WHITE PAPER

Flexible approach for semantic validation in the context of Internet of Things

Contact: philippe.cousin@eglobalmark.com

THE NEED FOR SEMANTIC VALIDATION

Internet of Things (IoT) is an emerging area that not only requires development of infrastructure but also deployment of new services capable of supporting multiple, scalable (cloud-based) and interoperable (multi-domain) applications. With the rapidly increasing amount of data from IoT, the interoperability problem has become one of the core research topics in order to release the full potential of the technology. Technical and syntactical interoperability has been the focus of research and development for last years, and also in the interest of most standards organizations, alliances and consortia. However, taking into account the very dynamic nature of the IoT context where technology evolves rapidly and thus data sources change formats frequently, this approach is far from being sufficient. This describes the necessity for IoT platforms to be interoperable at the data/event level so that it becomes easier to combine/aggregate data/event coming from heterogeneous data sources. The work that has been done on these challenges in relation to semantic-web intends to achieve the next interoperability level: the semantic one. Recent activities within the AIOTI[i] working group on Standardisation have confirmed that one of the most important topic is on the semantic Interoperability. Standard Development Organisations such as W3C, ETSI, oneM2M are spending a lot of resources to address this topic.

One of the core aspect to achieve semantic interoperability is the ontology and the data annotation using ontology. From years of research, some ontology development methodologies, best practices and guidelines[ii] have been proposed that help people to develop and use ontology from scratch or by adapting existing ontologies. The evaluation of such ontologies and annotated data is essential to determine if they are interoperable with other ontologies, especially the reference ontologies recognized by standardization bodies such as W3C-SSN ontology, ETSI-SAREF ontology and oneM2M base ontology.

Taking into account the nature of an ontology file and semantically annotated IoT data, we can distinguish two levels of validation:

1) syntactic checks including use of undefined properties and classes, poorly formed namespaces, problematic prefixes, literal syntax;
2) semantic checks including respect of OWL 2 specifications, logical issues like contradictory inferred result.

There are already some validation tools that deal with the above issues, for example[iii] Eyeball, HermiT that we present in the next subsection. Unfortunately, we notice that a complete tool kit covering the above ontology validation aspects with comprehensive validation report generation and corrective suggestion is still missing. Therefore, a tool is proposed with the following features:

1. Automatic validation chain that covers essential syntactic and semantic aspects
2. Easy configuration and usage
3. Integrated and comprehensive validation report

The design and the development of the presented ontology validator is a first initiative towards this market demand.

STATE OF THE ART

In the last decades a huge amount of research on ontology evaluation has been performed. As results, guidelines and best practice for ontology design are available, and several tools for helping users in the task of evaluating ontology have been proposed. But we can

notice that these tools suffer from one or more of the following weak points:

1. Need of desktop software installation and configuration. Some tools imply the user to install the ontology editor in which the validation tool is included as a plug-in. These plug-ins are often outdated due to the lack of maintenance resource for multiple reasons which may imply to some problems of incompatibility, for example, the tool needs a specific environment version which is different from the latest stable version of the editor. For example:
   a. **Fluent Editor** : is a tool for editing, manipulating and querying complex ontologies written in OWL, RDF or SWRL. It has tools that helps the user to manage complex ontologies (A reasonner, Protégé interoperability plugins).
   b. **Infovore**: is a Map/Reduce framework for processing RDF data based on the Hadoop Framework. Infovore can handle big data sets through on Hadoop clusters for example. Originally developed to process the Freebase dump, Infovore contains a number of tools including the Parallel Super Eyeball validator.
2. Web based technologies as MoKi [iv] that consists of a wiki-based ontology editor that incorporates ontology evaluation functionalities. This kind of technology forces an installation process too in other to set up the wiki system.
3. Command line tools. These tools do not have any friendly interface and their use is reserved to users with technological background. These include:
   a. **qSKOS**[v]: measures the quality of SKOS concept schemas according to their defined criteria. It can be also used as API in Java.
   b. **Eyeball** [vi] : is a Jena-based tool for checking RDF models (including OWL) for common problems
4. There is also another kind of validation tools which are the online tools. Unfortunately, most of these tools offer one single service which only provide a limited validation coverage of ontologies or RDF data validation. Some examples are:
   a. **OWL Validator**[vii]: an online tool validates OWL based ontologies. It inspects an input file comparing it with a list of potential problems. At the end of the process, a list of errors and recommendations is produced. The tool also identifies compilation problems.
   b. **RDF validator**[viii]: an online tool, based on a RDF Parser (ARP), to validate an RDF/XML document and should return meaningful error messages and warnings when it encounters erroneous or suspicious RDF/XML.
   c. **RDF Distiller** [ix] : an online tool to transform data between different RDF Formats (RDF, RDFa, Turtle, Trig, N-Triples, N-Quads).
   d. In addition, there are tools only for a specific ontology validation, which means the tool will check the ontology only for a precise vocabulary. As an example, **The GoodRelations Validator**[x] is a tool that helps you to check your product and company meta-data based on the GoodRelations Web Vocabulary for E-Commerce.

ONTOLOGY VALIDATION

The EGM Ontology and data validator is a web service integrated within a web-based client-server architecture. A simple client is proposed offering an easy and intuitive user interface to evaluate the service. Through the GUI, user is able to upload his ontology and annotated

linked data in order to be validated against a reference ontology such as the W3C SSN ontology. The validator detects syntactic and semantic issues if any, and produces a test report at the end of the process. As a part of the EGM Test-as-a-Service (TaaS) framework, the ontology validator share the same infrastructural elements.

Thanks to its web service approach associated with a GUI, usage of the tool is simplified so the user can focus on the validation. Nevertheless, beside its simplicity a range of functions have been integrated to increase versatility in both accepted input formats, validation functions and reporting.
Popular RDF serialization formats are accepted as input thanks to an integrate parser. These include N-Quads (.nq), N-Triples (.nt), N3 (.n3), Turtle (.ttl), TriG (.trig), TriX (.trix), RDF/JSON(.rj), JSON-LD (.jsonld), BinaryRDF (.brf), RDF/XML (.rdf), OWL (.owl). The resulting RDF is checked against RDF model and specifications[xi] before proceeding the ontology validation.

Then ontology validation takes place, adressing both syntactic and semantic validation.

Examples of checked attributes are:

Syntactic validation

- **Unknown properties and classes** with respect to the reference ontology: the predicate used in the ontology is not declared in any imported ontologies (i.e. the ontologies declared as prefix) or the reference .
- **Problematic prefix namespaces**: checks that the prefix declarations of the model have namespaces that are valid URIs and that if the prefix name is "well-known" (rdf, rdfs, owl, xsd, and dc) then the associated URI is the one usually associated with the prefix.
- **Ill-formed URIs and language tags on literals**: Checks literals for syntactically correct language codes, syntactically correct datatype URIs, and conformance of the lexical form of typed literals to their datatype.
- **Unexpected local names in schema namespaces**: Checks that every URI in the model is well-formed according to the rules of the Jena IRI library. May apply additional rules specified in the configuration file.
- **Untyped resources and literal**: Checks that all URI and literals resources in the model have an rdf:type property in the model or the schema(s).

Semantic validation

- **Inheritance relationships for classes and properties**: checks whether there exists a model of Ontology, that is, whether there exists a (relational) structure that satisfies all axioms in this ontology.
- **Subsumption of concepts**: determine whether concept C subsumes concept D, i.e., whether description of C is more general than the description of D.
- **Consistency of A with respect to B**: determine whether individuals in A do not violate descriptions and axioms described by B.
- **Other logical inferencable relationship**s according to OWL2 semantics

Finally, a best practice checks is conducted to provide recommendations to ontology developers so they avoid pitfalls in their ontology. This reduces the risk of issues during the reasoning phase, avoids maintainability issues and improves accessibility and clarity.

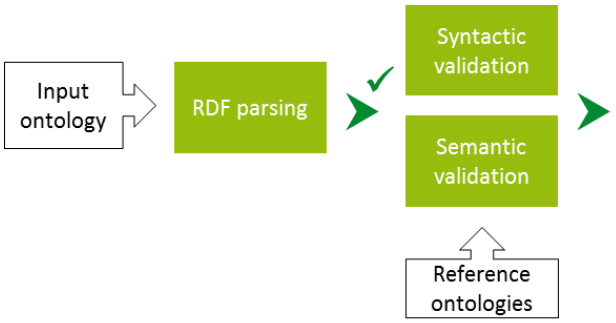

Figure 1: ontology validation process

Ontology validation is of interest when conducted in relation toward a reference ontology. A typical case would be to validate that a oneM2M device properly instantiates the oneM2M base ontology. However, ontologies are rarely used alone and generally refer to external ontologies, extending or restricting their scope. The EGM ontology validator analyses the input ontology and **automatically extends the list with relevant referent ontologies** proposed for the validation (see figure 2).

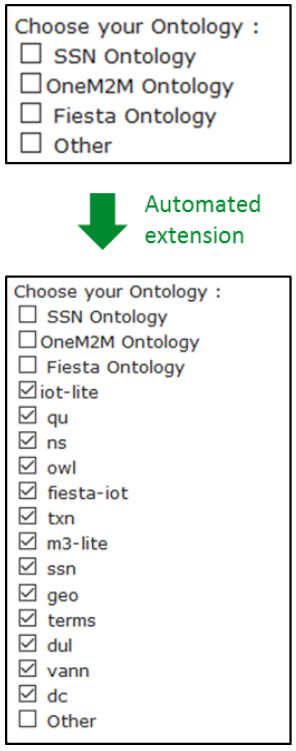

Figure 2: automated extension of relevant referent ontologies

TEST CASE

For illustration purpose, a GuruPlug_E24.owl file semantically describing an Ethernet enabled plug, and including a number of errors, has been used. Some of the error messages are reported and explained below.

Syntactic errors

```
On statement: :GuruPlug_E24_USB
rdf:type
DeviceModel:WiredCommunicationDevice
    class not declared in any schema:
DeviceModel:WiredCommunicationDevice
```

"`class not declared in any schema`" error is reported when the class used here is not declared in any imported ontologies (i.e. the ontologies declared as prefix) or the reference ontology (i.e. the SSN ontology in the current case)

```
On
statement: :GuruPlug_E24_Ethernet :ha
sConnector DeviceModel:RJ-45_Connector
    predicate not declared in any
schema: :hasConnector
On
statement: :GuruPlug_E24_Ethernet :av
ailableBandwidth test
    predicate not declared in any
schema: :availableBandwidth
```

"`predicate not declared in any schema`" error is reported when the predicate used here is not declared in any imported ontologies (i.e. the ontologies declared as prefix) or the reference ontology (i.e. the SSN ontology in the current case)

```
On
statement: :GuruPlug_E24_Ethernet :av
ailableBandwidth test
    eye:badLexicalForm: "test"
    eye:forDatatype: xsd:integer
```

"`eye:badLexicalForm`" error is reported when the actual literal data isn't conform to the declared datatype. In the present example, the .literal type xsd:Integer is expected while the data is a string ("text").

```
On statement:    :GuruPlug_E24_Computer
rdf:type DeviceModel:ComputingDevice
    eye:badURI:
```

"eye:badURI" error is reported when the URI form is not conform to the URI specification, or fails a user-specified spelling constraint. In the current example, the URI "http://wot.ccsr.ee.surrey.ac.uk/DeviceModel. owl#ComputingDevic e" isn't a legal URI because it contains a space.

```
bad           namespace           URI:
"http://www.w3.org/2000/01/rdf-sc#"
    on prefix: "rdfs"
    for reason: "non-standard namespace
for prefix"
```

"bad namespace URI" error is reported because the prefix "rdfs" is one of the "standard" prefixes that are taken from Jena's PrefixMapping.Extended, whereas the URI "http://www.w3.org/2000/01/rdf-sc#" bound isn't the usual one which is "http://www.w3.org/2000/01/rdf-schema#".

Semantic errors

The syntactic error detection part checks above problems regarding to the "standard" ontologies (rdfs, owl, etc.) and the reference ontology. The semantic error detection part will check other problems such as logical consistency in the current RDF data or ontology, in relation to all the ontologies referenced in the file rather than limiting itself to the reference one. Two illustrations of such issues are provided below.

```
==========    Axioms    causing    the
unsatisfiability 1: Temp1De ==========
SubClassOf(<http://www.semanticweb.org
/egm002/ontologies/2016/4/untitled-
ontology-46#Temp1De>
<http://www.semanticweb.org/egm002/ont
ologies/2016/4/untitled-ontology-
46#TempDeploy>)

SubClassOf(<http://www.semanticweb.org
/egm002/ontologies/2016/4/untitled-
ontology-46#TempDeploy>
<http://purl.oclc.org/NET/ssnx/ssn#Dep
loyment>)

SubClassOf(<http://www.semanticweb.org
/egm002/ontologies/2016/4/untitled-
ontology-46#Temp1De>
<http://www.semanticweb.org/egm002/ont
ologies/2016/4/untitled-ontology-
46#TempDevice>)

DisjointClasses(<http://purl.oclc.org/
NET/ssnx/ssn#Deployment>
<http://purl.oclc.org/NET/ssnx/ssn#Sen
singDevice>)

SubClassOf(<http://www.semanticweb.org
/egm002/ontologies/2016/4/untitled-
ontology-46#TempDevice>
<http://purl.oclc.org/NET/ssnx/ssn#Sen
singDevice>)
=====================================
====================
```

In this case we use the file Test.owl, having two disjoint classes (Deployment and SensingDevice) and third class (Temp1De) which is a subclass of the two last ones thus creating a logical error as two disjoint classes should not have any class in common.

```
========= Axioms causing the
inconsistency: 1 ==========
EquivalentClasses(<http://owl.man.ac.uk
/2006/07/sssw/people#man>
ObjectIntersectionOf(<http://owl.man.ac
.uk/2006/07/sssw/people#adult>
<http://owl.man.ac.uk/2006/07/sssw/peop
le#male>
<http://owl.man.ac.uk/2006/07/sssw/peop
le#person>) )

ClassAssertion(<http://owl.man.ac.uk/20
06/07/sssw/people#kid>
<http://owl.man.ac.uk/2006/07/sssw/peop
le#bob>)

ClassAssertion(<http://owl.man.ac.uk/20
06/07/sssw/people#man>
<http://owl.man.ac.uk/2006/07/sssw/peop
le#bob>)

EquivalentClasses(<http://owl.man.ac.uk
/2006/07/sssw/people#kid>
ObjectIntersectionOf(<http://owl.man.ac
.uk/2006/07/sssw/people#person>
<http://owl.man.ac.uk/2006/07/sssw/peop
le#young>) )

DisjointClasses(<http://owl.man.ac.uk/2
006/07/sssw/people#adult>
<http://owl.man.ac.uk/2006/07/sssw/peop
le#young>)
=======================================
==================
```

In this case we use the file People.owl with two classes having a common class on one side (adult → Person) and (young → Person) whereas on the other side they have a disjoint class (adult different from young), then we create an instance (Bob a person) which is person, young and adult class in the same time.

PERSPECTIVES

This paper provides an overview of the new challenges related to syntactical and semantical validation of ontologies in the context of Internet of Things. It provides an overview of the Easy Global Market validation tool functionalities which are able to validate RDF data and ontologies against referenced ontologies. This tool combines 3 main functionalities: lexical validation of XML/JSON format, syntactic validation of an ontology file and RDF data regarding to standard specifications (rdfs, owl, etc) and a given reference ontology, and semantic validation. Several improvements of the tool are under research.

Firstly, it will be integrated within the Easy Global Market tool chain for IoT providing users with a unified experience for testing of their IoT deployments and increase reliability, interoperability and security of such deployments. Second, increased support will be provided to developers by the addition of a recommendation system helping developers to understand the validation results and providing suggestions to repair the reported errors and make the ontology or RDF linked data valid. Thirdly, an extension toward the validation of annotated data within domains context is foreseen and efforts are being made to scale such approaches to align with Big Data and stream analytics requirements.

Finally, the automated matching and ontology alignment is under research to automate the process of determining correspondences between concepts in other to self-improve interoperability between different ontologies developed within the IoT domain.

[i] https://ec.europa.eu/digital-single-market/aioti-structure
[ii] "READY4SmartCities," [Online]. Available:http://www.ready4smartcities.eu/.
[iii] http://wiki.opensemanticframework.org/index.php/Ontology_Tools
[iv] https://moki.fbk.eu/website/index.php
[v] https://www.w3.org/2004/02/skos/
[vi] http://jena.sourceforge.net/Eyeball/
[vii] http://owl.cs.manchester.ac.uk/validator/
[viii] https://www.w3.org/RDF/Validator/
[ix] http://rdf.greggkellogg.net/distiller
[x] http://www.ebusiness-unibw.org/tools/goodrelations-validator/
[xi] https://www.w3.org/TR/REC-rdf-syntax/