HORIZONS 2020 PROGRAMME

Research and Innovation Action – FIRE Initiative

Call Identifier:	H2020–ICT–2014–1
Project Number:	643943
Project Acronym:	FIESTA-IoT
Project Title:	Federated Interoperable Semantic IoT/cloud Testbeds and Applications

Semantic Models for Testbeds, Interoperability and Mobility Support and Best Practices V2

Document Id:	FIESTAIoT-D312-161115-Draft
File Name:	FIESTAIoT-D312-161115-Draft.pdf
Document reference:	Deliverable 3.2
Version:	V02
Editor:	Rachit Agarwal/Nikolaos Georgantas/Valerie Issarny
Organisation:	Inria
Date:	Nov 15th 2016
Document type:	Deliverable
Dissemination level:	PU

DOCUMENT HISTORY

Rev.	Author(s)	Organisation(s)	Date	Comments
V01	Rachit Agarwal	Inria	2016/09/13	Initial version of the document
	Rachit Agarwal , Garvita Bajaj	Inria	2016/09/19	LinDA Transformation Tool
	Rachit Agarwal , Garvita Bajaj	Inria	2016/09/22	M3-lite updates, Ontology updates
V02	Rachit Agarwal	Inria	2016/10/11	TOC Updates
	David Gómez, Jorge Lanza	UNICAN	2016/10/25	SmartSantander Annotator
V03	Mengxuan Zhao	EGM	2016/10/28	EGM contributions
V04	Minwoo Ryu	KETI	2016/10/31	KETI Contributions
V05	Rachit Agarwal	Inria	2016/10/28 2016/10/30	Executive summary, Requirement sections, Related work section, Ontology Section, SoundCity Annotator Mobility Management Conclusion
	Tarek Elsaleh	UniS	2016/11/02	UniS Contributions
V06	Konstantinos Bountouris	Com4Innov	2016/11/02	Com4Innov Contributions
	David Gomez	UniCan	2016/11/02	Guidelines Section
V07	Rachit Agarwal	Inria	2016/11/02	Integrate Contributions
	Juan Echevarría	SDR	2016/11/4	QR
	Ronald Steinke	FOKUS	2016/11/10	TR1
V08	Mengxuan Zhao	EGM	2016/11/11	TR2
V09	Rachit Agarwal	Inria	2016/11/12	Address review comments, Generate version for submission
V10	Martin Serrano	NUIG	2016/11/13	Circulated for Approval
Draft	Martin Serrano	NUIG	2016/11/15	EC Submitted

Overview of Updates/Enhancements over D3.1.1

Section	Description
1	Re order 1. Add requirement analysis section and relation with architecture section.
2.1	Updates to the IoT-related ontologies section. Added SSN updates, oneM2M base ontology and SAREF ontology, other ontologies
2.2	Add LinDA Transformation tool
3.1	Aligned FIESTA-IoT ontology with oneM2M.
3.2	Updated FIESTA-IoT ontology, with necessary modifications.
3.3	Added new concepts in M3-lite.
3.4	Added persistent link to existing version of the FIESTA-IoT ontology
3.5	Added sample SPARQL queries
4.1	Add Annotation tools that are made available as a part of In-house Testbeds
4.2	Added FIESTA-IoT annotation validator

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TERMS AND ACRONYMS

Acronym	Definition
4G	Fourth Generation
5G	Fifth Generation
API	Application Programming Interface
AVT	Annotation Validation Tool
CSV	Comma Separated Values
DB	DataBase
DUL	Dolce+DnS Ultralite
EPC	Evolved Packet Core
ERAB	E-UTRAN Radio Access Bearer
ETSI	European Telecommunications Standards Institute
EU	European Union
FG	Functional Group
FP	Framework Programme
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
IoT	Internet of Things
IRI	Internationalized Resource Identifier
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LinDA	Linked Data
LODE	Live OWL Documentation Environment
LTE	Long Term Evolution
M2M	Machine-to-Machine
M3	Machine-to-Machine Measurement
MAS	Management, Abstraction and Semantics
OCB	Orion Context Broker
oneM2M	International Machine-to-Machine Standardization
OWL	Web Ontology Language
QK	QuantityKind
RAT	Reference Annotator Tool
RDF	Resource Description Framework
RRC	Radio Resource Control
S1	The interface between an eNodeB and the Core Network
SAREF	Smart Appliances REFerence
SSN	Semantic Sensor Networks
STF	Special Task Force
TDB	Triple store
URI	Universal Resource Identifier
W3C	World Wide Web Consortium
WP	Work Package
XLS	Microsoft Excel spreadsheet
XML	Extensible Markup Language

Ontology Namespaces

Prefix	Ontology/Language	Namespace
dul	DOLCE+DnS Ultralite Ontology	http://www.loa.istc.cnr.it/ontologies/DUL.owl#
geo	Basic Geo (WGS84) ontology/ WGS84 Geo Positioning	http://www.w3.org/2003/01/geo/wgs84_pos#
iot-lite	IoT-lite	http://purl.oclc.org/NET/UNIS/fiware/iot-lite#
m3	M3	http://sensormeasurement.appspot.com/m3#
m3-lite	M3-lite	http://purl.org/iot/vocab/m3-lite#
oneM2M or base-ontology	oneM2M base Ontology	http://www.onem2m.org/ontology/Base_Ontology#
owl	Web ontology Language	http://www.w3.org/2002/07/owl#
qu	QuantityKind	http://purl.org/NET/ssnx/qu/qu#
qudt	Quantities, Units, Dimensions and Data Types Ontologies	http://qudt.org/schema/qudt#
qudt_unit	QUDT unit ontology	http://data.qudt.org/qudt/owl/1.0.0/unit.owl#
rdf	RDF Concepts Vocabulary	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	RDF Schema ontology	http://www.w3.org/2000/01/rdf-schema#
saref	Smart Appliances REFerence Ontology	http://ontology.tno.nl/saref.ttl
ssn	W3C SSN ontology	http://purl.oclc.org/NET/ssnx/ssn#
time	OWL time ontology	http://www.w3.org/2006/time#
xsd	XML schema Definition	http://www.w3.org/2001/XMLSchema#

1 EXECUTIVE SUMMARY

This deliverable provides an update to the already existing deliverable 3.1.1 [1]. It is noteworthy to report that the current deliverable should not be considered as a standalone deliverable but it instead should be read along with deliverable 3.1.1 [1]. This deliverable does not provide all the state of the art ontologies and annotation tools but provide our new findings only. It reports updates performed to the ontology with respect to the new additions. Some key updates performed in the ontology are: (a) addition of oneM2M concepts as `equivalentClasses` in the FIESTA-IoT ontology thereby allowing oneM2M specific testbeds to become part of the federation that is provided by FIESTA-IoT platform, (b) addition of more concepts in M3-lite taxonomy to use communication related `m3-lite:QuantityKinds` that are made available via Com4Innov testbed, and (c) replacement of `dul:TimeInterval` with `time:Instant`. The replacement of `dul:TimeInterval` is mainly due to the definition of `dul:TimeInterval`. All current testbeds provide instantaneous observation values, thus it is important to align with a concept that could support such instantaneous behavior. Further, use of `time:Instant` reduces the dependency on DUL that seems to be not persistent. It is also noteworthy to state that as more testbeds join FIESTA-IoT platform federation, need for having more concepts defined in the taxonomy will grow. Thus, we envision FIESTA-IoT ontology and its related M3-lite taxonomy to be living documents.

No ontology is significant unless it is provided with necessary relevant documentation and supporting Reference Annotator Tool (RAT). We provide the audience with a set of RATs build specifically to support current in-house testbeds (SmartSantander, SmartICS, testbed provided by Com4Innov and KETI and another new additional SoundCity testbed). For more information on the testbeds we redirect audience to Deliverable 2.2 [2]. The potential testbeds who are willing to join the federation provided by FIESTA-IoT platform can reuse a RAT. Although, testbed owners would have to create their own annotators, a RAT can guide them how to annotate the IoT data provided by their testbed. RATs follow some practices that each testbed should follow. Most of these are reported in the best practices and guidelines section. Before annotations can be stored in the FIESTA-IoT platform, Annotation Validation Tool (AVT) validates the annotations coming from a testbed. An overview of AVT is provided in the Section 4.2. The AVT performs checks on syntactical errors and checks for the required and imposed cardinalities by FIESTA-IoT ontology.

FIESTA-IoT platform uses Jena Triple store (TDB) to store the data provided by the testbeds. Along with TDB, we use Jena spatial to support spatial queries that help in creating experiments that require spatio-temporal support. As FIESTA-IoT platform stores instantaneous observation values, mobility of resources can be inferred using SPARQL queries. Jena spatial only provides support to query TDB with queries that requires knowledge either about which device is near, within a bounding box or within a range of “x” from a given location. More details about such queries are provided in Section 3.5. However, as this task is still ongoing, we do not report final results but provide current findings.

The APIs to access data and store annotated data in the FIESTA-IoT platform, however, being within the scope of this deliverable will be reported in another deliverable (deliverable 3.2.2 to be precise). This decision is made in order to have all the APIs developed as part of WP3 in one deliverable and to leverage from the task end time (due in Month 24). To be brief, there are two graphs in which data

provided by testbeds is stored. These graphs are resources and observations graph. To access data stored within these graphs, there are several sets of APIs (/testbed, /resources, /observations, /queries) to access data therein. These APIs are further classified into 2 categories; one that takes SPARQL query as input while another that uses already stored queries. However, as previously said, in this deliverable we do not report these APIs because the task related to the development of the APIs ends in month 24.

We refer the audience to deliverable 3.1.1 in order to know more about the WP3, its scope, related tasks and targeted audience.

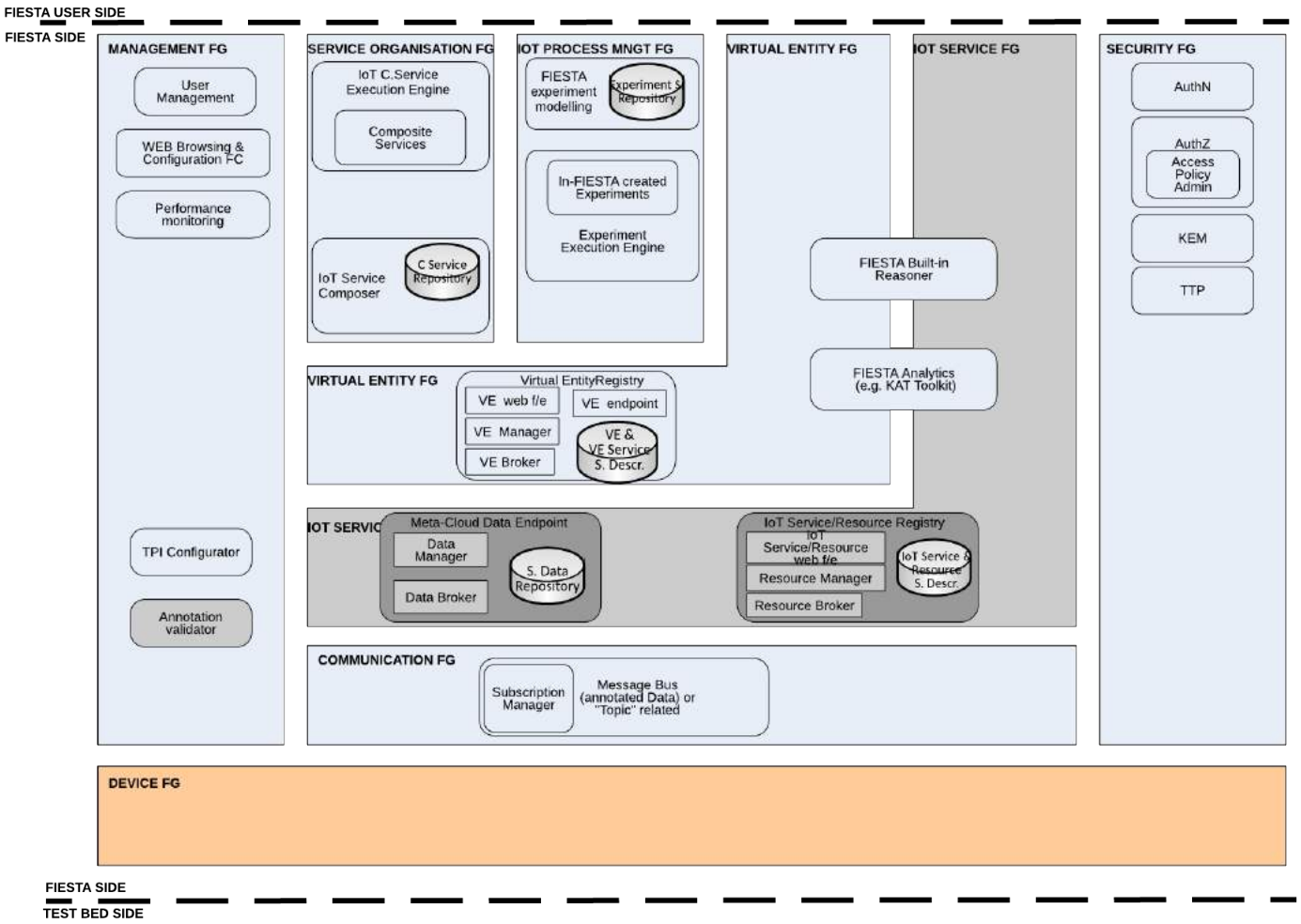

Figure 1: FIESTA-IoT Architecture with components addressed in this deliverable marked in grey.

1.1 Relation with the Architecture and Requirements

The functional architecture of the FIESTA-IoT platform is described in [3]. In relation to the platform, in this deliverable, we have mainly focused on describing a subset of its components, IoT Services Functional Group (FG) and Annotation Validation component in Management Functional Group. In Figure 1 (part of the FIESTA-IoT Architecture), we identify these components (marked in gray). Within IoT Service, in this deliverable we mainly focus on the semantic model used for storing data in “Semantic Data Repository” and “IoT Service and Resource Semantic Description” component. We name the semantic model developed as FIESTA-IoT ontology and

its supporting Taxonomy as M3-lite. As mentioned above, the component APIs and their description are made available in deliverable 3.2.2. Annotation Validator Component is also reported in the deliverable. As AVT also forms part of WP6 activities, it is also described in the WP6 related deliverables. In the following subsections, we report all the requirements addressed with respect to semantic model pertaining to the "Semantic Data Repository", "IoT Service and Resource Semantic Description" component and "Annotation Validator". In addition, we also report those requirements that could be realized using the semantic model and the SPARQL technology used to perform queries on the data stored in "Semantic Data Repository" and "IoT Service and Resource Semantic Description". For a list of all the functional and non-functional requirements we refer readers to deliverable 2.1 [4].

The semantic model is an essential part of the FIESTA-IoT platform. In order to query the semantic model we use the SPARQL query language. SPARQL querying enables users to query sample of data available in the FIESTA-IoT Meta-Cloud (14_FR_ACC_Sample_specific_fractions_data). Note that this sample can be the complete semantic repository. The semantic model is mainly focused on resources (sensors) and observations produced by them (more information on the semantic model is available in Section 3.2). The semantic model provides a link between resources and observations (satisfying 58_NFR_MEA_Link_measurements_resources) and making possible to know which resource has generated a particular observation. This enables us to address 08_FR_ACC_List_info_related_measurement. Both resources and observations are accompanied by related metadata (satisfying 60_NFR_MEA_Measurements_provide_metadata). Thus, it is possible to get all information related to a particular resource (satisfying 09_FR_ACC_List_info_related_resource) and a particular observation. Based on the requirements, from the semantic model it is possible to discover measurements (or observations) based on what phenomenon was observed (07_FR_ACC_Discover_data_phenomenon), chose associated metadata (12_FR_ACC_Choose_metadata_each_measurement), and discover observations that have specific metadata (16_FR_ACC_Discover_measurements_by_metadata). This is possible due to the link between observation concept and m3-lite:quantitykind (satisfying 62_NFR_MEA_Measurements_provide_phenomenon).

For resources, to identify from which testbed they are associated to, in the semantic model a deployment property (satisfying 63_NFR_RES_Link_resource_tesbed) has been added. Thus, following the semantic model, it is possible to create a query via in such a way that makes possible to get measurements from a particular testbed or different testbeds in a single request (satisfying 10_FR_ACC_Get_measurements_single_request). Further, it is possible to get resources based on metadata and thereby satisfying 15_FR_ACC_Discover_resources_by_characteristics. A platform can have multiple sensors. We use taxonomy to categorize sensors of different types. In this case, a platform (can also be called as a resource) produces multiple observations coming from different sensors that are in-built or available on a platform. Thereby, it satisfies 36_NFR_PLA_Resources_produce_different_ measurements.

As stated before, incoming data (annotated data (satisfying 72_NFR_SEM_Semantic_annotations_data) that ensures testbed that host data in

their own proprietary format, is mapped to FIESTA-IoT semantic model (satisfying 73_NFR_SEM_Mapping_semantic_model).) from testbeds to FIESTA-IoT platform IoT-Services FG is validated (satisfying 40_NFR_PLA_Process _feedbacks). As this annotated data is stored in one of the repositories in the FIESTA-IoT IoT-Services FG, each resource is uniquely identified by an URI (satisfying 65_NFR_RES_Resource_identified_code). Table 1 lists the above-mentioned requirements along with their descriptions.

Table 1: Requirements addressed

Requirement ID	Description
07_FR_ACC_Discover_data_phenomenon	It must be possible to get/discover measurements based on what phenomenon is observed
08_FR_ACC_List_info_related_measurement	It must be possible to list all information related to a measurement (metadata and characteristics of the resource that generated the measurement)
09_FR_ACC_List_info_related_resource	It must be possible to list all information related to a resource instance (characteristics of the resource, and also measurements and metadata generated by the resource)
10_FR_ACC_Get_measurements_single_request	It must be possible to get measurements from different testbeds with a single request
12_FR_ACC_Choose_metadata_each_measurement	It should be possible for the experimenter to choose the metadata associated with the measurements
14_FR_ACC_Sample_specific_fractions_data	It must be possible to sample a specific fraction of the data
15_FR_ACC_Discover_resources_by_characteristics	It must be possible to get/discover resources based on characteristics
16_FR_ACC_Discover_measurements_by_metadata	It must be possible to get/discover measurements based on metadata
36_NFR_PLA_Resources_produce_different_ measurements	FIESTA-IoT must support resources that produce different measurements that can be linked or not
40_NFR_PLA_Process _feedbacks	FIESTA-IoT should process the measurements and / or resources feedback to validate the functioning of resources
58_NFR_MEA_Link_measurements_resources	It must be possible to link the measurement with the resource (and its characteristics) that generated that measurement
60_NFR_MEA_Measurements_provide_metadata	Each collected measurement must provide metadata
62_NFR_MEA_Measurements_provide_phenomenon	Each measurement must state the phenomenon
63_NFR_RES_Link_resource_tesbed	It must be possible to link the resource with the testbed where it is hosted
65_NFR_RES_Resource_identified_code	Every resource must be univocally identified by a code
72_NFR_SEM_Semantic_annotations_data	Semantic annotations of data
73_NFR_SEM_Mapping_semantic_model	Mapping between testbed and FIESTA-IoT semantic model must be ensured

2 RELATED WORK

In this section we present only the extensions to the related work section that was made available in previous version of the deliverable 3.1.1 [1].

2.1 IoT-related Ontologies

In this section, we provide new findings with respect to IoT related ontologies. These findings, related to updated SSN, linking between oneM2M and SAREF ontologies and other ontologies that are planned to have extensions.

2.1.1 SSN Updates

Recently, an updated version of SSN ontology was made available as a working draft[1]. This version has many updates. For instance, some updates are related to: (a) separation of DUL ontology from SSN ontology and (b) movement of `ssn:Sensor` subclass from `dul:PhysicalObject` to `dul:Object`. However, as this is still a working draft and not a final release we have refrained ourselves in changing to the new namespace. As the FIESTA-IoT ontology is a live document, we will consider modifying FIESTA-IoT ontology with the new namespace of SSN as soon as it is released. Changing the SSN namespace and modifying the FIESTA-IoT ontology will have a widespread impact. All the testbeds would be required to modify their annotator to reflect the changes. Further, if they are already using SSN ontology to store their proprietary data, we assume that they would have already modified their process. Similarly experimenters would have to modify their queries to include new namespace. Moreover, from the FIESTA-IoT platform side, already available data that was made available before the change would not be accessible as it uses old SSN namespace. Further, we also need to assess changes in the imported SSN concepts in the FIESTA-IoT ontology. Also to mention, we would have to look forward to see if IoT-lite adopts new SSN. Thus, as this change in the namespace has a huge impact and we will follow carefully and closely the move.

2.1.2 Mapping between two standardized ontologies: OneM2M base ontology and SAREF ontology

The oneM2M base ontology and the SAREF ontology were introduced in D3.1.1. They are both standardized reference ontologies (oneM2M TS0012[2] [5] and ETSI TS103264[3] respectively) with a good visibility in the market. They were created and designed with a different aim:

- OneM2M base ontology aims to provide a high level ontology for the IoT market in order to provide a minimal set of common knowledge that enables the cross-domain syntactic and semantic interoperability. As it is quite high-level and abstract, oneM2M expects external ontologies that describe a specific domain of interest in a more detailed way to be mapped to the oneM2M base ontology. With these mappings of different domain-specific

[1] https://www.w3.org/TR/vocab-ssn/

[2] http://www.onem2m.org/images/files/deliverables/Release2/TS-0012-oneM2M-Base-Ontology-V2_0_0.zip

[3] http://www.etsi.org/deliver/etsi_ts/103200_103299/103264/01.01.01_60/ts_103264v010101p.pdf

ontologies to oneM2M base ontology, the internetwork between devices and things from different domains is enabled.

- SAREF aims to provide a common knowledge for the domain of Smart Appliance, especially on the energy consuming aspect. Compared to oneM2M base ontology, it is less high level and more applicable to describe devices.

The mapping between oneM2M base ontology and SAREF is performed by oneM2M WG5 MAS (Working Group 5 Management, Abstraction and Semantics). The working group who has drafted the TS0012 in which 4 relationships have been specified for mapping external ontologies to oneM2M base ontology: `rdfs:subClassOf`, `owl:equivalentClass`, `rdfs:subPropertyOf`, `owl:equivalentProperty`. Basically, SAREF is sub-classed to oneM2M base ontology. Table 2 provides a list of mapped classes using subclass relationship.

Table 2: Subclass mapping between SAREF and the base ontology [5]

Class in SAREF	Mapping relationship	Class in oneM2M Base Ontology
saref:Device	rdfs:subClassOf	oneM2M:Device
saref:BulidingObject	rdfs:subClassOf	oneM2M:Thing
saref: BulidingSpace	rdfs:subClassOf	oneM2M:Thing
saref:Command	rdfs:subClassOf	oneM2M:Command
saref:Commodity	rdfs:subClassOf	oneM2M:Thing
saref:Function	rdfs:subClassOf	oneM2M:Functionality
saref:Property	rdfs:subClassOf	oneM2M:InputDataPoint OR oneM2M:OutputDataPoint
saref:Service	rdfs:subClassOf	oneM2M:Service
saref:UnitOfMeasure	rdfs:subClassOf	oneM2M:MetaData
saref:ActuatingFunction	rdfs:subClassOf	oneM2M:ControllingFunctionality
saref:MeteringFunction	rdfs:subClassOf	oneM2M:MeasuringFunctionality
saref:SensingFunction	rdfs:subClassOf	oneM2M:MeasuringFunctionality
saref:State	rdfs:subClassOf	oneM2M:InputDataPoint OR oneM2M:OutputDataPoint
saref:Profile	rdfs:subClassOf	oneM2M:Thing
saref:Task	rdfs:subClassOf	oneM2M:Thingproperty
saref:DeviceCategory	rdfs:subClassOf	oneM2M:Thingproperty
saref:FunctionCategory	rdfs:subClassOf	oneM2M:Aspect

Another on-going working group entitled ETSI Special Task Force (STF) 513 is also working on the SAREF/oneM2M base ontology mapping.

2.1.3 Other Ontologies

Note that there are many other IoT related ontologies that are available. However, as said in the previous version Deliverable 3.1.1 [1] we only report few ontologies relevant to us. Further, we also know that many ontologies extensions (for example Schema.org[4]) are planned to also have IoT support. As some of these might be standards and widely used we would look forward to see the changes. As these ontologies would lack implementation from testbed perspective, we envision no updates on FIESTA-IoT ontology.

2.2 Ontology Annotation tools (LinDA Transformation tool)

In this section we provide related available annotation tools. Note that, there are many annotation tools available that could convert non-semantic data and represent it in a semantic form. Some of the tools were mentioned in the Deliverable 3.1.1. Further, a survey of few tools with such conversion functionality is available in [6]. We provide brief overview of LinDA transformation tool that provides such functionality.

LinDA Project[5] (an FP7 project) provides open-source Enterprise Linked Data tool that allows data from multiple sources to be transformed to linked data that can then be used for analysis. Using this tools, enterprises can combine data from multiple sources into a single format, thereby assisting easy management of data. One of the main component of LinDA Project architecture is the transformation tool that allows conversion of data stored in different formats (for e.g., XLS, CSV, and relational database (DB)) to RDF format. LinDA offers a workbench (see Figure 2) for novice users in the form of a Graphical User Interface (GUI) where the users can upload their datasets and convert it to the required RDF form. LinDA Transformation Tool (see Figure 3) interprets semantic annotations and let users validate those classes and properties found for the data. If issues are found, LinDA Transformation Tool allows users to define classes and properties for their data from a set of already available vocabularies (stored in a repository referred to as Vocabulary Repository). The workflow to transform data is listed below:

- A user should upload the data choosing the relevant data format.
- Once the file is uploaded, the tool allows the users to select fields (data) to be transformed.
- The users can then provide a base Universal Resource Identifier (URI) for the annotated data description. If the user does not want to provide a base URI, blank nodes are used.
- The tool then looks up the Vocabulary repository to identify suitable classes for the selected fields and associated properties.
- The users then validate the identified properties.
- After validation they specify datatypes for the classes.
- The Transformation tool allows the users to modify the RDF created.
- Publish their data as N-triples.

[4] https://github.com/schemaorg/schemaorg/issues/1272

[5] http://linda-project.eu

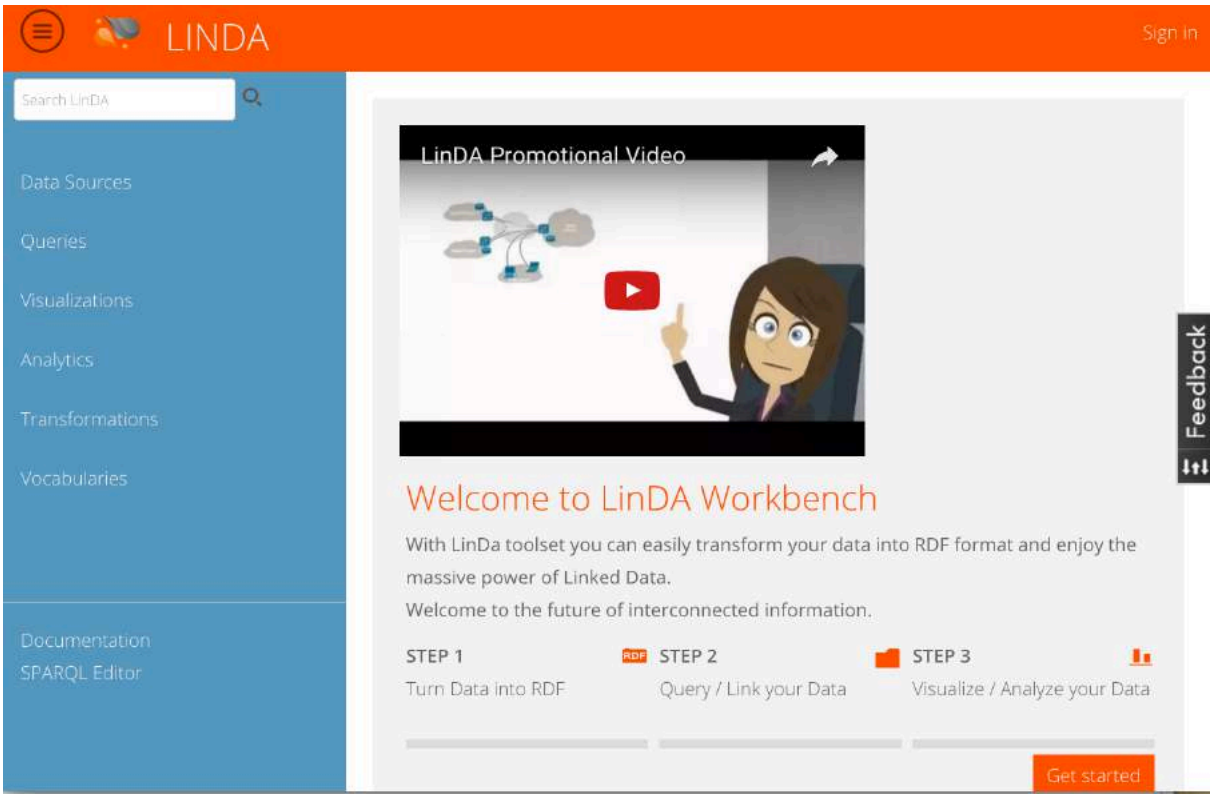

Figure 2: LinDA Workbench

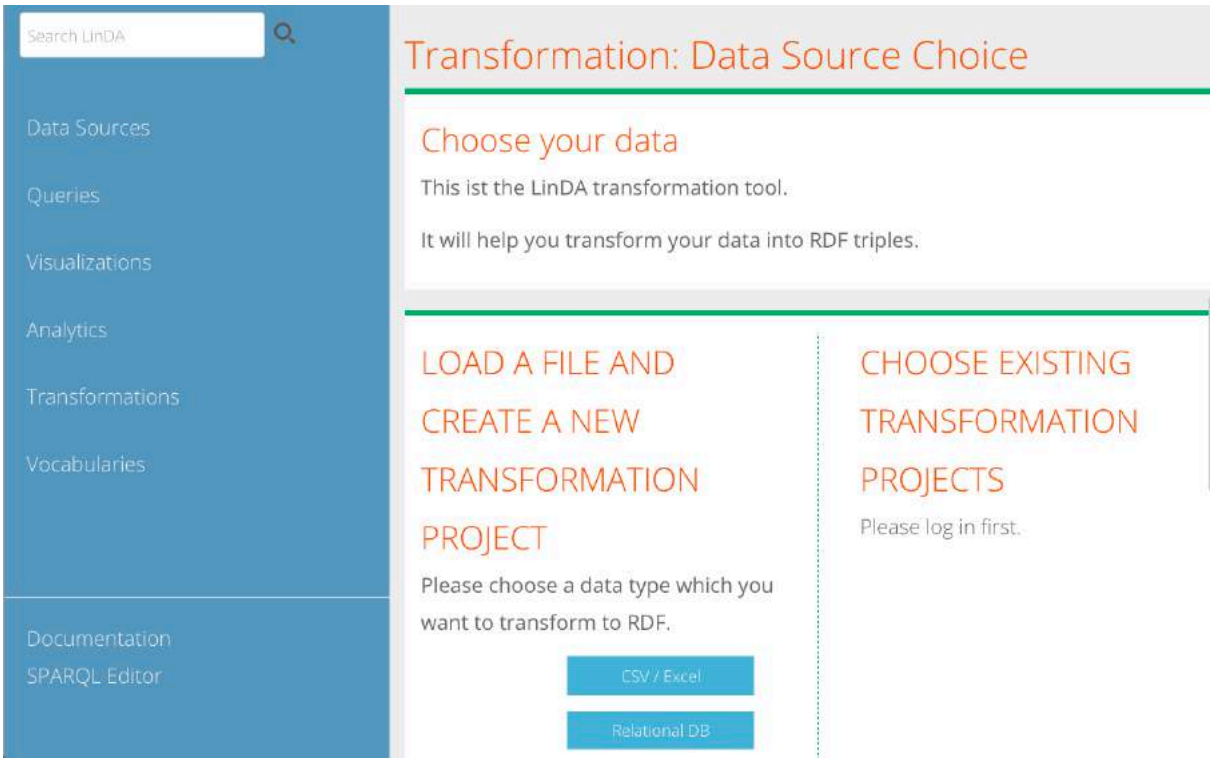

Figure 3: LinDA Transformation tool

3 ALIGNING EXISTING ONTOLOGIES

In this section, we provide information on more ontologies that are aligned within FIESTA-IoT ontology.

3.1 Ontologies Aligned

OneM2M and SAREF are well known ontologies. We consider the fact that many testbeds would be using such ontologies to store their data. In order to help testbeds in adopting FIESTA-IoT ontology we provide a mapping between oneM2M ontology and FIESTA-IoT ontology. We add such mappings within FIESTA-IoT ontology using either `owl:equivalentClass` or `owl:equivalentProperties`.

Table 3 and Table 4 show concepts from oneM2M base ontology that are mapped in FIESTA-IoT ontology.

Table 3: Classes mapping between oneM2M Base ontology and FIESTA-IoT ontology

Class in Base Ontology	Mapping relationship	Class in FIESTA-IoT
oneM2M:Thing	owl:equivalentClass	ssn:Sensor
oneM2M:ThingProperty	owl:equivalentClass	m3-lite:QuantityKind
oneM2M:Device	owl:equivalentClass	m3-lite:SensingDevice
oneM2M:Service	owl:equivalentClass	ssn:Observation
oneM2M:OperationOutput	owl:equivalentClass	ssn:ObservationValue
oneM2M:MetaData	owl:equivalentClass	m3-lite:Unit

Table 4: Object properties mapping between oneM2M Base ontology and FIESTA-IoT ontology

Object property in BaseOntology	Mapping relationship	Object property in FIESTA-IoT
oneM2M:hasThingProperty	owl:equivalentProperty	iot-lite:hasQuantityKind
oneM2M:hasService	owl:equivalentProperty	ssn:madeObservation
oneM2M:hasMetaData	owl:equivalentProperty	iot-lite:hasUnit

Note that, this mapping also allows SAREF compliant testbeds to also join FIESTA-IoT.

3.2 FIESTA-IoT Ontology

The current version of the FIESTA-IoT Ontology (see Figure 4) is a merge of existing IoT ontologies into a single one. As it can be seen in the Figure 4, it fosters concepts from a number of "third-party" ontologies such as WGS84[6], W3C SSN, IoT-lite, M3-lite Taxonomy, DUL, Time[7] and QU. Below, we present updates that we have performed:

- `m3-lite:QuantityKind` represents sensed phenomenon while `m3-lite:Unit` is the measurement unit. `m3-lite:QuantityKind` is also linked to `m3-lite:Source` that holds the source of sensed phenomenon for example `m3-lite:SoundSource` (that can be construction work, siren, etc.). More information on this concept is provided in Section 3.3.
- The observations taken by the sensing devices are linked to: (a), the corresponding `m3-lite:QuantityKind` via `ssn:observedProperty`, (b) `ssn:observationSamplingTime` that further links to `time:Instant` to represent when the observation was taken (i.e. timestamp), (c) `geo:Point`, (d) `ssn:ObservationValue` via `ssn:SensorOutput` (`ssn:ObservationValue` is linked to the sensed value of the QuantityKind via data property `dul:hasDataValue`, and `m3-lite:Unit` concept), and (e) `m3-lite:hasMeasurementType` to know whether the measurement was `m3-lite:Manual`, `m3-lite:Automatic` or generated from an `m3-lite:Experiment`. One important change that has been made to the ontology is the replacement of `dul:TimeInterval` to `time:Instant`. `dul:TimeInterval` concept relates to the interval in which the observation was taken. In FIESTA-IoT ontology we consider all the observations to be instantaneous. In case a resource provides an average observation value during the observation period we still link the observation value to a `time:Instant`. This `time:instant` can be any time during the observation period. The `time:Instant` is then linked to `xsd:dateTime` using `time:inXSDDateTime` data property. Previously, only `xsd:Double` datatype values were supported for `ssn:ObservationValue`. However, the observations could be of Boolean type depending on the type of sensors (for example, proximity sensor would result in a 'true' or 'false' value). We add the `xsd:boolean` to the range of `dul:hasDataValue`.
- We add oneM2M classes as equivalent classes and properties as equivalent properties to facilitate oneM2M compliant testbeds join FIESTA-IoT platform.

As discussed in the previous deliverable D3.1.1 [1], there are two parts in the FIESTA-IoT ontology. One related to resources and another related to observations. Figure 5 and Figure 6 show the concepts used in the resource graph and the observation graph respectively. Nevertheless both the graphs are connected.
For the sake of having an illustrative example, we encourage the reader to take a look at Section 4.1 and Appendix II – Sample Usage of Ontology, where we have put different annotations of resources or observations, gathered from the various "in-house" Testbeds.

[6] WGS84 is actually a basic RDF vocabulary that provides the Semantic Web Community with a namespace for the representation of latitude, longitude and other information about spatially-located things. For reference please see https://www.w3.org/2003/01/geo/

[7] https://www.w3.org/TR/owl-time/

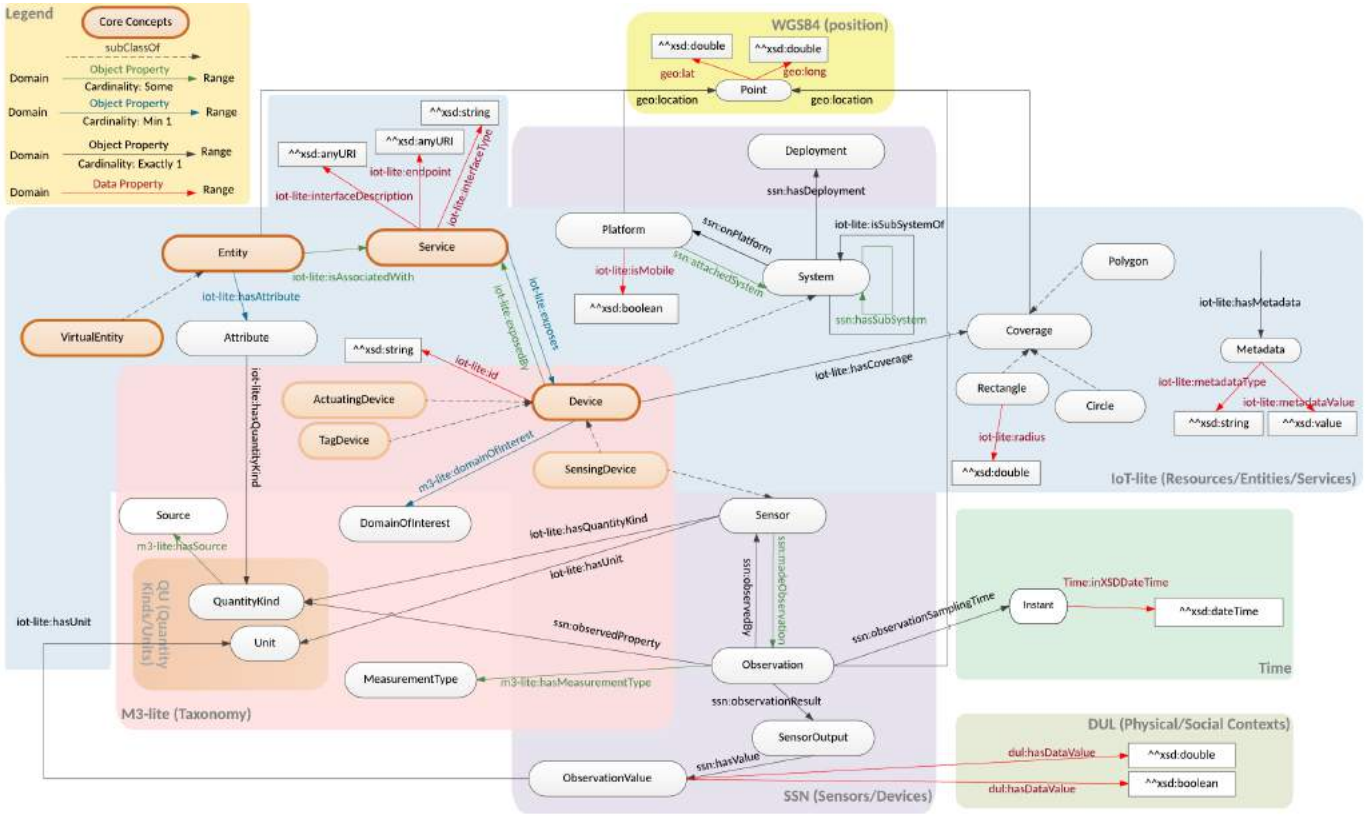

Figure 4: FIESTA-IoT Ontology

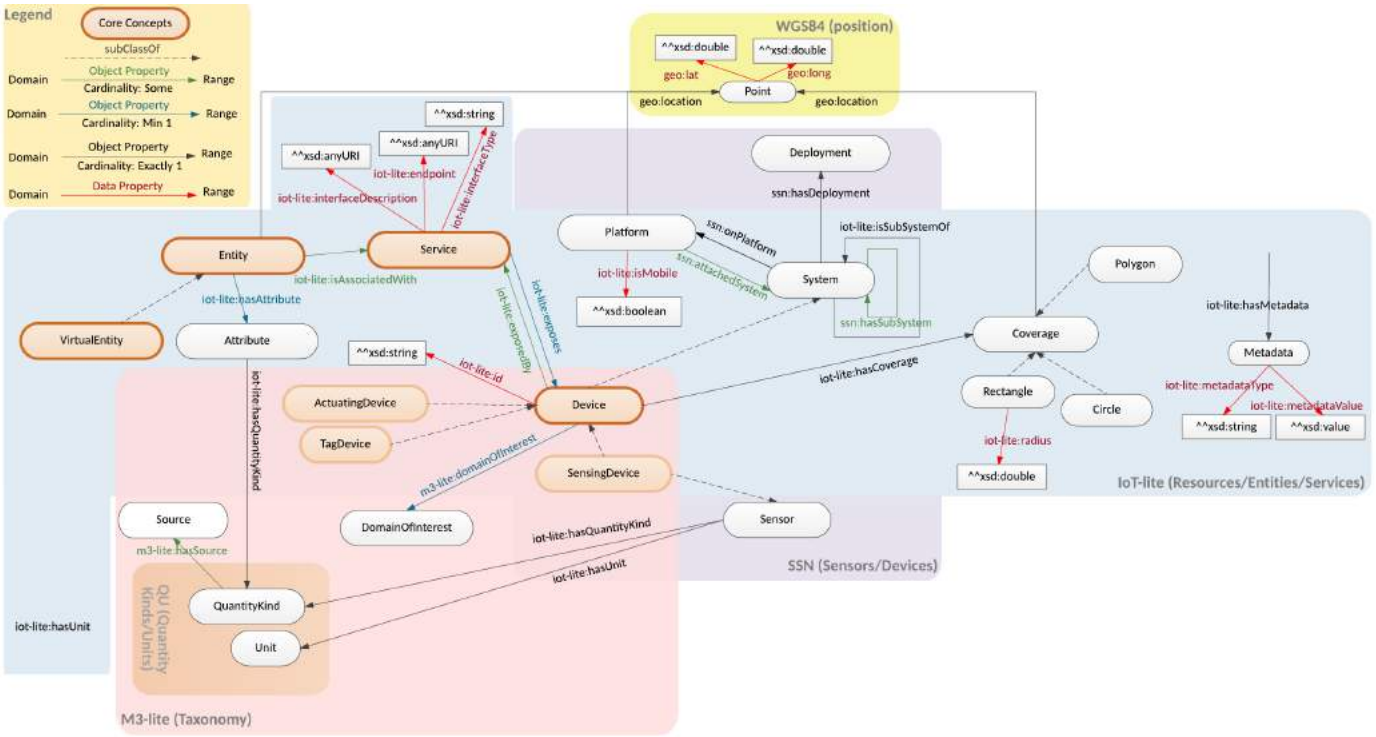

Figure 5: Resources related graph

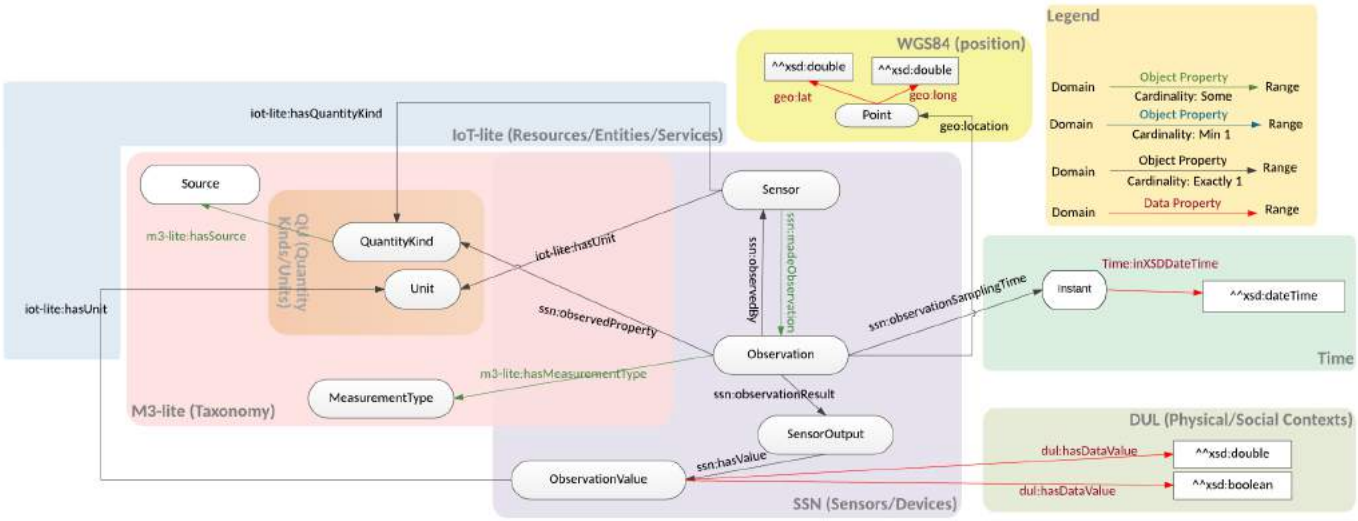

Figure 6: Observations related graph

3.2.1 Statistics Updates

Currently, within the ontology we have 450 classes, 28 object properties and 10 data properties. Note that we do not consider data properties of WGS84 as, currently, they are provided as annotation properties. Further, we have 7 classes that are equivalent (one being Entity and Object and other coming from the mapping of oneM2M base ontology), 3 equivalent object properties (coming from the mapping of oneM2M base ontology) and 4 inverse properties. Almost all the used object and data properties have domains and ranges specified. We let `iot-lite:metadata` domainless so that it can be attached to any concept. Moreover there are many `SubClassOf`, `SubObjectPropertyOf` and `SubDataPropertyOf` relation.

3.3 M3-lite taxonomy Updates

As M3-lite taxonomy is a live document, changes in the M3-lite taxonomy were also performed. These mainly include addition of more types of sensors, more sub-classes of `m3-lite:QuantityKind` and `m3-lite:Unit`. To further elaborate:

- We include `m3-lite:TouchSensor` and `m3-lite:ImageSensor` as subclasses of `ssn:SensingDevice`.
- New `m3-lite:QuantityKind` subclasses have been added. These subclasses span the domain of Communications (`m3-lite:Communication`). Further, more subclasses of `m3-lite:Communication` are added to support those concepts currently being provided by the Com4Innov testbed. Table 5 lists all those concepts that are added as subclass of `m3-lite:Communication`.

Table 5: Com4Innov Testbed QuantityKinds

QuantityKind	Description
Acc_InitialERABEstabSuccRate	Initial ERAB establishment Success Rate
Acc_InitialERABSetupSuccRate	Initial ERAB Setup Success Rate
Acc_RrcConnSetupSuccRate	RRC connection Setup Success Rate
Acc_S1SigEstabSuccRate	S1 Signaling Establishment Success Rate

Int_DlLatency	Downlink Latency
Int_DlThroughput_kbps	Downlink Throughput
Int_UlPacketLoss	Uplink Packet Loss
Int_ UlThroughput_kbps	Uplink Throughput
Mob_HoExecSuccRate	Handover Execution
Mob_HoPrepSuccRate	Handover Preparation Success Rate
Mob_MobilitySuccRate	Handover Mobility Success Rate
Res_AverageLicConnectedUsers	Connected Users
Ret_ERabDrop	ERAB Drop

In order to achieve this aim, we modify the definition of `m3-lite:QuantityKind` to include communication parameters as well.

- Observations being made by the sensors measure the phenomenon of interest. These observations are generated by a source. Thus, we incorporate `m3-lite:Source` concept and we associate it with `m3-lite:QuantityKind` using object property `m3-lite:hasSource`. We further provide subclasses to the `m3-lite:Source`. These subclasses are mainly: `m3-lite:Others` and `m3-lite:SoundSource`. `m3-lite:SoundSource` that is further sub-classed to include sound sources like `m3-lite:Fan`, `m3-lite:Animal`, `m3-lite:ConstructionWork`, `m3-lite:Crowd`, `m3-lite:Neighbors`, `m3-lite:PublicTransit`, `m3-lite:Sirens` and `m3-lite:Traffic`. `m3-lite:Others` is used to report any other type of source which might be currently missing. The `m3-lite:hasSource` has been classified into `m3-lite:hasSoundSource`.
- New `m3-lite:QuantityKinds` like `m3-lite:Proximity` and `m3-lite:RecognizedActivity` have also been added.
- Each sensing device can have a different sensing mechanism that may result in different kinds of sensor data. For example, a sensor that must be triggered manually may collect fewer observations than a sensor that is automatically triggered at regular intervals. The sensing mechanism is an important feature to classify different sensors into groups before the experimenters are allowed to use them. Thus, we include the concept of `m3-lite:MeasurementType`. The `m3-lite:MeasurementType` is further sub-classed to include concepts like `m3-lite:Automatic`, `m3-lite:Calibration`, `m3-lite:Experiment`, `m3-lite:Invalid`, `m3-lite:Manual`, and `m3-lite:Others`.

3.4 FIESTA-IoT ontology and M3-lite Taxonomy Documentation

As reported in the previous deliverable 3.1.1, we have used LODE documentation creator to create the ontology and taxonomy documentation. We take advantage of this opportunity to provide a link to the deployed version of the ontology. The FIESTA-IoT ontology can be accessed via http://ontology.fiesta-iot.eu/ontologyDocs/fiesta-iot.html and m3-lite taxonomy can be accessed via http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.html.

3.5 SPARQL Queries on the FIESTA-IoT ontology

In this section, we provide few more examples of SPARQL queries that can be run on the data that satisfies the FIESTA-IoT Ontology.

- The following query provides all sensors at a given location at a given time

```
Prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
Prefix iotlite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
Prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#>
Prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
Prefix time: <http://www.w3.org/2006/time#>
Prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#>
Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
select ?s ?val
where {
       ?o a ssn:Observation.
       ?o ssn:observedBy ?s.
       ?o ssn:observedProperty m3-lite:Sound.
       ?o ssn:observationSamplingTime ?t.
       ?o geo:location ?point.
       ?point geo:lat "48.8393406"^^xsd:double.
       ?point geo:long "2.3931164"^^ xsd:double.
       ?t time:inXSDDateTime "2016-08-08T20:00:09.016Z"^^xsd:dateTime.
       ?o ssn:observationResult ?or.
       ?or  ssn:hasValue ?v.
       ?v dul:hasDataValue ?val.
} group by ?s ?val
```

- The following query provides all sensors at a given location between a given time interval. This would return all those sensors (in the required time interval) that have been previously at the specified location and/or are at the given location.

```
Prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
Prefix iotlite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
Prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#>
Prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
Prefix time: <http://www.w3.org/2006/time#>
Prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#>
Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
select ?s
where {
       ?o a ssn:Observation.
       ?o ssn:observedBy ?s.
       ?o ssn:observedProperty m3-lite:Sound.
       ?o ssn:observationSamplingTime ?t.
       ?o geo:location ?point.
       ?point geo:lat "48.8393406"^^xsd:double.
       ?point geo:long "2.3931164"^^xsd:double.
       ?t time:inXSDDateTime ?ti.
       FILTER (?ti >= "2016-08-07T20:00:09.016Z"^^xsd:dateTime>
          && ?ti < "2016-08-09T20:00:09.016Z"^^xsd:dateTime)
} group by ?s
```

- The following query gets most recent location of sensors.

```
Prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
Prefix iotlite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
Prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#>
Prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
Prefix time: <http://www.w3.org/2006/time#>
Prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#>
Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
select ?s (max(?ti) as ?tim) ?val ?lat ?long
where {
       ?o a ssn:Observation.
```



```
        ?o ssn:observedBy ?s.
        ?o ssn:observedProperty ?qk.
        ?qk a m3-lite:Sound.
        ?o ssn:observationSamplingTime ?t.
        ?o geo:location ?point.
        ?point geo:lat ?lat.
        ?point geo:long ?long.
        ?t time:inXSDDateTime ?ti.
        ?o ssn:observationResult ?or.
        ?or  ssn:hasValue ?v.
        ?v dul:hasDataValue ?val.
        {
                select  (max(?dt)as ?ti) ?s
                where {
                        ?o a ssn:Observation.
                        ?o ssn:observedBy ?s.
                        ?o ssn:observedProperty ?qk.
                        ?qk a m3-lite:Sound.
                        ?o ssn:observationSamplingTime ?t.
                        ?t time:inXSDDateTime ?dt.
                }group by (?s)
        }
} group by (?s) ?tim ?val ?lat ?long
```

Further, we also provide examples of queries where spatial functions from Jena spatial are used. Jena spatial will be used to create spatial index on the TDB. This enables queries like nearby and withinBox.

- The following query provides all sensors with latest observation near to a given location

```
Prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
Prefix iotlite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
Prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#>
Prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
Prefix time: <http://www.w3.org/2006/time#>
Prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#>
Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix spatial: <http://jena.apache.org/spatial#>
select ?s (max(?ti) as ?tim) ?val ?lat ?long
where {
        ?o a ssn:Observation.
        ?o ssn:observedBy ?s.
        ?o ssn:observationSamplingTime ?t.
        ?o geo:location ?point.
        ?t time:inXSDDateTime ?ti.
        ?o ssn:observationResult ?or.
        ?or  ssn:hasValue ?v.
        ?v dul:hasDataValue ?val.
        ?point geo:lat ?lat;
                geo:long ?long.
        {
                select  (max(?dt) As ?ti) ?s
                where {
                        ?o a ssn:Observation.
                        ?o ssn:observedBy ?s.
                        ?point spatial:nearby (48.82 2.37 0.5 'km').
                        ?s ssn:madeObservation ?o.
                        ?o ssn:observedProperty ?qk.
                        ?qk a m3-lite:Sound.
                        ?o ssn:observationSamplingTime ?t.
                        ?t time:inXSDDateTime ?dt.
                }group by ?s
        }
} group by (?s) ?tim ?val ?lat ?long
```


4 TOOLS

In this section we provide the overview of FIESTA-IoT annotation tools and validation tools that uses FIESTA-IoT ontology developed in Section 3.2.

4.1 FIESTA-IoT Annotation Tool

For this version of the deliverable, unlike it was done in the former one, we individually describe the approach followed by each of the four testbeds that were part of the FIESTA-IoT federation line-up, nailing down their most remarkable features and highlights.

Nonetheless, there is one thing that is worth highlighting that is related to the `ssn:Deployment` class. Up to now, a testbed provider could directly start registering his/her devices without any kind of authentication or checking. Nonetheless, the breakthrough achieved in the different activities from WP3 and WP4 (mainly those of [7] and [8]) dictates that, prior to attempt to store any kind of resource (nor observation), the testbed itself must be registered. During this process, the testbed provider will get the set of credentials and endpoints through which the rest of the registration steps have to be addressed. Said in layman's terms, apart from the semantic and syntactic validation that will be described in Section 4.2, two additional steps will be carried out: (1) an authentication token must come along every resource registration request in order to authenticate and validate that the message comes from the actual testbed owner. (2) Besides the previous phase, the resource description `ssn:Deployment` object must coincide with the one[8] already registered by the own testbed provider.

Nonetheless, prior to store an observation (and after the legacy validation process), the system (i.e. the very own *triplestore/iot-service* module) will check whether the originator of the observation, through the `ssn:observedBy` object property, is already registered in the IoT Service and Resource Registry. If this is the case, the observation will be stored; otherwise, it will be directly dropped out.

4.1.1 SmartSantander

Once the ontology has been revamped and now presents its (most likely[9]) definitive shape, the SmartSantander annotator must be tailored in a way it fosters these modifications, in order to keep its compatibility with regards to the FIESTA-IoT semantic requirements. As described in [1], from a proprietary JSON schema (whose template was introduced in [2]), where SmartSantander's resources and observations are defined, the annotator yields as an outcome a document (or documents), annotated in any RDF format, compliant with the ontology defined throughout this task.

Technically speaking, as for the modifications that are to be made on the resource description side between the legacy version (recall that we took SmartSantander as the guidelines for explaining the tool itself) and the updated one, there is no remarkable issue to deal with. As can be seen in Figure 7, where we present a graph

[8] A testbed provider might own more than one testbed/platform.

[9] The reader has to bear in mind that the ontology is a living entity and, thus, it is prone to undergo modifications in the forthcoming months.

that shows the annotation of an arbitrary resource from SmartSantander, nothing has changed, since all the modifications undergone by the ontology are completely orthogonal from the classes, data and object properties that are required for the annotation.

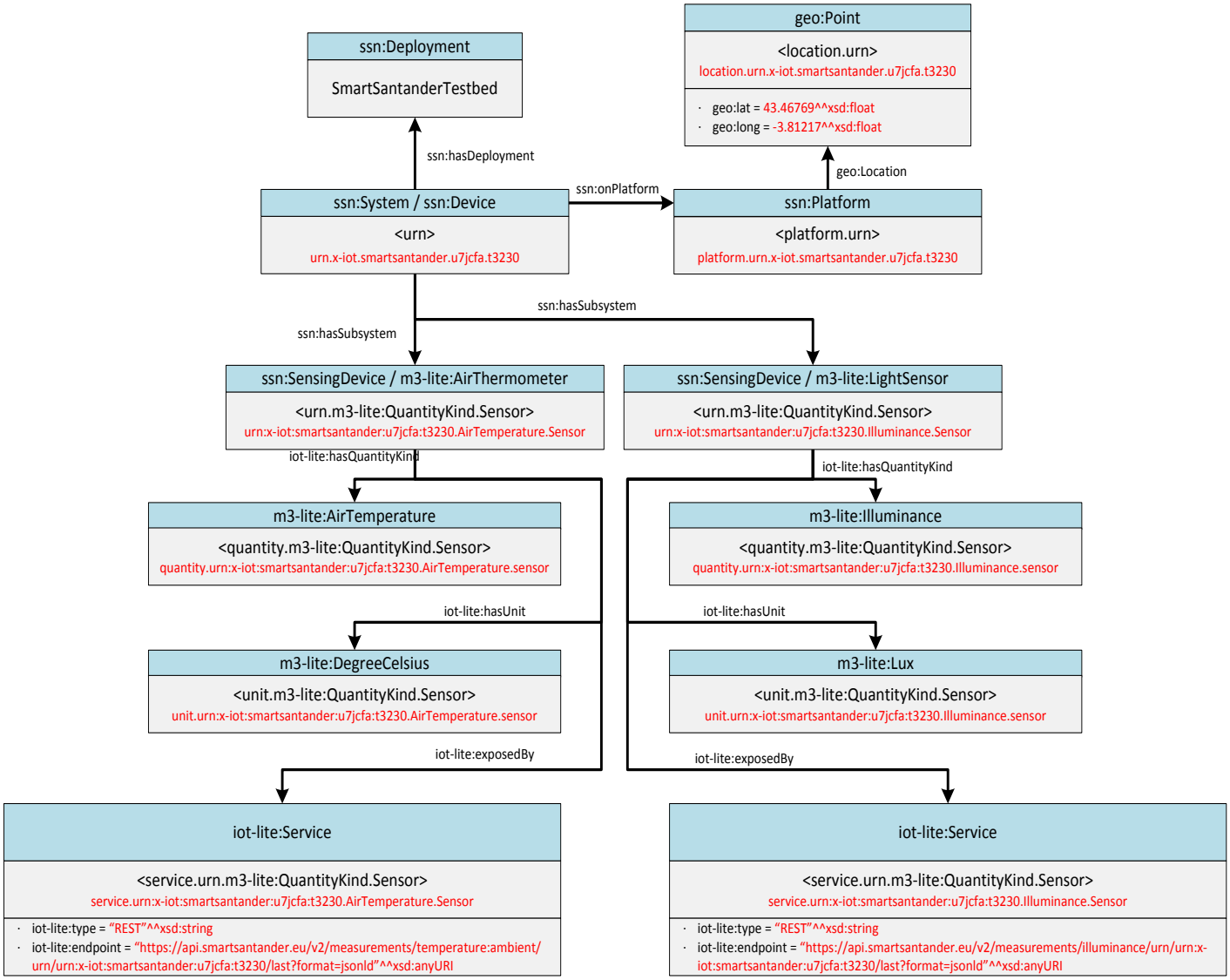

Figure 7: SmartSantander annotated resource description (graph)

When it comes to talk about the observations side, we did have to undertake a significant update: whereas in the legacy ontology we relied on *DUL* ontology to annotate the timestamp of a measurement (namely, through a `dul:TimeInterval` individual), in this second loop we leverage a concept from the *Time* Ontology: `Time:Instant` (see Figure 8).

The reason: for the sake of a better understanding, it is more intuitive to use a class that stresses the word "instant" rather than one that hints a time interval, that is, the time between two instants. So far, all the observations have "instantaneous" timestamp, thus we support `Time:Instant` and have modified SmartSantander annotator to reflect the change to `Time:Instant`.

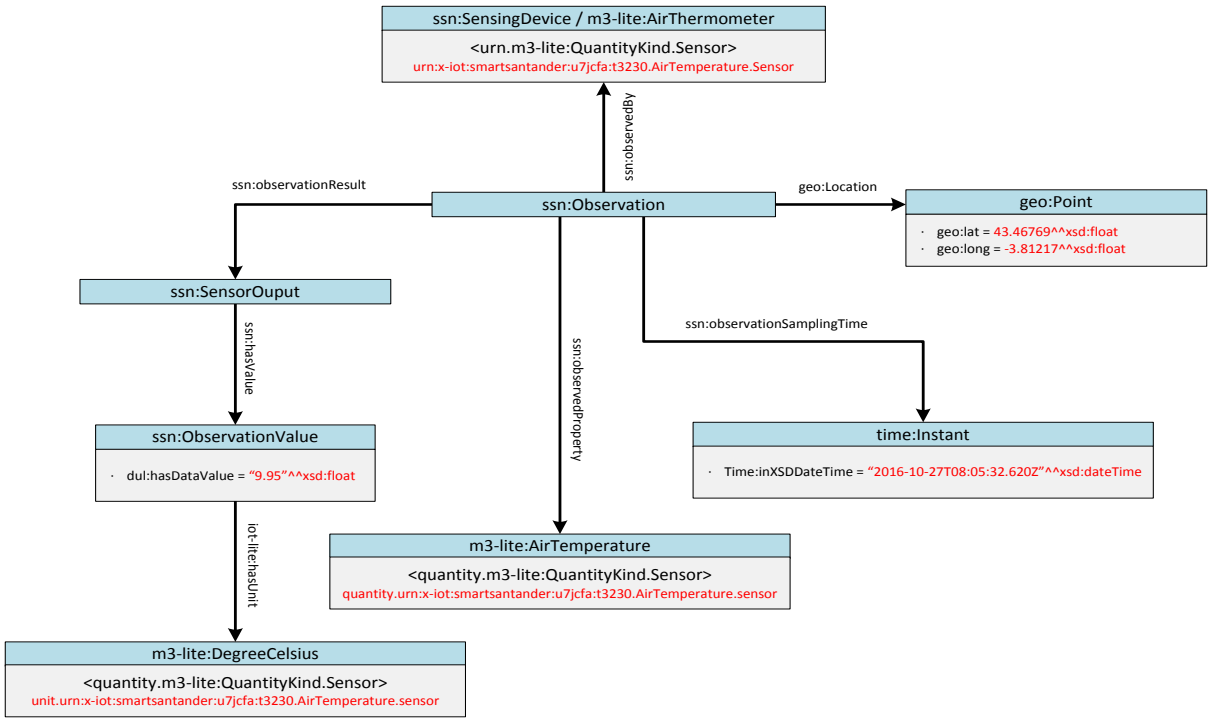

Figure 8: SmartSantander annotated observation (graph)

4.1.2 UNIS (Smart ICS)

The UNIS reference annotator allows the translation of the SmartICS testbed resource descriptions and data observations to the FIESTA-IoT compliant version (see Figure 9). For resource descriptions, some of the device variants, such as the "IoT Node", do not have any resource descriptions, and so semantic annotations are generated for each resource from scratch.

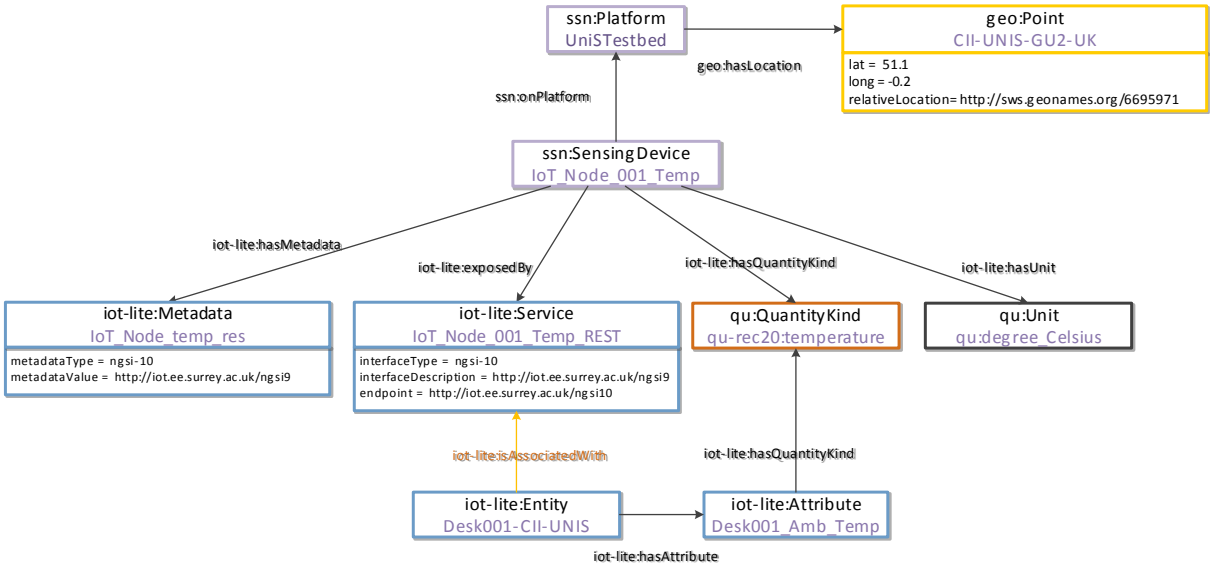

Figure 9: Modeling SmartCampus IoT Node Resources using FIESTA-IoT Ontology

For data observations, the original format is a JSON payload as shown in Figure 10.


```
{
  "reading" : {
    "light" : 501,
    "mic" : 20,
    "nodeId" : 1,
    "temp" : 23,
    "timeStamp" : "2016-11-01T11:56:39.119Z",
    "watts" : "2.43",
    "pir" : 1
  }
}
```


Figure 10: Sample Observation payload from SmartICS Testbed

As the "IoT Node" captures a set of readings from different phenomena, the FIESTA-IoT modeling is done on each sensing device. In the example below (see

Figure 11), the temperature sensor unit is modeled.

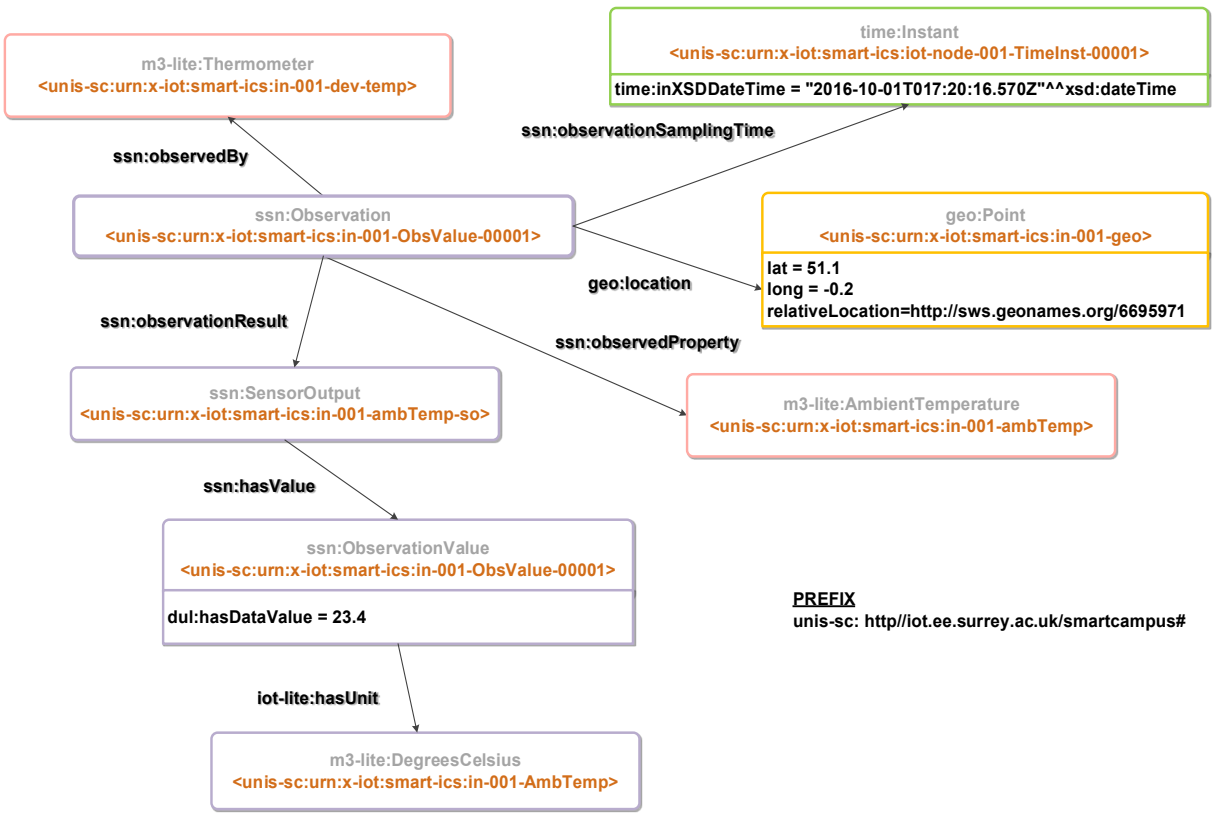

Figure 11: Modeling SmartCampus IoT Node Observations using FIESTA-IoT Ontology

4.1.3 KETI

The KETI annotator allows mapping oneM2M base ontology to FIESTA-IoT ontology in order to interwork FIESTA-IoT with existing oneM2M platforms or IoT testbeds in the semantic aspect. For this, KETI annotator supports re-annotation from oneM2M semantic data to FIESTA-IoT specific semantic data. To do this, KETI annotator uses mapping of oneM2M and FIESTA-IoT properties (provided in Section 2.1.2) for re-annotation.

For the mapping between two different ontologies, KETI annotator works as 1) retrieving semantic data from a resource, name <semanticDescriptor> of oneM2M Platform; 2) extracting classes and Individual from the semantic data according to oneM2M base ontology; 3) finding equivalent classes and object properties from the mapping ontology and then re-annotating semantic data according to FIESTA-IoT ontology; 4) finally, KETI annotator sends re-annotated semantic data to FIESTA-IoT according to upload procedure of FIESTA-IoT for example testbed registration, resource registration and sending real data.

4.1.4 Com4Innov

Com4Innov IoT testbed can provide different type of data from a big variety of type of resources. That way, it gives the opportunity to the FIESTA-IoT project and experimenter to choose from a broad collection of data in order to conduct their experiments. To start with, Com4Innov testbed provides the FIESTA-IoT user/experimenter with environmental measurements, namely Air Temperature, Board Temperature, Delta, Dewpoint, Output Voltage, Radiation Relative Humidity, SoundLevel, Latitude, Longitude, and Altitude. These measurements can be retrieved from the different types of devices that Com4Innov owns. Also, as Com4Innov is a 4G/LTE (soon to be 5G) telecommunications operator it can provide to the FIESTA-IoT experimenter data from the 4G core network (Evolved Packet Core – EPC) as well as from the eNodeBs' (air interface). These KPIs/measurements are the following: Initial ERAB establishment, Success Rate, Initial ERAB Setup Success Rate, RRC connection Setup Success Rate, S1 Signaling Establishment Success Rate, Downlink Latency, Downlink Throughput, Uplink Packet Loss, Uplink Throughput, Handover Execution, Handover Preparation, Success Rate, Handover Mobility Success Rate, the number of Connected Users and ERAB Drop.

All the data provided by the Com4Innov IoT testbed (namely the sensing devices, the EPC or the eNodeB) are gathered in a broker, which is the FIWARE Orion Context Broker (OCB) hosted in the Com4Innov data center. All the data, KPIs and measurements reach the FIWARE OCB in NGSI format and then, with the help of the annotator developed by Com4Innov, are converted to FIESTA-IoT specific semantic model and sent to the FIESTA-IoT metadata platform.

In Figure 12 and Figure 13 examples of resource graph and observation graph that is generated by the Com4Innov annotator are shown.

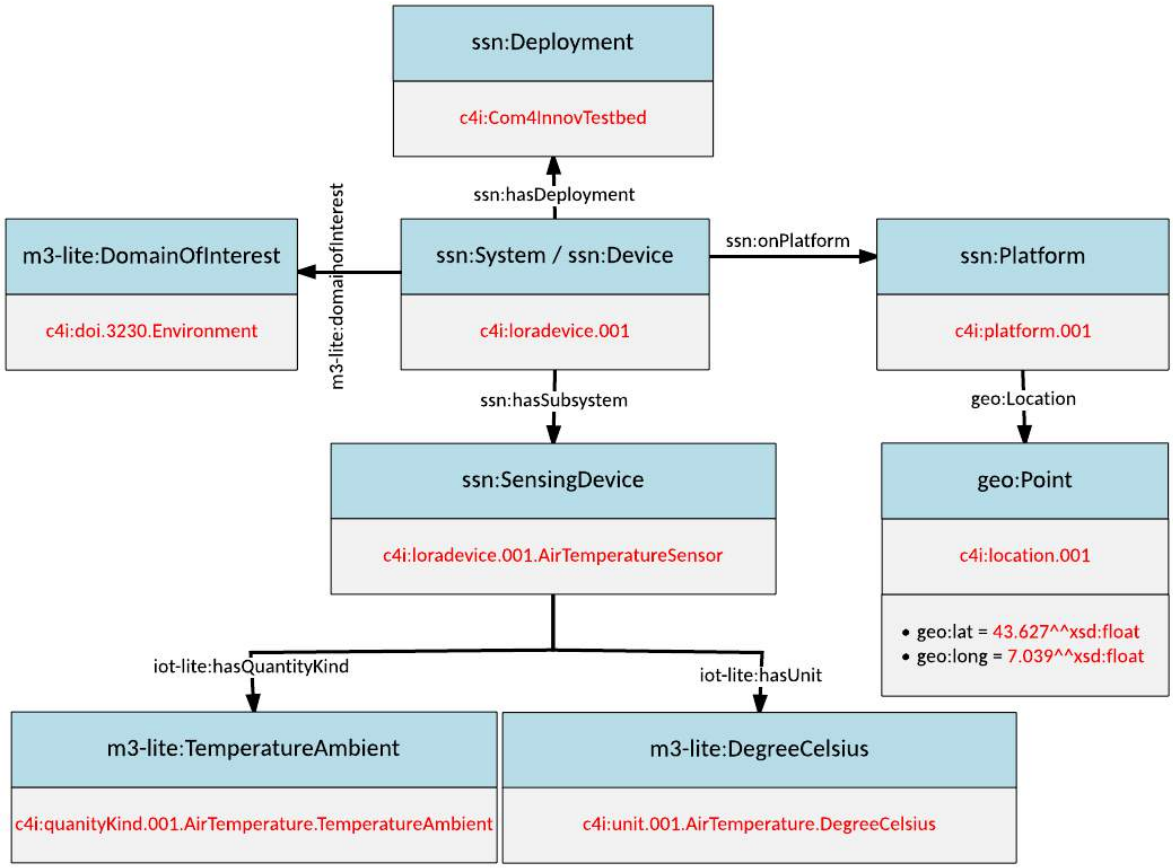

Figure 12: Com4Innov annotated resource description (graph)

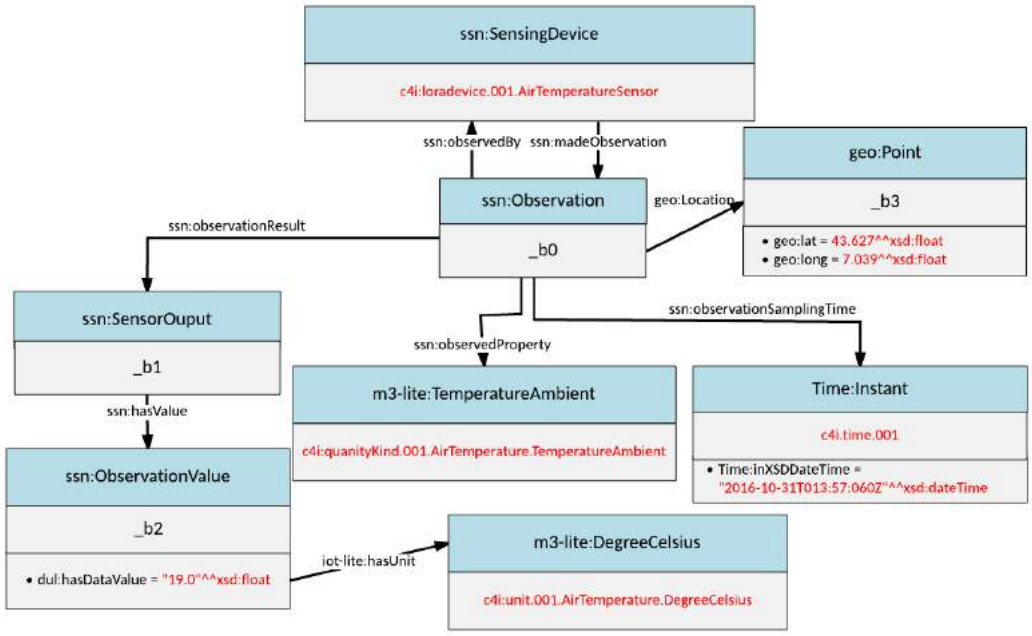

Figure 13: Com4Innov annotated observation (graph)

4.1.5 SoundCity

As a secondary and, an added value testbed, we introduce “SoundCity”. SoundCity leverages from environmental data sensed by mobile phone used by citizens. SoundCity testbed contains data about environmental phenomenon like noise and other phenomena like proximity, direction heading and average speed. In order to join to SoundCity testbed users need to install an application on their mobile phones. After installing the application, it enables microphone to be attached to the mobile phone in order to sense the noise around the phone. This observation is then sent to the central repository. In SoundCity testbed, a user can also associate his/herself to the FIESTA-IoT group using the provided web interface. Inria introduces this testbed. More information about the testbed can be accessed[10] and it can include devices from all over the world (any user who wants to join). A collection of data from the users within the FIESTA-IoT group is then sent to FIESTA-IoT platform using TPS. It is important to note that due to privacy reasons and the nature of the data, necessary precautions are taken when the user data is sent to FIESTA-IoT platform for experimentation. The SoundCity data object is a JSON object with information like noise level, timestamp, location information, proximity value and average speed (for a part of the JSON see below). This information is converted to FIESTA-IoT specific semantic model using the custom annotator.


```
{"client_timestamp":1477778306000,"crowdSourcedValue":{"_bias":-
13.2,"_location":{"accuracy":30,"altitude":0,"bearing":0,"hasBearing":false,"hasSpeed":fals
e,"latitude":48.8669749,"longitude":2.3537845,"speed":0,"time":1477575957258},"_referenceAv
erage":0,"_startTime":"2016-10-
29T23:58:26+0200","_unit":"dB(A)","_volumeAverage":26.182348,"_windowSize":0.125,"measureme
ntType":"automatic","proxSwitchNb":0,"proximity":false,"recogConfidence":0},"deviceId":"XXX
XXXXXXXXX","timestamp":1477780719023,"tz":3600000}
```


In the Figure 14 and Figure 15 we present resource graph and observation graph generated by the SoundCity annotator. Here we follow the convention of creating individuals for the classes used. The SoundCity namespace is http://www.soundcity.mobi/

[10] https://soundcity.mobi

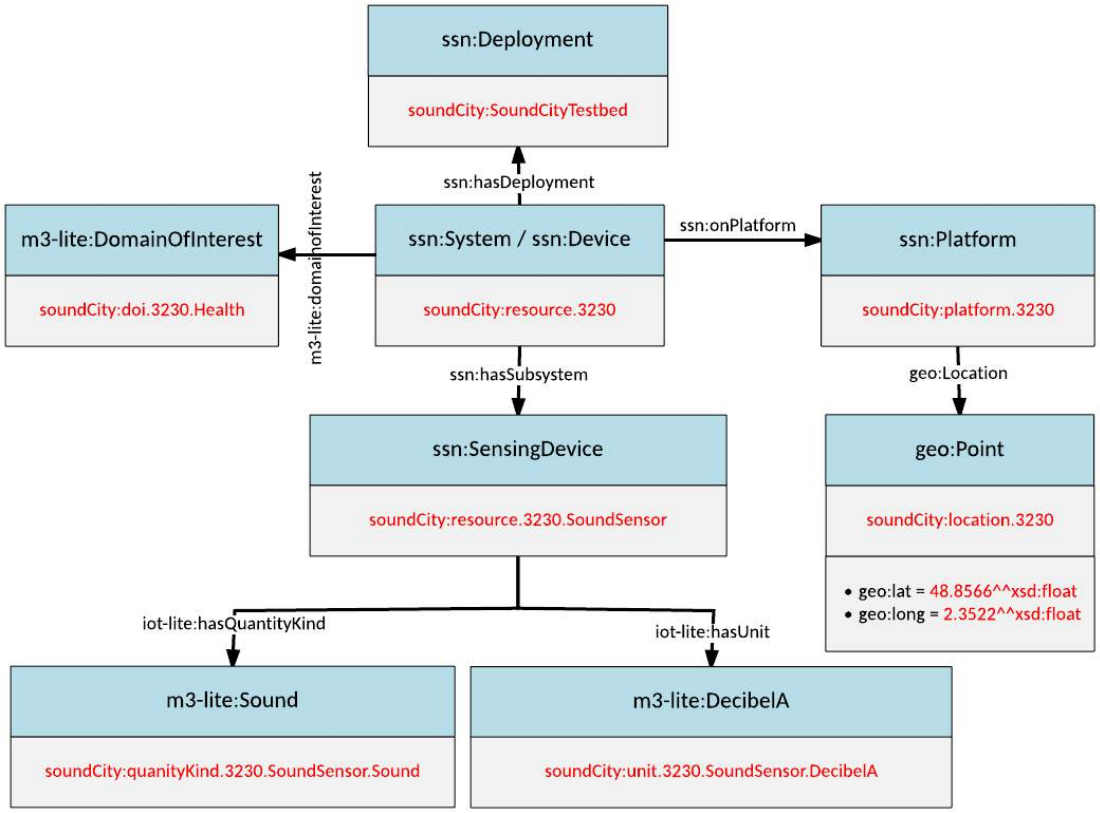

Figure 14: SoundCity annotated resource description (graph)

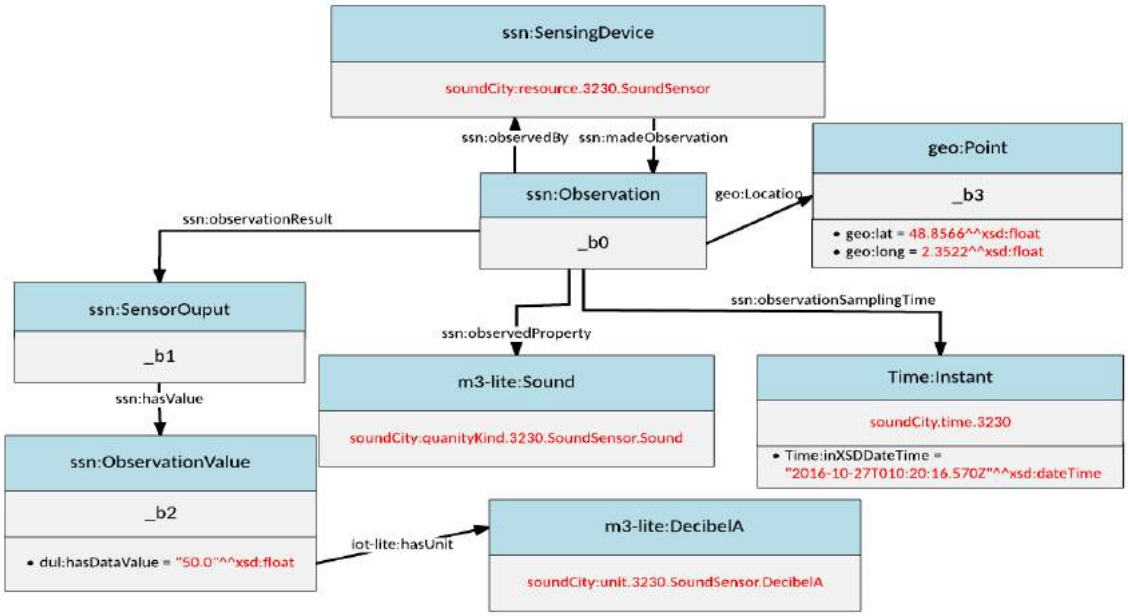

Figure 15: SoundCity annotated observation (graph)

4.2 FIESTA-IoT Validation Tool

In this section we present Validation Tool developed to perform validation on the annotations from the testbed. This tool is also referred to as Annotation Validation Tool (AVT).

4.2.1 Ontology and Data Validator

The Ontology and Data Validator is a tool that provides 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, a user is able to upload his/her ontology and annotated linked data in order to be validated against a reference ontology such as the W3C SSN ontology (see Figure 16). The validator detects syntactic and semantic issues if any, and produces a test report at the end of the process.

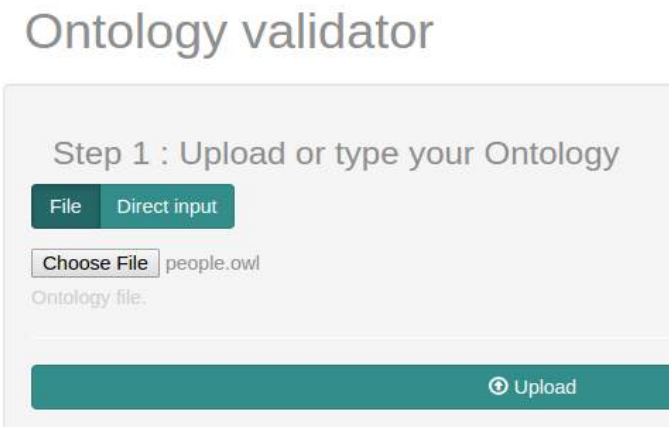

Figure 16: Uploading a file to be validated

Popular RDF serialization formats are accepted as input thanks to an integrated parser. Among these formats 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) are included. The resulting RDF is checked against RDF model and specifications before proceeding to 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 ontology (i.e. ontologies declared as prefix) or 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 Internationalized

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

Ontology validation is of interest when conducted in relation toward a reference ontology. Natively built-in reference ontologies include FIESTA-IoT ontology, SSN ontology, oneM2M base ontology and OWL2 ontology. User can also upload his own ontology to be considered as the validation reference ontology. He/She can also choose the types of error to be checked just in case he/she wishes to skip some. By default, all the types are checked.

A report is generated after the execution of tests (see Figure 18). If errors are detected, they are classified according to the four error types:

- Literal: include serialization format errors and untyped errors in literal resources
- Namespace and URI validation: include ill-formed URIs, namespace and prefix errors
- Predicate and class validation: predicate and class not-declared errors
- Semantic error: include all the semantic validation aspects mentioned previously

The four errors types are designed according to the semantic data validation requirements defined in WP6 as shown in Figure 17. Note that semantic error type includes the validation of “Type consistency”, “Domain and range type consistency” and “restriction on class”.

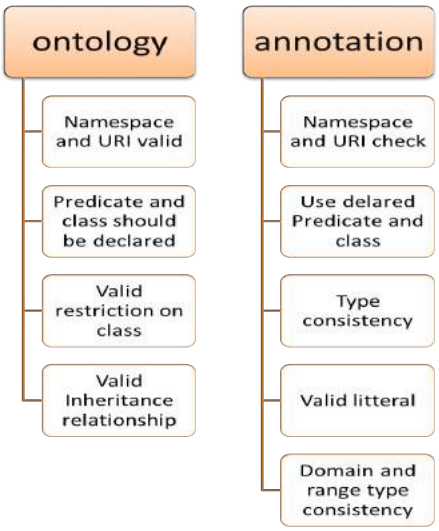

Figure 17: Semantic data validation requirements

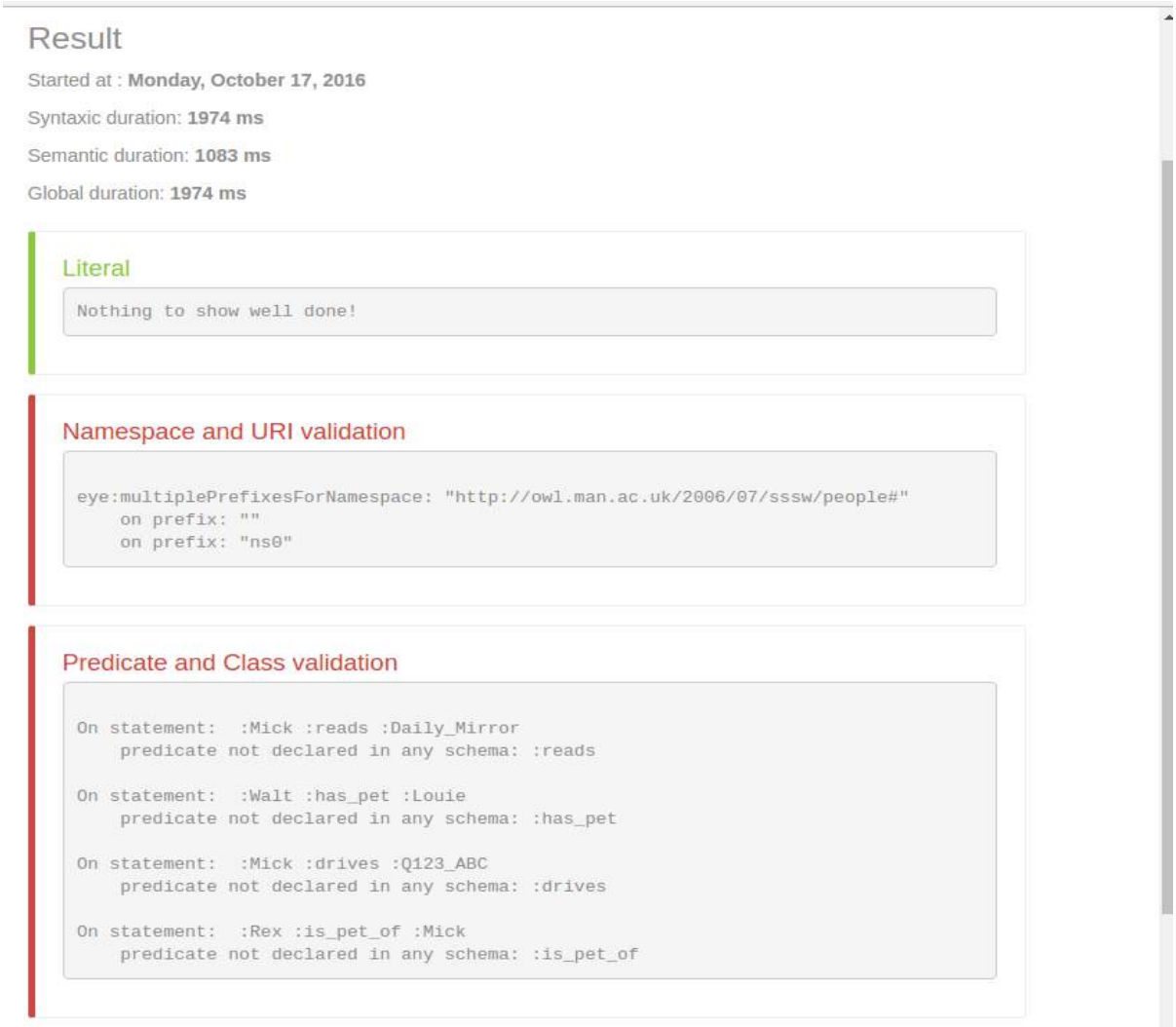

Figure 18: Validation report

The ontology validator service can be also invoked directly through HTTP requests, and returns the report in JSON format as the response, which makes the ontology validator easier to be integrated into any M2M workflow, including the FIESTA-IoT semantic data workflow. This is described in the following subsection 4.2.2.

4.2.2 Fiesta-IoT Testbed Validation

Semantic data produced by testbeds need to be validated before being pushed to the FIESTA-IoT semantic repositories (repositories within IoT-Service FG, see Figure 1). When a testbed joins the FIESTA-IoT federation, the FIESTA-IoT Meta-Cloud needs to make sure the semantic correctness of the resource descriptions submitted by the testbed. Every resource description needs to be validated before being registered in the Resource Registry, because if any resource is represented by an erroneous description, it will not be able to be inserted to the semantic resource graph, thus any observation made by this resource would not be able to link back to the resource. The observation data will not be usable by experiments conducted on the platform. Figure 19 shows that the validator is placed between the FIESTA-IoT platform and the testbed for validating every resource description. If a resource description is validated, it will be directly forwarded to the IoT-Service registry by the validator. If there is any error present in the resource description, the resource description will be intercepted by the validator and a report explaining the errors will be returned to the resource annotation provider.

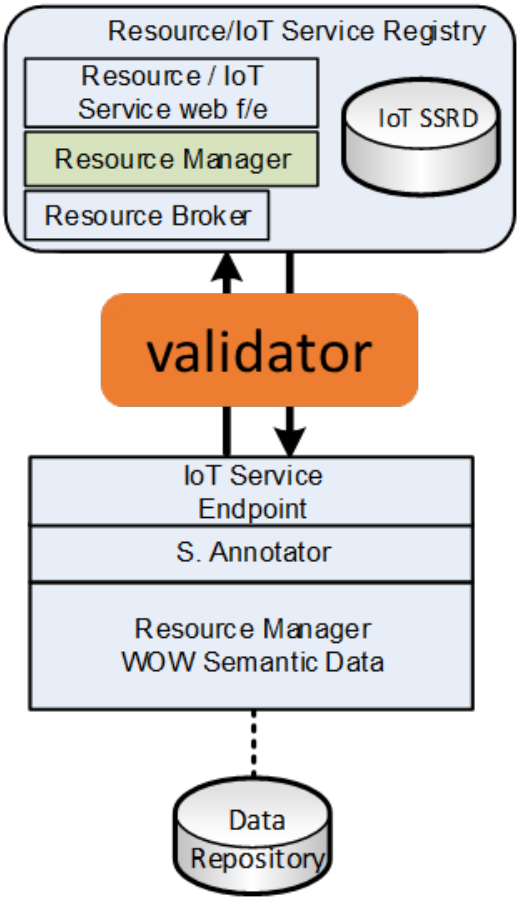

Figure 19: Validator validates all the resource descriptions

The validator takes several seconds to validate a piece of data. This delay should be taken into consideration while it is about validating the input observation data. In order to not impact too much the platform's performance, a sampling of observation data to be passed to the validator will be performed. The algorithm takes into consideration various parameters, such as the frequency of data input, the historical record of validation, the amount of data. Figure 20 shows a "validation sampling" module is integrated to the validator for observation data validation in order to find a balance point between the data correctness and the performance of the FIESTA-IoT platform. If the sampling module makes the decision to validate the current data, the workflow of validation is the same as in resource description validation case, which means validated data will be forwarded to the repository whereas the erroneous data will be intercepted and validation report will be returned. In case where the sampling module decides to trust the current data, the validator will simply plays the role of a forwarder towards the semantic data repository.

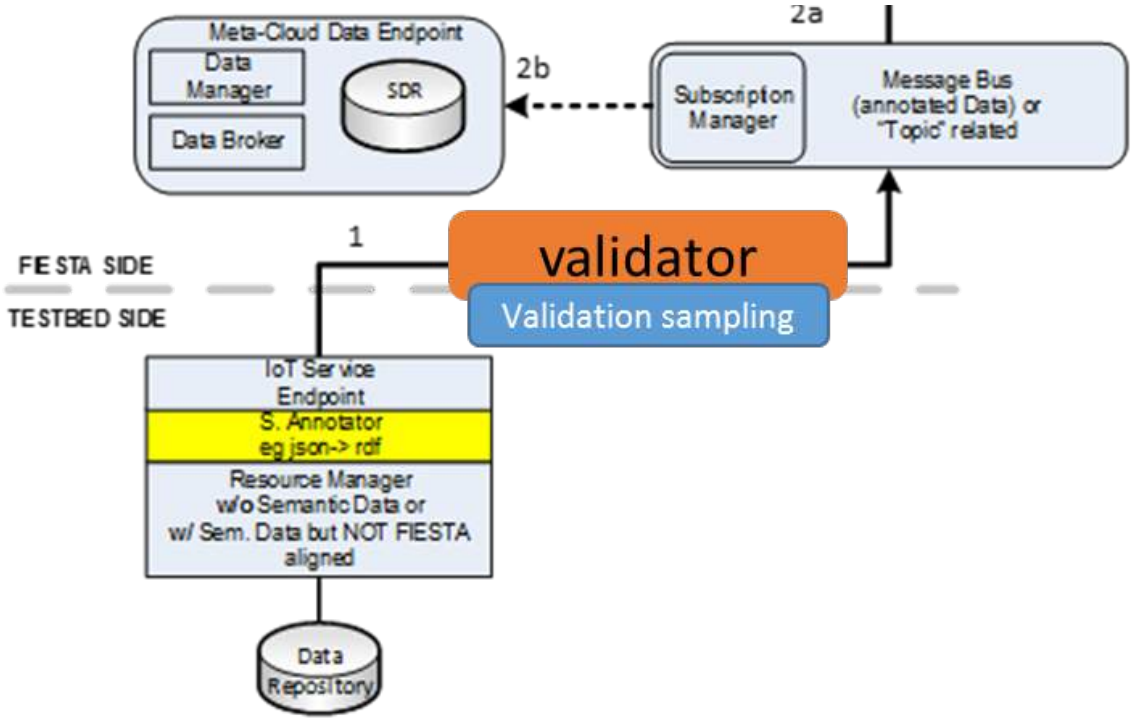

Figure 20: Validator validates observation data according to sampling algorithm

As the Task 3.3 is still ongoing, the development of the validator is still in progress and the performance of the validator and its impact on the platform is under assessment. Further we still have to decide the sampling algorithm. In the coming months, after a preliminary result of validator's performance assessment and some feedback from in-house experiments and testbeds on the FIESTA-IoT platform with the validator integrated, the validation sampling algorithm will be finalized and implemented in the sampling module.

5 FIESTA-IOT MOBILITY MANAGEMENT

In the previous version of this deliverable, we reported that there is no need for a dedicated mobility management component and FIESTA-IoT platform should handle updates to the resource registry. However, based on the ontology and the design of IoT-Services FG, queries that could query both observation and resource graph can be written. This removes the need to handle updates to the resource registry by using a kind of publish-subscribe system. Testbeds can push the observations using TPI module to IoT-Services observation graph. A sample SPARQL query to perform such search is hereafter:

```
Prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
Prefix iotlite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
Prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#>
Prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
Prefix time: <http://www.w3.org/2006/time#>
Prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#>
Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix spatial: <http://jena.apache.org/spatial#>
select ?s (max(?ti) as ?tim) ?val ?lat ?long ?dep
where {
        ?dev a ssn:Device .
        ?dev  ssn:hasSubSystem ?s .
        ?dev ssn:hasDeployment ?dep.
        ?s a ssn:SensingDevice .
        ?s ssn:madeObservation ?o.
        ?o ssn:observationSamplingTime ?t.
        ?o geo:location ?point.
        ?t time:inXSDDateTime ?ti.
        ?o ssn:observationResult ?or.
        ?or ssn:hasValue ?v.
        ?v dul:hasDataValue ?val.
        ?point geo:lat ?lat;
               geo:long ?long.
        {
               select  (max(?dt) As ?ti) ?s
               where {
                      ?o a ssn:Observation.
                      ?o ssn:observedBy ?s.
                      ?s ssn:madeObservation ?o.
                      ?o ssn:observedProperty ?qk.
                      ?qk a m3-lite:Sound.
                      ?o ssn:observationSamplingTime ?t.
                      ?t time:inXSDDateTime ?dt.
                      ?o geo:location ?point.
               }group by ?s
        }
} group by (?s) ?tim ?val ?lat ?long ?dep
```

Based on this need, a tool called Jena Spatial was identified that allows performing “spatial searched with SPARQL”. In order to perform such spatial searches a spatial index is created on the TDB store either by using Apache-Lucene[11] for smaller scale data or by using Apache-Solr[12] for large-scale search. As there is an observation graph where all the observations made from registered resources are stored, spatial index can be created on such graph. Note that in the observation graph, all observations have associated location and time information. With such a tool, which

[11] http://lucene.apache.org/core

[12] http://lucene.apache.org/solr/

is a link between observation and resource graph and spatio-temporal information, we are able to perform the following queries.

- Query 1: The following query provides all the sensors with their latest observation near to a given location

```
Prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
Prefix iotlite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
Prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#>
Prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
Prefix time: <http://www.w3.org/2006/time#>
Prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#>
Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix spatial: <http://jena.apache.org/spatial#>
select ?s (max(?ti) as ?tim) ?val ?lat ?long
where {
       ?o a ssn:Observation.
       ?o ssn:observedBy ?s.
       ?o ssn:observationSamplingTime ?t.
       ?o geo:location ?point.
       ?t time:inXSDDateTime ? ti.
       ?o ssn:observationResult ?or.
       ?or  ssn:hasValue ?v.
       ?v dul:hasDataValue ?val.
       ?point geo:lat ?lat;
              geo:long ?long.
       {
              select  (max(?dt) As ?ti) ?s
              where {
                     ?o a ssn:Observation.
                     ?o ssn:observedBy ?s.
                     ?point spatial:nearby (48.82 2.37 0.5 'km').
                     ?s ssn:madeObservation ?o.
                     ?o ssn:observedProperty ?qk.
                     ?qk a m3-lite:Sound.
                     ?o ssn:observationSamplingTime ?t.
                     ?t time:inXSDDateTime ?dt.
                     ?o geo:location ?point.
              }group by ?s
       }
} group by (?s) ?tim ?val ?lat ?long
```

- Query 2: The following query provides all the sensors with their latest observation near to a given location along with the deployment they belong to.

```
Prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
Prefix iotlite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
Prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#>
Prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
Prefix time: <http://www.w3.org/2006/time#>
Prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#>
Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix spatial: <http://jena.apache.org/spatial#>
select ?s (max(?ti) as ?tim) ?val ?lat ?long ?dep
where {
       ?dev a ssn:Device .
       ?dev  ssn:hasSubSystem ?s .
       ?dev ssn:hasDeployment ?dep.
       ?s a ssn:SensingDevice .
       ?s ssn:madeObservation ?o.
       ?o ssn:observationSamplingTime ?t.
       ?o geo:location ?point.
       ?t time:inXSDDateTime ?ti.
       ?o ssn:observationResult ?or.
```



```
        ?or  ssn:hasValue ?v.
        ?v dul:hasDataValue ?val.
        ?point geo:lat ?lat;
               geo:long ?long.
        {
               select  (max(?dt) As ?ti) ?s
               where {
                      ?o a ssn:Observation.
                      ?o ssn:observedBy ?s.
                      ?point spatial:nearby (48.82 2.37 0.5 'km').
                      ?s ssn:madeObservation ?o.
                      ?o ssn:observedProperty ?qk.
                      ?qk a m3-lite:Sound.
                      ?o ssn:observationSamplingTime ?t.
                      ?t time:inXSDDateTime ?dt.
                      ?o geo:location ?point.
               }group by ?s
        }
} group by (?s) ?tim ?val ?lat ?long ?dep
```

- Query 3:_Find all locations visited by a mobile sensor

```
Prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
Prefix iotlite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
Prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#>
Prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
Prefix time: <http://www.w3.org/2006/time#>
Prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#>
Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix spatial: <http://jena.apache.org/spatial#>
select ?s (max(?ti) as ?tim) ?val ?lat ?long
where {
        ?dev a ssn:Device .
        ?dev  ssn:hasSubSystem ?s .
        ?s ssn:madeObservation ?o.
        ?o ssn:observationSamplingTime ?t.
        ?o geo:location ?point.
        ?t time:inXSDDateTime ?ti.
        ?o ssn:observationResult ?or.
        ?or  ssn:hasValue ?v.
        ?v dul:hasDataValue ?val.
        ?point geo:lat ?lat;
               geo:long ?long .
} group by (?s) ?tim ?val ?lat ?long
```

Jena spatial allows queries requiring nearby, withinCircle, withinBox, at north of, south of, west of, east of, etc. In order to achieve such type of searches, an spatial index has to be created. Jena spatial uses two vocabularies (a) Assembler and (b) spatial. Using Assembler, the TDB store being used is referenced. While using spatial, a spatial index is created. As this task is ending in Month 24 we are currently working on setting up TDB in order it also includes spatial searches using Jena.
Further, we support using m3-lite taxonomy, direction and speed. In case a sensor is providing such type of information, it can be stored. However, it should be noted that these will be separate observations and again it will hold instantaneous values.

6 GUIDELINES FOR TESTBEDS AND BEST PRACTICES TO PUBLISH IOT DATA IN FIESTA-IOT

From all the lessons learnt throughout the design and implementation phases of the ontology, plus the subsequent annotators (one per testbed) and validator, a kind of guidelines that aim to guarantee the homogeneity between all the platforms that want to inject data to the federation is provided in this section.

First and foremost, it is worth highlighting the openness of the platform related to the serialization format of the data sets, when they are for describing a resource or outlining an observation. Thanks to semantic technologies, the core of the system, or said in other words, the triple stores (i.e. semantic databases) defined to keep the information, follow the well-known RDF data model. The key point of this is we are not stuck to a single serialization format, but open and compatible with a set of them: *Turtle*, *N-Triples*, *N-Quads*, *N3*, *JSON-LD*, *RDF/XML*, etc. This means that testbed providers can choose whichever format they feel more comfortable in.

We proceed below to enumerate a number of critical points that should be undertaken in order to respect the foundations settled down for the ontology.

- Before delving into the graph per se, it is extremely important to take into account that all the context (or prefixes) MUST be exactly the same, thus pointing out to the very same IRIs (Internationalized Resource Identifier). Otherwise, SPARQL queries will lead to unexpected behaviors. As an illustrative example (see Figure 21: SmartSantander context), we extract below the prefixes used for the SmartSantander annotators. As can be inferred, it embraces all the links to the ontologies whose concepts have been borrowed to shape the FIESTA-IoT ontology, beside other required prefixes, like XSD (XML Schema Definition) or the prefixes that defines SmartSantander' assets.

```
"@context": {
    "owl": "http://www.w3.org/2002/07/owl#",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "rdfa": "http://www.w3.org/ns/rdfa#",
    "xhv": "http://www.w3.org/1999/xhtml/vocab#",
    "xml": "http://www.w3.org/XML/1998/namespace",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "dul": "http://www.loa.istc.cnr.it/ontologies/DUL.owl#",
    "ssn": "http://purl.oclc.org/NET/ssnx/ssn#",
    "iot-lite": "http://purl.oclc.org/NET/UNIS/fiware/iot-lite#",
    "geo": "http://www.w3.org/2003/01/geo/wgs84_pos#",
    "time": "http://www.w3.org/2006/time#",
    "qu-unit": "http://purl.oclc.org/NET/ssnx/qu/unit#",
    "qu-quantity": "http://purl.oclc.org/NET/ssnx/qu/quantity#",
    "m3-lite": "http://purl.org/iot/vocab/m3-lite#",
    "sms-srd": "http://api.smartsantander.eu#",
    "sms-srd-href": "http://api.smartsantander.eu/v2/resources/"
  }
```

Figure 21: SmartSantander context information

- Another thing that it is worth to comment is the fact that two different realms that we are going to deal with (resources and observation) are connected under the **same graph**. This means that through a SPARQL originally addressed to resources, it would be possible to grasp data pertaining to observations and vice versa.

- **Only authorized testbed providers' accounts are allowed to write** any kind of data with regard to their testbeds. Hence, every message between the testbed side and the FIESTA-IoT core must be authenticated, thus users must gain their access before populating the triple stores.
- **Not all the resource description has to be registered at once**. Indeed, a testbed provider might need to "discretize" this registration phase into multiple stages. For that reason, IoT-service API permits to accordingly update any resource.
  However, if a resource is registered the resource description should include minimum number concepts at the time of registration. The required list of nodes include: `ssn:Deployment`, `ssn:Device` and `ssn:Platform` that defines the testbed that owns it, resources unique ID and the platform to which a resource's is attached to. Moreover, if e.g. a `ssn:SensingDevice` (resource) is being registered, it should go along with its respective `m3-lite:QuantityKind` and `m3-lite-lite:Unit`.
- Together with the previous point, it is essential to **keep the cardinality** of the annotations. In other words, the added graph must have a seamless connection and must not present any kind of inconsistency with regards to the already defined graph. This would jeopardize the whole triple store homogeneity by adding loose nodes that might not be retrieved in forthcoming queries.
- One of the differential factors that we have relied on when we have defined the ontology model is the use of linkable objects so as to fulfill the "***Web of Things***" paradigm. This outlook dictates that all the information has to be connected altogether. Said in layman's terms, the graph[13] can be navigated by simply clicking on the different individuals that populate the tree.
- Recalling that we rely on the `ssn:Device` **concept to define an asset** in the generic way of the term itself (being a sensor, a tag or an actuator). Then, it can host one or more `ssn:SensingDevice` (the `ssn:Sensor` concept is fostered for a slightly different purpose), `iot-lite:ActuatingDevice` or `iot-lite:TagDevice`. Attached to this base class, the **physical location**[14] through the `ssn:Platform` class should be specified.
- Lots of discussions have taken place in order to decide the most appropriate basic level for binding the concept of observation/measurement, namely through the `ssn:observedBy` object property (which is the actual property that connects the two "worlds" hinted above). From the different approaches undergone by the "in-house" testbeds, there were two main alternatives that stand out over the rest: we could have stuck to the `ssn:Device` level, linking a number of observations/measurements to that device; however, by taking this path we would lose the granularity to individually handle every observation. The limitations of this former case have led us to get to the immediately lower level, attaching each observation to its corresponding `ssn:SensingDevice` (actually, through its subclass `ssn:Sensor`), where every `ssn:Observation` must be linked (through `ssn:observedBy`) to a sensor.

[13] When it comes to the SmartSantander case, `ssn:Device`, `ssn:SensingDevice` and `iot-lite:Service` class are foreseen to be "linkable", thus they will dynamically address to their respective connection points.

[14] Unless it is a mobile device, whose description varies a little bit and registering its location is not a key necessity.

- After having defined the sensing entities, it is time to talk about their capabilities, that is to say, their underlying physical phenomena (a.k.a. quantity kinds) and units of measurement (both of them are to be heavily tied). For the sake of visibility, **every sensor will instance an individual** for each of its underlying quantity kinds and phenomena. Figure 22 exemplifies the particular case of one of the SmartSantander sensors (i.e. `ssn:SensingDevice`) that embraces an air thermometer which annotates its data in Degree Celsius. As it can be seen, instead of having a unique individual for all `m3-lite:AirTemperature` cases, we define a different one for each case.

```
{
  "@id": "sms-srd:urn:x-iot:smartsantander:u7jcfa:t72.temperature-ambient.sensor",
  "@type": "ssn:SensingDevice",
   "iot-lite:exposedBy": {
     "@id":"sms-srd:service.urn:x-iot:smartsantander:u7jcfa:t72.temperature-ambient.sensor"
    },
   "iot-lite:hasQuantityKind": {
    "@id":"sms-srd:quantity.urn:x-iot:smartsantander:u7jcfa:t72.temperature-ambient.sensor"
    },
    "iot-lite:hasUnit": {
      "@id":"sms-srd:unit.urn:x-iot:smartsantander:u7jcfa:t72.temperature-ambient.sensor"
    }
},
{
  "@id":"sms-srd:quantity.urn:x-iot:smartsantander:u7jcfa:t72.temperature-ambient.sensor",
  "@type": "m3-lite:AirTemperature"
},
{
  "@id": "sms-srd:unit.urn:x-iot:smartsantander:u7jcfa:t72.temperature-ambient.sensor",
  "@type": "m3-lite:DegreeCelsius"
}
```

Figure 22: Use of individuals to instance quantity kinds and units

- When an observation is annotated, as can be seen from Figure 8, Figure 10, Figure 13, and Figure 14, the root node belongs to `ssn:Observation`. In order to link it to the resource descriptions' realm, we have come up with the `ssn:ObservedBy` object property.
- An added value that can be brought in a semantic description is the explicit introduction of an **IoT service endpoint** through which externals can directly get data. By invoking these services, testbeds' resource and data managers will internally return the corresponding data (e.g. the last measurement harvested by a particular `ssn:SensingDevice`).However, as hinted at the very beginning of this paragraph, it is an optional nice-to-have feature.
- In order to reduce overhead, it is highly recommended, if possible, to **avoid the inclusion of unnecessary information** (e.g. not abusing metadata). If we stick to only save the essential stuff, the system performance would be much better than that if the graph was flooded with pointless nodes.

7 CONCLUSION AND FUTURE WORK

This deliverable provides an update to the previous version of this deliverable (D3.1.1). In this deliverable we report proceedings of two tasks within WP3. These tasks are task 3.1 and task 3.3. Out of the two above-mentioned tasks, Task 3.1 ends in month 21 while Task 3.3 continues until month 24. As this deliverable comes before the end of task 3.3, we report on-going development within the framework of task 3.3.

We build semantic model that each testbed would have to comply with in order to be a part of federation provided by FIESTA-IoT. As we have already reported, the semantic model is targeted towards the description of Sensors and the Observations provided by them. Based on the analysis and requirements, we have performed a number of updates to the semantic model and we envision more updates in the near future due to the involvement of more testbeds as a consequence of the results of the open calls. Citing from previous deliverable we want to again emphasize on "*that the taxonomy and FIESTA-IoT ontology will be a living entity that might be updated with all the new elements coming from potential new testbeds that become part of the FIESTA-IoT federation*". Nevertheless, current updates are targeted towards mainly 5 aspects. These aspects are:

(a) Minimization of usage of DUL ontology: we replace `dul:TimeInverval` with `time:Instant` because most of the observations made available are instantaneous observations. We also justify the use of `time:Instant` even when the sensors produce values over a time interval. This is justified as, during a time interval, multiple observations are generated. Thus, a testbed is required to provide each of these observations as instantaneous values. Similarly, if only one observation is generated over a period of time, we refer it as sampling frequency and thus the observation is still considered as an instantaneous value.
(b) Addition of more datatypes to the `dul:hasDataValue` in order to support needs of different testbeds.
(c) Addition of oneM2M concepts as equivalent classes within FIESTA-IoT ontology so that oneM2M compliant testbeds are able to join easily the FIESTA-IoT federation. These additions are done envisioning a huge impact of oneM2M ontology.
(d) Addition of more subclasses within concepts like Sensor, QuantityKind and Units to support needs of current in-house testbeds.
(e) Addition of more concepts within m3-lite taxonomy such as `m3-lite:Source` and `m3-lite:MeasurementType`.

Apart from the above-mentioned updates to the ontology and its supporting taxonomy, we also report different reference annotators built to facilitate understanding of how data is being stored in IoT-Service FG. This is supported by conventions and guidelines that must be followed by testbed to publish data to IoT-Services FG. These conventions and guidelines help to achieve common understanding and help to formulate uniform queries. Such guidelines should be used by testbeds to annotate their resources.

All observations and resource annotations have to pass the validation process before they are added to the semantic repository. The validator checks for missing

concepts, cardinalities, syntactic errors and semantic errors. Once the annotations are validated they are sent to semantic repository.

Based on the semantic data that supports `geo:location`, mobility of the resources can be inferred by querying the semantic repository. Once the data is obtained after querying, experimenters can infer mobility. We provide sample use-cases and queries to show how different types of spatial searches can be performed. We leverage from open source Jena spatial library that creates spatial index on the geo-localized data.

However, there are still several open issues regarding Task 3.3 in which we are working on. These issues include: (a) finalizing validator integration in the FIESTA-IoT workflow and (b) finalizing the integration of Jena Spatial.

8 REFERENCES

[1] FIESTA-IoT, “Deliverable 3.1.1: Semantic models for testbeds, interoperability and mobility support and best practices,” 2016.

[2] FIESTA-IoT, “Deliverable 2.2: Analysis of IoT Platforms and Testbeds,” 2015.

[3] FIESTA-IoT, “Deliverable 2.4: FIESTA-IoT Meta Cloud Architecture,” 2015.

[4] FIESTA-IoT, “Deliverable 2.1: Stakeholders Requirements.”

[5] oneM2M, “TS-0012 oneM2M Base Ontology,” 2016.

[6] S. Mouzakitis, D. Papaspyros, S. Koussouris, J. Psarras, L. Farid, A. Schramm, B. Kapourani, A. Zafeiropoulos, E. Fotopoulou, K. Thellmann, J. Attard, F. Orlandi, N. Zanetti, F. Molinari, and C. Rubattino, “LINDA Project Deliverable1.3: Linked Data Components and Tools SotA and Business Scenarios for Linked Data Utilization - Updated Version 1.1,” Dec. 2015.

[7] FIESTA-IoT, “Deliverable 3.2.1: Specification and implementation of common Testbed interfaces,” 2016.

[8] FIESTA-IoT, “Deliverable 4.2.1: Techniques for Secure Access and Reservation of Resources,” 2016.

APPENDIX I – "IN-HOUSE" TESTBED QUANTITY KINDS AND UNITS

Below different types of QuantityKinds (QK) available with different in-house Testbeds along with the units are tabulated. These tables also contain M3-lite equivalent QKs and units for the Testbed defined QKs and units.

Table 6: SmartSantander Testbed QuantityKinds and Units

SmartSantander QK	M3-lite QK	Unit	M3-lite Unit
activePower	ActivePower	watt	Watt
atmosphericPressure	AtmosphericPressure	bar	Bar
batteryLevel	BatteryLevel	percent	Percent
chemicalAgentAtmosphericConcentration:03	ChemicalAgentAtmosphericConcentration03	microgramPerCubicMetre	MicrogramPerCubicMetre
chemicalAgentAtmosphericConcentration:airParticles	ChemicalAgentAtmosphericConcentrationAirParticles	milligramPerCubicMetre	MilligramPerCubicMetre
chemicalAgentAtmosphericConcentration:CO	ChemicalAgentAtmosphericConcentrationCO	scale	Scale
chemicalAgentAtmosphericConcentration:NO2	ChemicalAgentAtmosphericConcentrationNO2	microgramPerCubicMetre	MicrogramPerCubicMetre
direction:heading	DirectionHeading	index	Index
electricCurrent	ElectricCurrent	ampere	Ampere
electricField:1800mhz	ElectricField1800mhz	millivoltPerMetre	MillivoltPerMetre
electricField:2100mhz	ElectricField2100mhz	millivoltPerMetre	MillivoltPerMetre
electricField:2400mhz	ElectricField2400mhz	millivoltPerMetre	MillivoltPerMetre
electricField:900mhz	ElectricField900mhz	millivoltPerMetre	MillivoltPerMetre
electricPotential	ElectricPotential	volt	Volt
fillLevel:gasTank:1	FillLevelGasTank:1	percent	Percent
fillLevel:gasTank:2	FillLevelGasTank:2	percent	Percent
fillLevel:wasteContainer	FillLevelWasteContainer	percent	Percent
fuelComsumption:total	FuelComsumptionTotal	litre	Litre
fuelConsumption:instantaneous	FuelConsumptionInstantaneous	litrePer100Kilometres	LitrePer100Kilometres
illuminance	Illuminance	lux	Lux
mass	Mass	kilogram	Kilogram
mileage:distanceToService	MileageDistanceToService	kilometre	Kilometre
mileage:total	MileageTotal	metre	Metre
motionState:vehicle	MotionStateVehicle	index	Index
position:altitude	PositionAltitude	metre	Metre
position:latitude	PositionLatitude	degreeAngle	DegreeAngle
position:longitude	PositionLongitude	degreeAngle	DegreeAngle

presenceState:driverCard:1	PresenceStateDriverCard1	index	Index
presenceState:driverCard:2	PresenceStateDriverCard2	index	Index
presenceState:parking	PresenceStateParking	index	Index
rainfall	Rainfall	millimetrePerHour	MillimetrePerHour
reactivePower	ReactivePower	var	Var
relativeHumidity	RelativeHumidity	percent	Percent
roadOccupancy	RoadOccupancy	percent	Percent
rotationalSpeed:engine	RotationalSpeed:engine	revolutionPerMinute	RevolutionPerMinute
soilMoistureTension	SoilMoistureTension	centibar	Centibar
solarRadiation:par	SolarRadiationPar	wattPerSquareMetre	WattPerSquareMetre
soundPressureLevel:ambient	SoundPressureLevelAmbient	decibelA	DecibelA
speed:average	SpeedAverage	kilometrePerHour	KilometrePerHour
speed:instantaneous	SpeedInstantaneous	metrePerSecond	MetrePerSecond
speed:median	SpeedMedian	kilometrePerHour	KilometrePerHour
temperature:ambient	TemperatureAmbient	degreeCelsius	DegreeCelsius
temperature:engine	TemperatureEngine	degreeCelsius	DegreeCelsius
temperature:soil	TemperatureSoil	degreeCelsius	DegreeCelsius
temperature:wasteContainer	TemperatureWasteContainer	degreeCelsius	DegreeCelsius
timeRelatedState:driver:1	TimeRelatedStateDriver:1	index	Index
timeRelatedState:driver:2	TimeRelatedStateDriver:2	index	Index
timestamp	Timestamp	dimensionless	Dimensionless
trafficCongestion	TrafficCongestion	index	Index
trafficIntensity	TrafficIntensity	vehiclePerMinute	VehiclePerMinute
vehicleOverspeedState	VehicleOverspeedState	index	Index
windDirection	WindDirection	degreeAngle	DegreeAngle
windSpeed	WindSpeed	kilometrePerHour	KilometrePerHour
workingState:driver:1	WorkingStateDriver1	index	Index
workingState:driver:2	WorkingStateDriver2	index	Index

Table 7: Com4Innov's Testbed QuantityKinds and Units

Com4Innov Testbed QK	M3-lite QK	Unit	M3-lite Unit
AirTemperature	TemperatureAmbient	Celsius	DegreeCelsius
BoardTemperature	TemperatureBoard	Celsius	DegreeCelsius
Delta Dewpoint	DeltaDewPoint	Natural Number	Index
Dewpoint	DewPoint	Celsius	DegreeKelvin
OutputVoltage	ElectricPotential	Volt	Volt
Radiation	SolarRadiation	W/m^2	WattPerSquareMetre
RelativeHumidity	RelativeHumidity	Percentage	Percent
SoundLevel	SoundPressureLevelAmbient	dB	Decibel
Acc_InitialERABEstabSuccRate	Acc_InitialERABEstabSuccRate	Percentage	Percent
Acc_InitialERABSetupSuccRate	Acc_InitialERABSetupSuccRate	Percentage	Percent
Acc_RrcConnSetupSucRate	Acc_RrcConnSetupSuccRate	Percentage	Percent
Acc_S1SigEstabSuccRate	Acc_S1SigEstabSuccRate	Percentage	Percent
Int_DlLatency	Int_DlLatency	ms	MilliSecond
Int_DlThroughput_kbps	Int_DlThroughput_kbps	kbps	KilobitsPerSecond
Int_UlPacketLoss	Int_UlPacketLoss	Percentage	Percent
Int_ UlThroughput_kbps	Int_ UlThroughput_kbps	kbps	KilobitsPerSecond
Mob_HoExecSuccRate	Mob_HoExecSuccRate	Percentage	Percent
Mob_HoPrepSuccRate	Mob_HoPrepSuccRate	Percentage	Percent
Mob_MobilitySuccRate	Mob_MobilitySuccRate	Percentage	Percent
Res_AverageLicConnectedUsers	Res_AverageLicConnectedUsers	Number	Dimensionless
Ret_ERabDrop	Ret_ERabDrop	Percentage	Percent

Table 8: KETI's Testbed QuantityKinds and Units

KETI's Testbed QK	M3-lite QK	Unit	M3-lite Unit
Office Temperature	BuildingTemperature	Celsius	DegreeCelsius
Office Humidity	RelativeHumidity	Percentage	Percent
Office Illumination	Illuminance	Lux	Lux
Office CO2	ChemicalAgentAtmosphericConcentrationCO2	ppm	PPM
Electricity	ActivePower	Watt	Watt
Parking	PresenceStateParking	Boolean	Index
User Occupancy	PresenceStatePeople	Boolean	Index

Table 9: UNIS's Testbed QuantityKind and Units

UNIS's Testbed QK	M3-lite QK	Unit	M3-lite Unit
Temperature	BuildingTemperature	Celsius	DegreeCelius
Office Humidity	RelativeHumidity	Percentage	Percent
Office Illumination	Illuminance	Lux	Lux
Office CO2	ChemicalAgentAtmosphericConcentrationCO2	ppm	PPM
Electricity	ActivePower	Watt	Watt
User presence	PresenceStatePeople	Boolean	Index

Table 10: SoundCity Testbed QuantityKind and Units

SoundCity Testbed QK	M3-lite QK	Unit	M3-lite Unit
sound	Sound	dBa	DecibelA
proximity	Proximity	boolean	Others
bearing	DirectionHeading	degree	DegreeAngle
speed	SpeedAverage	m/s	MeterPerSecond

APPENDIX II – SAMPLE USAGE OF ONTOLOGY

In this appendix, we provide sample annotations generated by SmartSantander, SamatICS and SoundCity Testbed for resources as well as for observations.

Sample Resource Annotation from the SmartSantander Testbed

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdfa: <http://www.w3.org/ns/rdfa#> .
@prefix xhv: <http://www.w3.org/1999/xhtml/vocab#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .
@prefix sms-srd: <http://api.smartsantander.eu#> .
@prefix sms-srd-href: <http://api.smartsantander.eu/v2/resources/> .

sms-srd:SmartSantanderTestbed a ssn:Deployment .

sms-srd:platform.urn:x-iot:smartsantander:u7jcfa:t3230 a ssn:Platform ;
       geo:location sms-srd:location.urn:x-iot:smartsantander:u7jcfa:t3230 .

sms-srd:location.urn:x-iot:smartsantander:u7jcfa:t3230 a geo:Point ;
       geo:lat "43.47171"^^xsd:float ;
       geo:long "-3.80014"^^xsd:float .

sms-srd-href:urn:x-iot:smartsantander:u7jcfa:t3230 a ssn:Device ;
       ssn:hasDeployment sms-srd:SmartSantanderTestbed ;
       ssn:hasSubSystem sms-srd:urn:x-iot:smartsantander:u7jcfa:t3230.temperature-
ambient.sensor , sms-srd:urn:x-iot:smartsantander:u7jcfa:t3230.illuminance.sensor ;
       ssn:onPlatform sms-srd:platform.urn:x-iot:smartsantander:u7jcfa:t3230 .

sms-srd:urn:x-iot:smartsantander:u7jcfa:t3230.temperature-ambient.sensor a m3-
lite:AirThermometer ;
       iot-lite:exposedBy sms-srd:service.urn:x-
iot:smartsantander:u7jcfa:t3230.temperature-ambient.sensor ;
       iot-lite:hasQuantityKind sms-srd:quantity.urn:x-
iot:smartsantander:u7jcfa:t3230.temperature-ambient.sensor ;
       iot-lite:hasUnit sms-srd:unit.urn:x-iot:smartsantander:u7jcfa:t3230.temperature-
ambient.sensor .

sms-srd:urn:x-iot:smartsantander:u7jcfa:t3230.illuminance.sensor a m3-lite:LightSensor ;
       iot-lite:exposedBy sms-srd:service.urn:x-
iot:smartsantander:u7jcfa:t3230.illuminance.sensor ;
       iot-lite:hasQuantityKind sms-srd:quantity.urn:x-
iot:smartsantander:u7jcfa:t3230.illuminance.sensor ;
       iot-lite:hasUnit sms-srd:unit.urn:x-
iot:smartsantander:u7jcfa:t3230.illuminance.sensor .

sms-srd:service.urn:x-iot:smartsantander:u7jcfa:t3230.temperature-ambient.sensor a iot-
lite:Service ;
       iot-lite:endpoint "https://api-
dev.smartsantander.eu:10443/v2/measurements/temperature:ambient/urn/urn:x-
iot:smartsantander:u7jcfa:t3230/last?format=jsonld"^^xsd:anyURI ;
       iot-lite:interfaceType "REST"^^xsd:string .

sms-srd:quantity.urn:x-iot:smartsantander:u7jcfa:t3230.temperature-ambient.sensor a m3-
lite:AirTemperature .

sms-srd:unit.urn:x-iot:smartsantander:u7jcfa:t3230.temperature-ambient.sensor a m3-
lite:DegreeCelsius .
```



```
sms-srd:service.urn:x-iot:smartsantander:u7jcfa:t3230.illuminance.sensor a iot-lite:Service
;
       iot-lite:endpoint "https://api-
dev.smartsantander.eu:10443/v2/measurements/illuminance/urn/urn:x-
iot:smartsantander:u7jcfa:t3230/last?format=jsonld"^^xsd:anyURI ;
       iot-lite:interfaceType "REST"^^xsd:string .

sms-srd:quantity.urn:x-iot:smartsantander:u7jcfa:t3230.illuminance.sensor a m3-
lite:Illuminance .

sms-srd:unit.urn:x-iot:smartsantander:u7jcfa:t3230.illuminance.sensor a m3-lite:Lux .
```

Sample Observation Annotation from the SmartSantander Testbed

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdfa: <http://www.w3.org/ns/rdfa#> .
@prefix xhv: <http://www.w3.org/1999/xhtml/vocab#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#> .
@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .
@prefix sms-srd: <http://api.smartsantander.eu#> .

_:b0 a ssn:Observation ;
       ssn:observationResult _:b1 ;
       ssn:observationSamplingTime _:b2 ;
       ssn:observedBy sms-srd:urn:x-iot:smartsantander:u7jcfa:t3230.AirTemperature.sensor ;
       ssn:observedProperty sms-srd:quantity.urn:x-
iot:smartsantander:u7jcfa:t3230.AirTemperature.sensor ;
       geo:location _:b3 .

_:b1 a ssn:SensorOutput ;
       ssn:hasValue _:b4 .

_:b2 a time:Instant ;
       time:inXSDDateTime "2016-10-27T08:05:32.620Z"^^xsd:dateTime .

sms-srd:quantity.urn:x-iot:smartsantander:u7jcfa:t3230.AirTemperature.sensor a m3-
lite:AirThermometer .

_:b3 a geo:Point ;
       geo:lat "4.346769E1"^^xsd:float ;
       geo:long "-3.81217E0"^^xsd:float .

_:b4 a ssn:ObservationValue ;
       iot-lite:hasUnit sms-srd:unit.urn:x-
iot:smartsantander:u7jcfa:t3230.AirTemperature.sensor ;
       dul:hasDataValue "9.95E0"^^xsd:float .

sms-srd:unit.urn:x-iot:smartsantander:u7jcfa:t3230.AirTemperature.sensor a m3-
lite:DegreeCelsius .
```

Sample Resource Annotations from the SmartICS Testbed

```
prefix :      <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .
@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .
@prefix qu:    <http://purl.org/NET/ssnx/qu/qu#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
```



```
@prefix fiesta-res: <http://platform.fiesta-iot.eu/srd/registry/poc/> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .
@prefix ssn:   <http://purl.oclc.org/NET/ssnx/ssn#> .
@prefix sc:    <http://iot.ee.surrey.ac.uk/smartcampus#> .
@prefix geo:   <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml:   <http://www.w3.org/XML/1998/namespace> .
@prefix time:  <http://www.w3.org/2006/time#> .

m3-lite:DegreeCelsius
        a       qu:Unit .

sc:UK-GU-UNIS-ICS-Desk1
        a                          geo:Point ;
        iot-lite:RelativeLocation  "Desk1" ;
        geo:lat                    "51.24332260763518" ;
        geo:long                   "-0.5932226718630889" .

sc:SmartCampus  a  ssn:System .

m3-lite:Temperature  a  qu:QuantityKind .

<http://platform.fiesta-iot.eu/srd/registry/poc/urn:x-iot:smartcampus:smart-ics:n001>
        a                       ssn:Device ;
        iot-lite:isSubSystemOf  sc:smart-ics ;
        ssn:hasSubSystem        <http://platform.fiesta-iot.eu/srd/registry/poc/urn:x-
iot:smartcampus:smart-ics:n001.Temperature> ;
        ssn:onPlatform          sc:Desk1 .

<http://platform.fiesta-iot.eu/srd/registry/poc/urn:x-iot:smartcampus:smart-
ics:n001.Temperature>
        a                         ssn:SensingDevice ;
        iot-lite:exposedBy        <http://iot.ee.surrey.ac.uk/smartcampus#urn:x-
iot:smartcampus:smart-ics:n001ServiceTemperature> ;
        iot-lite:hasQuantityKind  m3-lite:Temperature ;
        iot-lite:hasUnit          m3-lite:DegreeCelsius ;
        iot-lite:isSubSystemOf    <http://platform.fiesta-iot.eu/srd/registry/poc/urn:x-
iot:smartcampus:smart-ics:n001> .

sc:smart-ics  a                 ssn:System ;
        iot-lite:isSubSystemOf  sc:SmartCampus .

sc:Desk1  a           ssn:Platform ;
        geo:location  sc:UK-GU-UNIS-ICS-Desk1 .

<http://iot.ee.surrey.ac.uk/smartcampus#urn:x-iot:smartcampus:smart-
ics:n001ServiceTemperature>
        a                  iot-lite:Service ;
        iot-lite:endpoint  "http://mu.tlmat.unican.es:8080/smart-ics/iot-
node/1/temperature"
```


Sample Resources Annotation from SoundCity Testbed

```
@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .
@prefix qu:    <http://purl.org/NET/ssnx/qu/qu#> .
@prefix ns:    <http://creativecommons.org/ns#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix txn:   <http://lod.taxonconcept.org/ontology/txn.owl#> .
@prefix base-ontology: <http://www.onem2m.org/ontology/Base_Ontology/base_ontology-v0_9_0#>
.
@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .
@prefix ssn:   <http://purl.oclc.org/NET/ssnx/ssn#> .
@prefix geo:   <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix soundCity: <http://www.soundcity.mobi/> .
@prefix terms: <http://purl.org/dc/terms/> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix base_ontology: <http://www.onem2m.org/ontology/Base_Ontology/base_ontology#> .
@prefix dul:   <http://www.loa.istc.cnr.it/ontologies/DUL.owl#> .
@prefix time:  <http://www.w3.org/2006/time#> .
@prefix vann:  <http://purl.org/vocab/vann/> .
@prefix dc:    <http://purl.org/dc/elements/1.1/> .
@prefix qudt:  <http://data.qudt.org/qudt/owl/1.0.0/unit.owl#>

geo:location   a             owl:ObjectProperty ;
               rdfs:range    geo:Point .

ssn:Platform  a  owl:Class .

soundCity:quantityKind.1001.SpeedSensor     a       m3-lite:SpeedAverage .

m3-lite:Health  a  owl:Class .

soundCity:doi.1001.City      a       m3-lite:City .

soundCity:resource.1001.SpeedSensor  a                          ssn:SensingDevice ;
                     iot-lite:hasQuantityKind  soundCity:quantityKind.1001.SpeedSensor ;
                     iot-lite:hasUnit          soundCity:unit.1001.SpeedSensor .

iot-lite:hasQuantityKind     a                       owl:ObjectProperty ;
                             owl:equivalentProperty  base_ontology:hasThingProperty .

ssn:attachedSystem    a              owl:ObjectProperty ;
                      rdfs:domain    ssn:Platform ;
                      owl:inverseOf  ssn:onPlatform .

ssn:Deployment  a  owl:Class .

iot-lite:isMobile     a              owl:DatatypeProperty ;
                      rdfs:domain    ssn:Platform ;
                      rdfs:range     xsd:boolean .

m3-lite:domainOfInterest     a              owl:ObjectProperty ;
                             rdfs:domain    ssn:Device .

ssn:Device  a   owl:Class .

m3-lite:SpeedAverage  a  owl:Class .

ssn:hasSubSystem  a  owl:ObjectProperty .

soundCity:resource.1001   a                  ssn:Device ;
                          ssn:hasDeployment  soundCity:SoundCityTestbed ;
                          ssn:hasSubSystem   soundCity:resource.1001.SoundSensor ,
```



```
                                        soundCity:resource.1001.AccelerometerSensor ,
                                        soundCity:resource.1001.SpeedSensor ,
                                        soundCity:resource.1001.ProximitySensor ;
                        ssn:onPlatform     soundCity:platform.1001 ;
                        m3-lite:domainOfInterest  soundCity:doi.1001.Health ,
                                                  soundCity:doi.1001.Pollution ,
                                                  soundCity:doi.1001.City ,
                                                  soundCity:doi.1001.Transportation .

m3-lite:Others  a  owl:Class .

soundCity:quantityKind.1001.AccelerometerSensor    a       m3-lite:DirectionHeading .

soundCity:unit.1001.SoundSensor     a       m3-lite:DecibelA .

m3-lite:City  a  owl:Class .

m3-lite:MeterPerSecond       a       owl:Class .

soundCity:unit.1001.SpeedSensor     a       m3-lite:MeterPerSecond .

soundCity:resource.1001.AccelerometerSensor        a            ssn:SensingDevice ;
              iot-lite:hasQuantityKind  soundCity:quantityKind.1001.AccelerometerSensor ;
              iot-lite:hasUnit          soundCity:unit.1001.AccelerometerSensor .

ssn:SensingDevice     a                     owl:Class ;
                      rdfs:subClassOf       ssn:Device .

m3-lite:DecibelA  a  owl:Class .

soundCity:doi.1001.Pollution a       m3-lite:Pollution .

m3-lite:Transportation       a       owl:Class .

soundCity:SoundCityTestbed   a       ssn:Deployment .

geo:Point  a    owl:Class .

soundCity:doi.1001.Transportation    a       m3-lite:Transportation .

soundCity:location.1001      a         geo:Point ;
        geo:alt   "0.0"^^xsd:double ;
        geo:lat   "0.0"^^xsd:double ;
        geo:long  "0.0"^^xsd:double .

geo:lat  a           owl:AnnotationProperty ;
        rdfs:domain  geo:Point .

ssn:onPlatform  a            owl:ObjectProperty ;
                rdfs:range   ssn:Platform .

geo:long  a            owl:AnnotationProperty ;
          rdfs:domain  geo:Point .

soundCity:quantityKind.1001.SoundSensor     a       m3-lite:Sound .

soundCity:resource.1001.SoundSensor  a                         ssn:SensingDevice ;
                      iot-lite:hasQuantityKind  soundCity:quantityKind.1001.SoundSensor ;
                      iot-lite:hasUnit          soundCity:unit.1001.SoundSensor .

soundCity:unit.1001.AccelerometerSensor     a       qudt:DegreeAngle .

geo:alt  a            owl:AnnotationProperty ;
         rdfs:domain  geo:Point .

m3-lite:Pollution  a  owl:Class .
```



```
ssn:hasDeployment     a             owl:ObjectProperty ;
                      rdfs:range    ssn:Deployment .

m3-lite:DirectionHeading     a       owl:Class .

iot-lite:hasUnit  a                         owl:ObjectProperty ;
                  owl:equivalentProperty    base_ontology:hasMetaData .

qudt:DegreeAngle       a       owl:Class .

soundCity:platform.1001      a                   ssn:Platform ;
                             iot-lite:isMobile   true ;
                             ssn:attachedSystem  soundCity:resource.1001 ;
                             geo:location        soundCity:location.1001 .

soundCity:resource.1001.ProximitySensor     a                          ssn:SensingDevice ;
               iot-lite:hasQuantityKind  soundCity:quantityKind.1001.ProximitySensor ;
               iot-lite:hasUnit          soundCity:unit.1001.ProximitySensor .

soundCity:doi.1001.Health    a       m3-lite:Health .

soundCity:quantityKind.1001.ProximitySensor a       m3-lite:Proximity .

m3-lite:Proximity  a  owl:Class .

m3-lite:Sound  a  owl:Class .

soundCity:unit.1001.ProximitySensor  a       m3-lite:Others .
```

Sample Observation Annotations from the SoundCity Testbed

```
@prefix :      <http://ontology.fiesta-iot.eu/ontologyDocs/fiesta-iot#> .
@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .
@prefix qu:    <http://purl.org/NET/ssnx/qu/qu#> .
@prefix ns:    <http://creativecommons.org/ns#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix fiesta-iot: <http://ontology.fiesta-iot.eu/ontologyDocs/fiesta-iot#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix txn:   <http://lod.taxonconcept.org/ontology/txn.owl#> .
@prefix base-ontology: <http://www.onem2m.org/ontology/Base_Ontology/base_ontology-v0_9_0#>
.
@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .
@prefix ssn:   <http://purl.oclc.org/NET/ssnx/ssn#> .
@prefix geo:   <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix soundCity: <http://www.soundcity.mobi/> .
@prefix terms: <http://purl.org/dc/terms/> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix base_ontology: <http://www.onem2m.org/ontology/Base_Ontology/base_ontology#> .
@prefix dul:   <http://www.loa.istc.cnr.it/ontologies/DUL.owl#> .
@prefix time:  <http://www.w3.org/2006/time#> .
@prefix vann:  <http://purl.org/vocab/vann/> .
@prefix dc:    <http://purl.org/dc/elements/1.1/> .

soundCity:resource.6136.SoundSensor
        a               ssn:SensingDevice ;
        ssn:madeObservation  [ a                            ssn:Observation ;
            ssn:observationResult        [ a             ssn:SensorOutput ;
                      ssn:hasValue  [ a                 ssn:ObservationValue ;
                      iot-lite:hasUnit  soundCity:unit.6136.SoundSensor.DecibelA ;
                      dul:hasDataValue  "63.63859"^^xsd:double
                                  ]
                                ] ;
```



```
                    ssn:observationSamplingTime  [ a         time:Instant ;
                    time:inXSDDateTime  "2016-11-01T10:54:08.865Z"^^xsd:dateTime
                                  ] ;
                    ssn:observedBy      soundCity:resource.6136.SoundSensor ;
                    ssn:observedProperty soundCity:quantityKind.6136.SoundSensor.Sound ;
                    m3-lite:hasMeasurementType   m3-lite:Automatic ;
                    geo:location                 [ a         geo:Point ;
                                                 geo:alt   "0.0"^^xsd:double ;
                                                 geo:lat   "48.8278504"^^xsd:double ;
                                                 geo:long  "2.3495289"^^xsd:double
                                                 ]
                     ] .

soundCity:unit.6136.SoundSensor.DecibelA
        a       m3-lite:DecibelA .
soundCity:quantityKind.6136.SoundSensor.Sound
        a       m3-lite:Sound .
```

