedings of the 9th International Conference on Software Paradigm Trends. pp. 169–176. Available at: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0004999801690176.
7. **Fielding, R T,** and **Taylor, R N,** 2000 Principled design of the modern Web architecture. Proceedings of the 2000 International Conference on Software Engineering. ICSE 2000 the New Millennium, 2(2), pp.115–150.
8. **Open Geospatial Consortium** http://www.opengeospatial.org/
9. **Hickson, I, Berjon, R, Faulkner, S, Leithead, T, Navara, E, O'Connor, E,** and **Pfeiffer, S,** 2014. HTML5, W3C Recommendation. REC-html5-20141028. Available at: <http://www.w3.org/TR/2014/REC-html5-20141028>.
10. **European Commission Joint Research Centre** 2013 INSPIRE Metadata Implementing Rules: Technical Guidelines based on EN ISO 19115 and EN ISO 19119, version 1.3. Available at: http://inspire.ec.europa.eu/documents/Metadata/MD_IR_and_ISO_20131029.pdf
11. **Open Geospatial Consortium** 2014 OGC® SensorML: Model and XML Encoding Standard, version 2.0.0. Available at: https://portal.opengeospatial.org/files/?artifact_id=55939
12. **International Organization for Standardization** 2003 ISO 19115 Geographic Information – Metadata, ISO/TC-211
13. **International Organization for Standardization** 2005 ISO 19119 Geographic information – Services, ISO/TC-211
14. **International Organization for Standardization** 2007 ISO/TS 19139:2007 Geographic information – Metadata – XML schema implementation. ISO. Available at: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=32557.

How to cite this article: Pavesi, F, Basoni, A, Fugazza, C, Menegon, S, Oggioni, A, Pepe, M, Tagliolato, P and Carrara, P 2016 EDI – A template-driven metadata editor for research data. *Journal of Open Research Software,* 4: e40, DOI: http://dx.doi.org/10.5334/jors.106

Submitted: 19 November 2015 **Accepted:** 27 September 2016 **Published:** 25 October 2016