Ref. Ares(2017)519621 - 31/01/2017

TagItSmart!
Smart Tags driven service platform for enabling ecosystems of connected objects

Grant agreement 688061

Resource and Service Modelling Specification

Deliverable ID:	**D 4.1**
Deliverable Title:	**Resource and Service Modelling Specification**
Revision #:	**1.0**
Dissemination Level:	**Public**
Responsible beneficiary:	UNIS
Contributing beneficiaries:	DONDE, DNET, EVT, UPC, SIE, FLE
Contractual date of delivery:	31.12.2016
Actual submission date:	31.01.2017

Start Date of the Project: 1 January 2016 Duration: 36 Months

Contents

Section 1 - Introduction 4

Section 2 - Related work and state-of-the-art on Service and Entity modelling and Domain Ontologies 5

2.1 State-of-the-art in Virtual Entity, Sensor and Actuator ontologies 5

2.1.1 Virtual Entity ontologies 5

2.1.2 Sensor and Actuator ontologies 7

2.2 State-of-the-art in IoT Service ontologies 9

2.2.1 OWL-S based ontologies 9

2.2.2 Linked Services 11

2.3 State-of-the-art in User ontologies 12

2.4 State-of-the-art in Domain ontologies 13

2.5 Discussion on Related Work 15

Section 3 - Modelling Concepts 16

3.1 Physical and Virtual Entity 16

3.2 Services 16

3.3 Domain knowledge 17

3.3.1 Digital Products 17

3.3.2 Recycling Domain 18

Section 4 - Requirements for Entity, Service and User models 22

4.1 Virtual Entity Modelling Requirements 22

4.2 Service Modelling Requirements 22

4.3 User Modelling Requirements 23

Section 5 - Semantic Model overview 24

5.1 Virtual Entity Model 24

5.2 Service model 27

5.3 User model 28

Section 6 - Conclusions and future work 30

Section 7 - References 31

Document history

Revision	Date	Modification	Authors
0.1	19.10.2016	Draft created	Suparna De, Stelios Georgoulas (UniS)
0.2	26.10.2016	Inputs to sections 1, 2 and 5	Suparna De
0.3	18-28.11.2016	Input from different partners	Nenad Gligoric (DNET), Pekka Liedes (UPC), Bogdan Tarnauca (SIE), Suparna De (UNIS), Iker Larizgoitia Abad (EVT), Iván González (DONDE)
0.4	09.12.2016	First integrated version	Suparna De
0.5	14.12.2016	Inputs to section 2.2 (EVT) and from FLE	Iker (EVT), Yiwei Fang, Dave Snelling(FLE)
0.6	10.01.2017	Updates to section 3,4,5	Suparna De
0.7	25.01.2017	Updates to sections 3.3.2	Iván González
0.8	27.01.2016	Version available for internal review	Suparna De
0.9	27-30.1.2017	Internal review	Stylianos Georgoulas, Yiwei Fang
1.0	30.1.2017	Final version ready for submission	Suparna De

Section 1 - Introduction

The TagItSmart project aims to create an open, interoperable cloud-based platform for managing the lifecycle of everyday mass-market objects that are made part of the Internet of Things (IoT) through SmartTags. The SmartTags combine the capabilities of functional inks with the pervasiveness of digital QR-codes and electronic printed circuits (e.g. NFC tags). The resultant smart objects then act as intelligent markers for a variety of environmental conditions by changing their status in response to these and can also be seamlessly tracked throughout their lifecycle, through embedded tag readers in smartphones and NFC readers. The data extracted from the SmartTags observations, in conjunction with contextual information, enables a new ecosystem of SmartTag-enabled services and intelligent applications for new markets.

The TagItSmart platform supports the tag-enabled process, from tag creation to the acquisition of the data from the SmartTag enabled objects, through a service composition platform which enables developers to efficiently create and quickly deploy their SmartTag enabled objects and corresponding services. As part of this objective, Work Package 4 (WP4) aims to provide the toolset and supporting mechanisms, realised as a service composition platform, for users and developers to create and instantiate SmartTag enabled services and retrieve existing ones to compose more complex services based on these.

To support and initiate these tasks, this first deliverable of WP4 sets out the modelling aspects of the proposed platform. This includes definition of structured, formal models for the relevant components, which include the smart object, i.e. the physical entity and its SmartTag enabled digital version, i.e. virtual entity (VE), users as well as the services themselves and domain knowledge for assisting the service composition process. The models defined in this deliverable describe properties that link to the SmartTag model defined in WP2, in order to interact with them, thus together forming the core set of data models of the TagItSmart project. The defined models will ease the data integration between the different architectural components defined in WP1.

It is envisaged that the models are implemented using semantic Web technologies (e.g. Resource Description Framework (RDF), Web Ontology Language (OWL)) in order to cope with the challenging problems of heterogeneity and interoperability caused by the large number of objects with different characteristics. Semantic modelling for the IoT domain provides a basis for interoperating among different systems and applications by providing machine understandable and processable annotations.

In this deliverable, we present an ontology for virtual entities, users and services by integrating and extending existing work in modelling concepts on the IoT. The ontology helps exploit the synergy of the existing efforts and provides support for crucial tasks in the TagItSmart platform such as IoT service discovery, composition, user role and access definition etc. The ontology is compatible with several widely used semantic models in IoT and is designed to be lightweight to promote reuse and support efficient inference and reasoning for the use-cases defined in the project.

Section 2 - Related work and state-of-the-art on Service and Entity modelling and Domain Ontologies

This section provides an overview of relevant state of the art for semantic data models and data standards for virtual entity, service and user modelling as well as those in the domains relevant to the TagItSmart project use-cases such as location modelling, upper ontologies and quantities.

2.1 State-of-the-art in Virtual Entity, Sensor and Actuator ontologies

2.1.1 Virtual Entity ontologies

One of the first European Union (EU) research initiatives in the IoT area, the IoT-A project[1], defined the IoT domain model [1], identifying the main concepts of the IoT and the inter-relationships between them. A semantic model [2] for entities, resources and services was also proposed, with resources forming the software representation of device functionalities. The focus of interactions by human users or software agents was identified to be the 'entity'. An entity is modelled to have attributes that tie it to the domain (i.e. observable or actionable features), location attributes as well as type and identifier specifications. Also captured are optional temporal features and links to known vocabularies for specifying ownership. A similar device and entity model is proposed in [3] and in [4] with resources identified as the computational element of a device and categorised as on-device or network resource to represent events in an IoT environment.

In contrast to the entity modelling constructs reviewed above, Zhu *et al.* [5] define a Physical Thing-Service Oriented Architecture (PT-SOA) ontology for physical things (PTs) and services in the cyber-physical domain, where they model the PTs as providers (and recipients) of services. The PT ontology has 4 classes: *physical profile* to depict the things' physical properties, any constituent PT and components; *operation profile* to specify the resources required for its operation, control mechanisms, maintenance and physical constraints when providing services; *context* to depict the dynamic state of the PT and the *scheduled service* specification for scheduling the PT to serve multiple requests to its provided services.

The EU ICT-FP7 project iCore[2] defined the concept of a Virtual Object (VO) as the virtual representation of a real world object, such as sensors, devices, or everyday objects, in order to abstract the technical heterogeneity of the possible objects in the IoT. The VO model was defined [6] to support the cognitive management of virtual objects. Each VO concept in the model represents an Information and Communication Technologies (ICT) object, which in turn is associated to a non-ICT object. The model also specifies the VO function in terms of locations and functions of the ICT object. The functions of ICT objects are further expressed as *Input*, *Output*, *Cost*, and *Utility*. *Output* is also bound with *OutputMetadata* to describe its data.

A similar VO concept is also outlined by Christophe *et al.* [7] who present a semantic model for representing virtual objects, to define the structure and capabilities of objects. The model reuses Friend-of-a-Friend (FOAF)[3] vocabulary to indicate the owner of objects, and links to the GeoNames ontology[4] to show the location of the corresponding smart space.

[1] Internet of Things Architecture, http://www.iot-a.eu/public

[2] Internet Connected Objects for Reconfigurable Ecosystems (iCore), http://www.iot-icore.eu

3 http://xmlns.com/foaf/spec/

[4] http://www.geonames.org/ontology/documentation.html

A simplified version of the iCore VO ontology was adopted in [8], focussing on the data provisioning capabilities of a VO. The VO concept was used to represent different sensing objects, including those with no fixed locations (e.g. sensors mounted on public transport). The VO metadata proposed in this work includes the VO ID, name, type, a Boolean property indicating if the underlying real world object is mobile and the location, expressed in terms of a WGS84 [9] modelled latitude, longitude and a geohash[5] value.

The COSMOS project[6] provided also a semantic model derived from the IoT-A reference architecture covering the entire chain of description from the topmost abstractions of the Virtual Entities (VEs) and Group Virtual Entities (GVEs) down to the Physical Entities (PEs) and the IoT services and their endpoints parameters. In COSMOS, the PE represents any entity from the physical world (an office room, a vehicle, a person) whose properties can be measured and upon which one can act. The VE is not only the abstraction of the PE (which reads or acts on resources such as sensors and actuators) but also a wrapper for additional functions augmenting the sensing, controlling and analysis of the PE. Therefore, a VE exposes IoT Services (for accessing the resources such as sensors and actuators related to a PE) and also VE Services providing functions such as context awareness, social relations modelling and autonomous behaviour.

The Web Thing Model[7] is a member submission to the W3C that originated from the EU ICT-FP7 project COMPOSE[8] and the work carried out in the Web of Things W3C Interest Group. This model defines how to describe the virtual counterpart of any physical object to enable them for the Web of Things. It defines as well a Web API for Things and guidelines for the creators of applications for the Web of Things. It mainly defines how a physical thing can expose properties and actions to the Web and how the interactions between them are realised by following Web standards.

URL	Description
{wt}	The root resource URL
{wt}/model/	The model of a Web Thing
{wt}/properties/	The list of properties
{wt}/properties/{id}	A specific property
{wt}/actions/	The list of actions
{wt}/actions/{id}	A specific action
{wt}/actions/{id}/{actionId}	A specific execution of an action
{wt}/.../	...

Figure 1: Web Resources exposed by a Thing according to the Web Thing Model

Based on this simple model and different possible architectures, any object can define a digital representation accessible via a RESTful Web API. Examples of Web Things are: an Arduino board, a garage door, a bottle of soda, a building, a TV, etc. This model provides a common and basic vocabulary and Web friendly interface for things to expose their capabilities and

[5] Geohash, http://geohash.org/

[6] Cultivate resilient smart Objects for Sustainable city applicatiOnS (COSMOS), http://iot-cosmos.eu/

[7] Web Thing Model, W3C Member Submission, https://www.w3.org/Submission/2015/SUBM-wot-model-20150824/

[8] Collaborative Open Market to Place Objects at your Service http://www.compose-project.eu/

data. The model can be easily extended with additional semantics by combining the JavaScript Object Notation (JSON) models with JSON-Linked Data (JSON-LD) context references to external ontologies.

2.1.2 Sensor and Actuator ontologies

Early modelling efforts for sensor and actuator networks were driven by the Sensor Web Enablement (SWE) initiative [10] of the Open Geospatial Consortium (OGC) which defined a set of XML schemas and standards aimed at discovering Web accessible sensor networks and archived data. The widely accepted sensor ontology, which builds upon the SWE initiatives, is the SSN ontology [11] from the W3C's Incubator Group on Semantic Sensor Networks[9] to describe sensors and sensor networks. The SSN ontology describes sensors from a number of perspectives [12], including those representing: a) *Sensor*: definition of a sensor, to include anything that can estimate or calculate the value of a phenomenon, the sensing procedure and what is being sensed, b) *System*: to model systems of sensors and their deployment, c) *Feature*: the property being sensed and, d) *Observation*: observation values and their metadata. The different modules of the SSN ontology are shown in Figure 2. The SSN ontology, however, does not include modelling aspects for features of interest, units of measurement and domain knowledge that are related to sensor data and need to be associated with the observed data to support autonomous data communications and efficient reasoning and decision making processes. As the domain knowledge and unit of measurements are not specified in the SSN ontology, these are intended to be included from other ontologies via OWL imports. Extensions to other components in the IoT domain are not specified in the ontology.

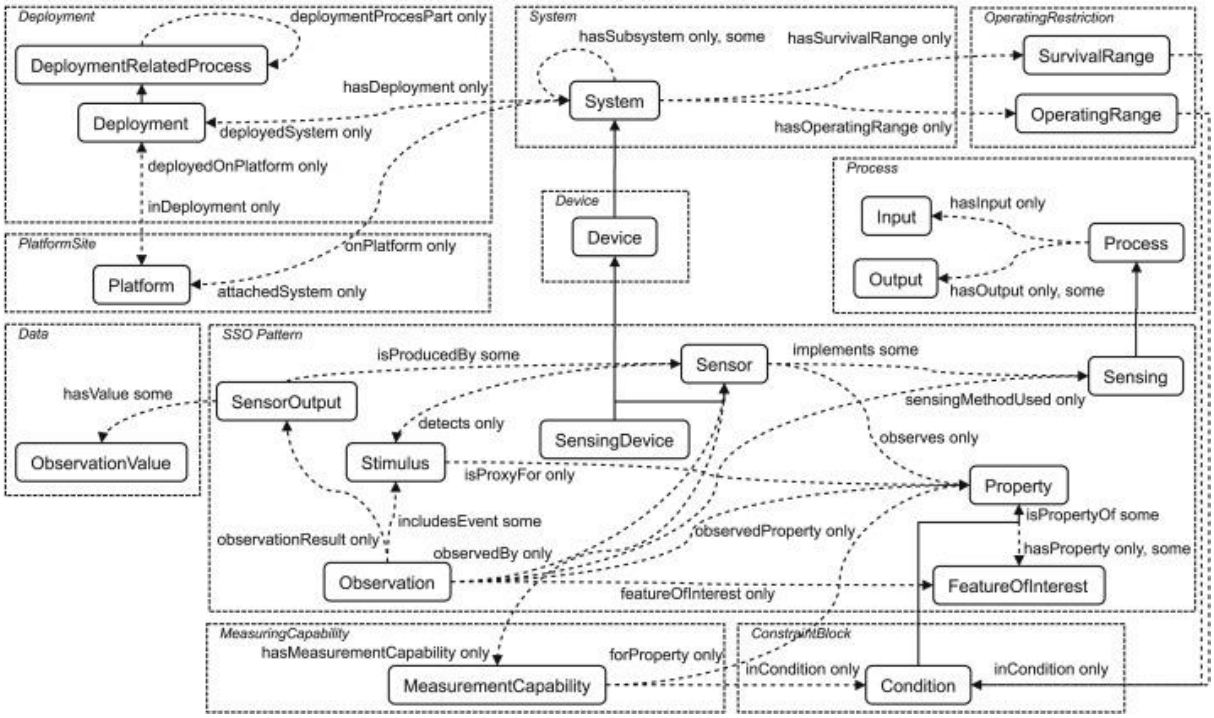

Figure 2: The SSN ontology and conceptual modules; W3C SSN-Incubator Group (SSN-XG) report [13]

[9] http://www.w3.org/2005/Incubator/ssn/

The CSIRO sensor ontology [14] was the precursor of the W3C SSN sensor ontology. It provides a semantic description of sensors in terms of the sensor grounding (platform, dimensions, calibration, power-source and access mechanism) and operation specification (operation, process and results). Concepts for sensor measurements are not part of the ontology. Moreover, similar to the SSN ontology, concepts for domain knowledge, units of measurement, location etc. are not included. Thus, more modelling concepts are needed to link the sensor descriptions to sensor measurements and then to the real-world objects in the IoT domain. Previous sensor description models include OntoSensor [15] which constructs an ontology-based descriptive specification model for sensors by excerpting parts of SensorML [16] descriptions and extending the IEEE Suggested Upper Merged Ontology (SUMO)[10]. However, it does not provide a descriptive model for observation and measurement data.

IoT-Lite[11] is another ontology derived from SSN, made as a result of EU FP7 FIWARE[12] project and the EU H2020 FIESTA-IoT[13] project. The main goal was to provide lightweight ontology for devices with constrained resources by lowering the complexity of the ontology tree. The IoT-Light ontology can be used to represent resources, entities and services, but it is as well a meta ontology that could be extended and combined with data streams representation ontologies such as Stream Annotation Ontology (SAO) [17].

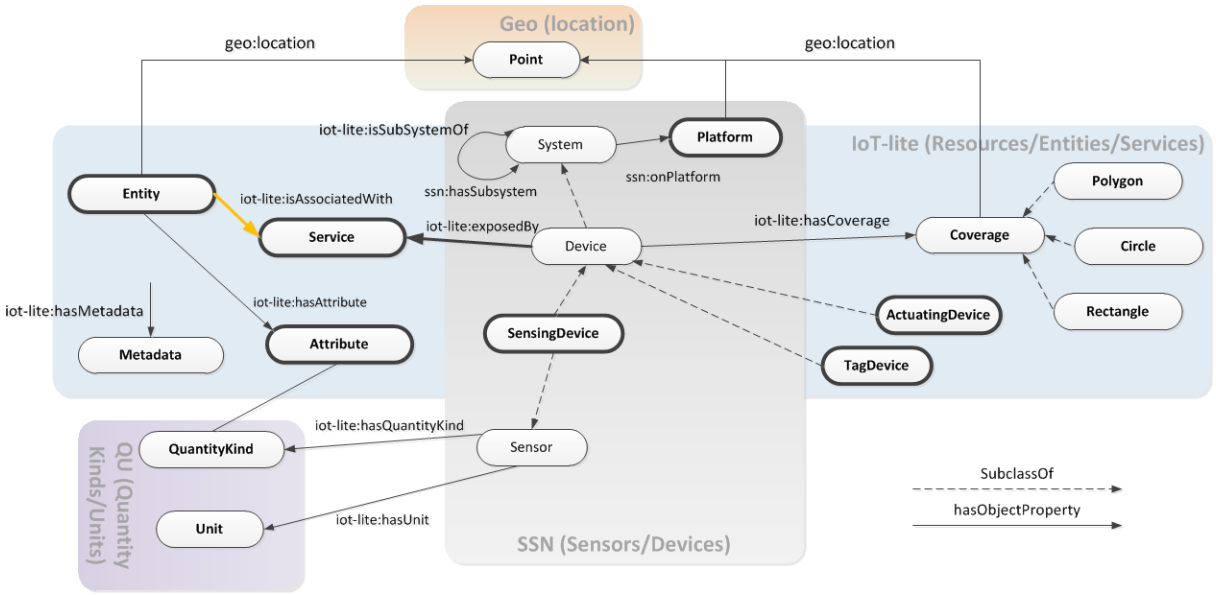

Figure 3 IoT-Lite ontology[10]

In the figure above the concept of the ontology is presented with main relationships, which describes IoT concepts into three classes: Objects, system or resources and services. The devices are also split into, although not restricted to, three classes: sensing devices actuating devices and tag devices. The services are described with an availability or access control and a coverage. This coverage represents the area covered by the IoT device.

[10] http://www.ontologyportal.org/

[11] http://iot.ee.surrey.ac.uk/fiware/ontologies/iot-lite

[12] https://www.fiware.org/

[13] Federated Interoperable Semantic IoT Testbeds and Applications, fiesta-iot.eu/

In contrast to generic things or device ontologies which may be applied to sensor or actuator instantiations, a number of recent research initiatives have specifically looked at modelling actuators and their functionalities by extending the SSN ontology. The Semantic Actuator Network (SAN) ontology [18] proposes the *Actuator-Stimulus-Operation* modelling pattern as analogous to the *Sensor-Stimulus-Operation* pattern of the SSN ontology. According to this model, an actuator is an object that modifies an environment property by producing a stimulus. The *Operation* concept describes the dynamic description of an actuator by defining the actuator operation that modifies the property of a specific *feature of interest*. The *feature* and *property* perspective are the same as in the SSN ontology, with the only change being that features and properties are `changed' by actuators instead of being 'sensed'.
A simplified actuator ontology for home care systems has been proposed by Wang et al. [19] where an actuator is modelled as a refinement of a *Device* and provides a *Service*. A service in turn supports a number of *Operations* which have zero or more *Parameters*. The *Actuator* class is further sub-classed into concepts suited to the home domain, such as *Alarm*, *DVDPlayer*, *Light*, *MobilePhone*, and *TV*.

2.2 State-of-the-art in IoT Service ontologies

To facilitate the development of large-scale, loosely coupled IoT applications, researchers have been applying service-oriented principles to decouple the sensing/actuating functionalities of the IoT things and their hosts. The main tenet has been to abstract the things' functionalities and capabilities in terms of standard service interfaces to support uniform service operations.

2.2.1 OWL-S based ontologies

The semantic annotation methodology of Web Services has been adopted in the IoT domain by researchers to expose services offered by the smart objects. The modelled services provide functionalities to provide information about entities they are associated with or to manipulate the physical properties of the related entities or their surrounding environment. The OWL-S model for SOAP/WSDL services is based on the 'Profile-Process-Grounding' pattern and much of the complexity stems from the process modelling. Semantic modelling efforts in the WoT area apply a modified version of the OWL-S modelling constructs, which is referred to as the 'profile-model-grounding' design pattern, where the profile and grounding are adapted from OWL-S and refined to fit RESTful services and the model excludes process modelling and is based on the atomic service modelling in OWL-S and RESTful service modelling in hREST. The service profile represents the non-functional aspects of the service; what the service is. The service model defines the functional aspects of the service to describe its behaviour and the grounding describes the access details of the service.

The IoT-A project applied the OWL-S principles to service modelling as part of their entity-resource-service ontology suite. The service model [2] represents the functionalities provided by the resource in terms of the IOPE (input, output, precondition, effect) terms. The input and output types are detailed by linking to defined concepts in the Quantity Kinds and Units (QU) ontologies[14]. The service model also specifies the service area (i.e. the observed area for 'sensing services' and the operation area for 'actuating services') and the service schedule (to specify time constraints on service availability).

[14] Library for Quantity Kinds and Units: schema, based on QUDV model OMG SysML, Version 1.2, https://www.w3.org/2005/Incubator/ssn/ssnx/qu/qu-rec20.html

The IoT.est project[15] focussed on the sensor-as-a-service paradigm [20] and proposed a semantic IoT service description model for knowledge representation [21]. They adopt the design pattern of [22] for modelling in the IoT domain, and provide a service model which can be accessed through SOAP/WSDL and RESTful services. The service model applies the OWL-S [23] principles to define a service profile which includes the service name, category, QoS and location; a service model which defines the operations allowed on the service; and the grounding which describes the interaction elements, (e.g. endpoint addresses and communication protocols), and captures the RESTful resources (e.g. Input/Output parameters and the corresponding access URLs). Quality of Service (QoS) and Quality of Information (QoI), are also modelled, including relationships that link to the 'service' class. The defined ontology also incorporates test features to enable automated test case creation and execution. The designed test model enables model-based testing of IoT service capabilities and automated test data creation. The various modules of the service ontology are shown in Figure 4.

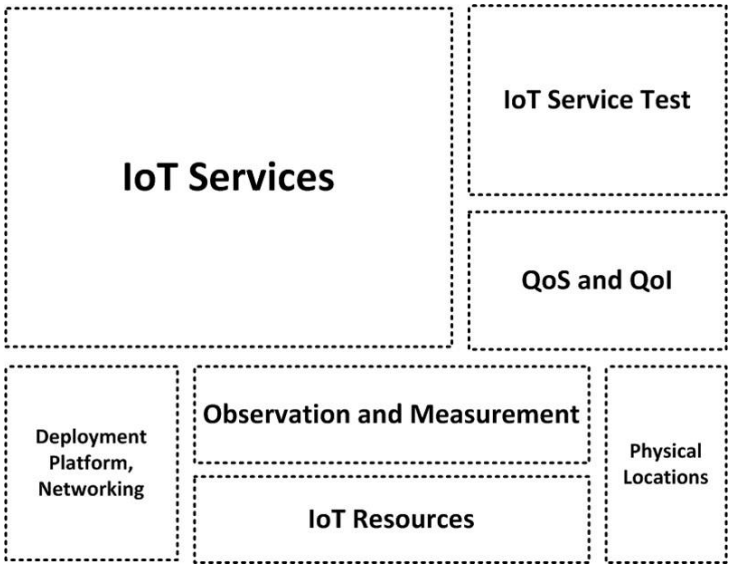

Figure 4: IoT.est service description ontology modules; Wang et al. [21]

The OWL-S specification also forms the basis of the dynamic service ontology proposed in [24] to model transactions in the IoT domain. The service profile element of OWL-S is expanded with a *ServiceStatus* class to describe the current status of entity services. The *ServiceStatus* class is defined in terms of a *name* (as string), *location* (as latitude, longitude or other physical locations), *current status* (which can take 3 values: atomic, composite or simple status), *complicating count* (representing the number of concurrent services) and *waiting sequence* (representing the transactions waiting on the entity). The *ServiceStatus* class instance values support the selection, combination and control of entity services.

The semantic information model in the *ForwarDS-IoT* discovery service [25] also applies the profile-model-grounding pattern of OWL-S to describe services. The Service class is linked to both the Sensor and Actuator classes in the information model through the *exposes* property.

In contrast to the above OWL-S based approaches, the PT-SOA ontology model [5] applies the OWL-S constructs of process-profile-grounding as is. The *Service* class is linked to a PT instance through the *provided by* and *applied to* properties, as a service is provided by a

[15] Internet of Things Environment for Service Creation and Testing (IoT.est), http://ict-iotest.eu/iotest/

physical thing and may also be applied to, i.e. influence, another thing. The conventional OWL-S ontology is extended with the *context precondition* and *context effect* classes to specify the dynamic requirements and subsequent effects of each service of a PT.

The work presented in [26] proposes an ontology-based model for service oriented sensor data and networks. The ontology consists of three main classes: *ServiceProperty*, *LocationProperty*, and *PhysicalProperty*. *ServiceProperty* explains the functionality of a service, while properties in the other two components describe contextual and physical characteristics of the sensor nodes in a wireless sensor network architecture. The system, however, does not specify how sensor data will be described and interpreted in a sensor network application.

The service ontology defined as part of the IoT-based service integration ontology (IIO) [27] represents services in a smart city. It is represented in terms of a *Topic*, defined by developers and used to distinguish between service offerings in the city. The service class is further defined in terms of the *Method* concept (Create, Read, Update, Delete methods) to reference IoT resources, URLs and repositories to reference the service platform, and links to the *User* class.

At the bottom of the COSMOS semantic model lie the actual endpoints of all the exposed services (IoT Services and VE Services). COSMOS supports REST endpoints as well as publish-subscribe broker based endpoints (e.g. ActiveMQ, RabbitMQ). These endpoints are also enriched semantically. Endpoint parameters descriptions include not only the interface types (REST service based or publish-subscribe broker based) and their datatypes but also the semantic meaning of each parameter. This extends the search capabilities and matching of interfaces whenever needed (e.g. for plain resource searching or for more complex scenarios such as service orchestration) thus facilitating interoperability.

2.2.2 Linked Services

The SOA4All project[16] developed Linked Services [28], a framework for dealing with services on the Web. It extends notions around Linked Data, whose main objective is exposing and interlinking data previously enclosed within silos. The resulting Web of Data is based upon four simple principles, known as the Linked Data principles [29] which essentially dictate that every piece of data should be given an HTTP URI which, when looked up, should offer useful information using Web standards such as RDF for representing data and SPARQL to support online data queries. Importantly, data should be linked to other relevant resources thereby allowing humans and computers to discover additional information.

Linked Services build upon Linked Data and govern the way data sources and services are described, discovered, invoked, and integrated. In a nutshell, Linked Services are services that can consume and produce Linked Data and whose descriptions (such as their functionality and input/output data types) are also published as Linked Data. Linked Services simplify the integration of heterogeneous services by relying on a common means for representing data. Linked Services can easily be integrated with existing Linked Data sources as both data and service interfaces are semantically described according to shared vocabularies. The data from the Web of Data can be directly used to invoke services. Combining Linked Services with Linked Data also enhances service discovery due to the provision of semantic descriptions that include links to/from other datasets and that are exposed using standards for data access

[16] http://projects.kmi.open.ac.uk/soa4all/

and querying (esp. HTTP, RDF and SPARQL). For example, based on the types of data in an application's workspace, it is possible to exploit the semantic description of service inputs in order to obtain only services that can process the available data.

The Linked Services approach uses the Minimal Service Model (MSM) [30] as its core conceptual model. The Minimal Service Model, driven by Semantic Web best practices, builds upon existing vocabularies, namely SAWSDL[17], WSMO-Lite [31] and hRESTS [32].

In a nutshell, MSM is a simple RDFS integration ontology based on the principle of minimal ontological commitment; it captures the maximum common denominator between existing conceptual models for services, covering the core semantics of both Web services and Web APIs in a common model, homogeneously supporting publication, discovery and invocation.

MSM, shown in white and denoted by the `msm` namespace in Figure 5, defines `Services` that have a number of `Operations`, which in turn have `input`, `output` and `fault` (error) `MessageContent` descriptions. `MessageContent` may be composed of mandatory or optional `MessageParts`. The intent of the message part mechanism is to support finer-grained input/output discovery, as available in OWL-S and WSMO, especially including support for optional parts.

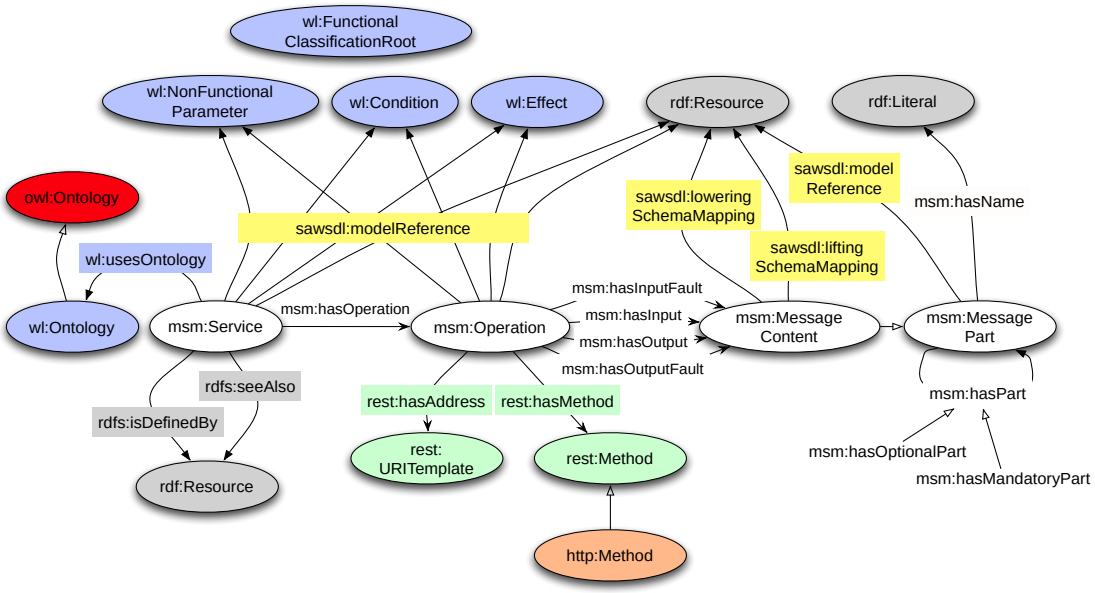

Figure 5: The Minimal Service Model [30]

MSM descriptions can be annotated with human-readable labels, comments and documentation, using existing RDF properties `rdfs:label`, `rdfs:comment`, and `dc:description`, which can support full-text search capabilities that will complement the matchmaking algorithms that use the formal semantics of the service descriptions.

2.3 State-of-the-art in User ontologies

User profile modelling has mainly been performed in related works to enable personalised services by adapting a service to the requirements and needs of a particular user [33]. The early focus of such efforts has been for user interface design [34] and Web information retrieval

[17] http://www.w3.org/TR/sawsdl/

[35], with more recent efforts leaning towards capture of user profiles and personalization relating to health factors [33, 36, 37].
The Assisted Living ontology [36] defines a number of modules that include a Profile encompassing the person's habits, impairment and preferences; Health encompassing disease, its symptoms and treatment aspects and an Activity recognition module. The user profile ontology in the MobileSage project [33] is aimed at user modelling and personalisation reasoning to provide assistance services to people with dementia. The profile aspects are modelled through five classes: *CapabilityProfile*, *InterestProfile*, *PreferenceProfile*, *EducationProfile* and *HealthProfile*. The ambient environment is modelled through Context, Location and Activity classes. The user Profile ontology in [37] facilitates a personalisation reasoning mechanism by modelling user preferences in the ontology in terms of preferred media and text size, personal information such as name, age, date of birth, gender, language and health status, physical and cognitive health conditions and activity. The H2020 EU-Japan iKaaS[18] project proposes a user model [38] as part of a suite of data models. The user model encompasses a user's static characteristics (such as name, date of birth, etc.), profile, preferences and dynamic roles in order to tailor service requests in multi-cloud environments accordingly. The iKaaS model also includes the concept of a user relationship (to model trust between two user instances) and attaches access rights depending upon the role specification.

The TagItSmart user model takes as a starting point the user models from the iKaaS project and that by Skillen *et al.* [37]. To keep the resulting model lightweight, common concepts between these two approaches are taken into the model, with more properties added as identified from the TagItSmart Use case (UC) analysis.

2.4 State-of-the-art in Domain ontologies

TagItSmart intends to provide a generic platform and set of services for advanced tagging and labelling whose applicability is not limited to a set of goods or domains. While the core TagItSmart ontology will be able to describe the entities, components and relations which compose the TagItSmart platform, it cannot model all the applications and fields where it can be used. As a result, the TagItSmart semantic model will rely on the TagItSmart core ontology and on domain specific ontologies.

Domain specific ontologies, are created with the goal of providing users the means to model, as the name indicates, domain specific concepts and relations. They are created with a specific modelling target which can range from medicine, music, social relations to online shopping, geography, project planning, to name just a few[19,20].

By using domain specific ontologies, one can reuse elaborate models which otherwise would have required extensive expertise and development effort. Another advantage of reusing existing domain specific ontologies is related to interoperability. Since the semantic field is based on the linked data paradigm, the use of common semantic model facilitates such linkage.

Quantities and Units: The OMG SysML Quantities, Units, Dimensions and Values (QUDV) working group, in cooperation with the Semantic Sensor Network group, produced a set of

[18] Intelligent Knowledge-as-a-Service, www.ikaas.com

[19] http://wiki.dbpedia.org/

[20] http://www.geonames.org/

ontologies for common quantities and units (QU)[21]. On top of these ontologies, the SSN group added a Climate and Forecast (CF)[22] ontology that lists a number of meteorology-related quantities and units. These ontologies are used whenever quantities and units are required in VE or service descriptions.
These ontologies define the namespaces `dim`, `qu`, `quantity`, `unit`, and `cf`.

Upper Ontology - DOLCE Ultra-Lite: The DOLCE Ultra-Lite (DUL)[23] foundational or top-level ontology provides a semantic description for general terms, such as object, event, artifact, etc. that are common across a number of domains. It thus provides a common starting point for formulation of definitions in more domain-specific ontologies where these definitions can be used or extended and thus supports semantic interoperability due to the reuse.

The SSN Ontology is aligned with the DUL ontology. According to the SSN report [13]: "the alignment between the SSN ontology and the DOLCE Ultra Lite upper ontology has helped to normalise the structure of the ontology to assist its use in conjunction with ontologies or linked data resources developed elsewhere". The DUL terms have been used in various places in this deliverable, as detailed in section 5.

Geolocation: Semantic definitions for geolocation terms have been provided through a number of standardisation efforts as well as in research works.

The W3C Semantic Web Interest Group has produced a basic geolocation vocabulary[24] using the WGS84 datum shared with the GPS system. The WGS84 RDF model [9] is a simple vocabulary for representing latitude, longitude and altitude information of a geographical point.

For representing city-scale location features, the GeoNames geographical database is an important source as it provides user-friendly location names or geographical coordinates of geographical features and captures the associated contextual information on region containment and distance among locations. Following the linked data principles, ontology instances in other domain independent models (e.g. device, service etc.) can refer to GeoNames place names through the unique URI to enable semantic reasoning related to relative positioning.

The DBpedia RDF dataset also hosts unique semantic identifiers and descriptions for cities and region features around the world as instances of the *Place* concept in the DBpedia ontology, containing properties defining region containment as a hierarchy of places and other information. The DBpedia instances can also enable location-aware semantic reasoning in WoT applications.

The indoor location ontology in [39] is used to describe a *place* and its location relatively to others, to allow reasoning on adjacency, access etc. The *place* concept is further refined into *building*, *premise* and *floor*. These concepts are formally defined in terms of logical OWL-DL predicates, e.g. a *building* is defined as a 'place not contained in any other place' and containing at least one *floor*. *Premises* are further refined into *Room*, *House*, *Shop* etc. Properties interlinking the various place concepts are also defined, including those defining containment, adjacency (including directions), access and inclusion. A similar indoor location ontology is defined in [40] representing buildings, floors, laboratories and meeting rooms in a university campus.

---

[21] https://www.w3.org/2005/Incubator/ssn/ssnx/qu/qu
[22] http://purl.oclc.org/NET/ssnx/cf/cf-property
[23] http://ontologydesignpatterns.org/wiki/Ontology:DOLCE+DnS_Ultralite
[24] http://www.w3.org/2003/01/geo/#vocabulary

2.5 Discussion on Related Work

A review of the semantic models on entity, sensor, object and services reveals that most research works define semantic models for a combination of these concepts. Figure 6 depicts how these crucial IoT concepts are interrelated.

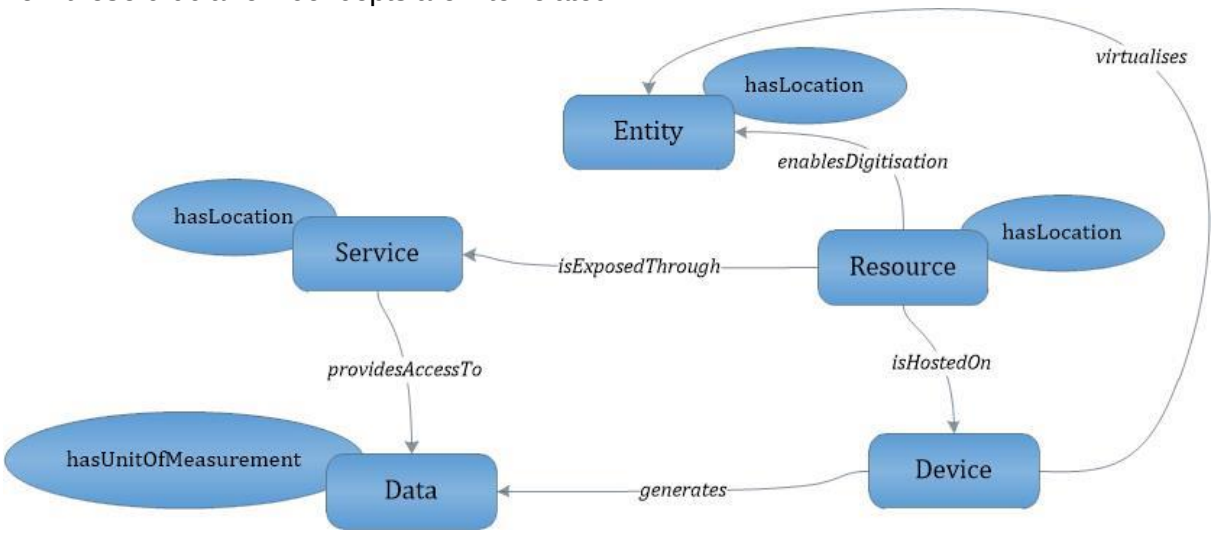

Figure 6. Entity-device-service-data interactions [41]

TagItSmart takes these existing models and concepts as its starting point, with extensions defined to meet the requirements coming from specific UCs. The next section sets out the VE, PE and service concepts and delineates where these diverge from the state-of-the-art definitions. The concept of a user with different characteristics, preferences and roles is as analysed in the state of the art in section 2.3; thus, not defined anew in section 3. For example, the TagItSmart VE model takes the DUL upper ontology and extends the definition of some of its classes as well as defined properties through OWL-DL constructs and restrictions.

Section 3 - Modelling Concepts

The focus in TagItSmart is on the objects tagged with the SmartTags and the resulting information about the objects and the surrounding environment that can be retrieved (and not on the description of the sensing elements of the NFC electronics or the dynamic inks), as well as the innovative services that this concept enables. Thus, the modelling effort in TagItSmart takes the concepts of physical and virtual entities as well as services from the IoT-A and COSMOS projects as a starting point. In the sections below, the definitions of these important concepts are presented, including explanations for cases where TagItSmart extends the definitions of these concepts.

In order to support development of a generic service model, section 3.2 presents some examples of different types of services encountered within the planned TagItSmart UCs. An over-arching list of requirements is then drawn in section 4 based on these, leading up to the definition of the classes and properties of the semantic model in section 5.

3.1 Physical and Virtual Entity

Similar to IoT-A and COSMOS, a physical entity (PE) is anything from the physical world, including a human being, animal, car, store or logistic chain item, electronic appliance or a closed or open environment (e.g. a building).

A virtual entity is the digital counterpart of the PE, which allows it to interact with its surrounding environment and is the core concept in the TagItSmart ontology. The digitisation is made possible by attaching either a SmartTag, sensor or actuator to the PE. Existing digital artefacts which have capabilities relevant to the TagItSmart project are also considered as VEs. Such a VE has certain capabilities, which can be offered as services through the TagItSmart platform. For instance, this would be the case of printers, which can be used in the SmartTag creation service process.

VEs equipped with SmartTags, sensors and/or actuators can expose not only the sensed data (from the SmartTag or sensor) but also additional functionalities which the VE concept introduces, i.e. data related about the underlying PE (e.g. its material composition, location etc.) or those derived from situational awareness of the VE.

3.2 Services

The service concept considered in TagItSmart includes those provided by VEs, which are typically simple services that are associated to one VE function. Backend services are those that are pertinent to the platform, e.g. notification generation and delivery.

An example of a service from the Digital Products use case is the scanning validation service, which requires the FC-scanner to work together with the TagItSmart platform to ensure the scanning process is valid. As part of this service, VE properties are compared to results delivered by the FC-scanner. This applies also to situations where the FC-scanner changes lifecycle (or state) of the VE. For example, if the FC-scanner wants to apply current temperature of VE to 30-degrees but according to the modelled VE properties, it is known that the constituent sensor in this VE can only measure temperatures up to 20-degrees, then the Service can flag that the scanned results must be invalid or tampered.

In the dynamic pricing use case described in the TagItSmart deliverable D1.1[25], there are several relations between the VE and the Backend services. In this use case, the scanning device acts as a gateway by forwarding information to the backend on a scan event. As a part of the VE service, the backend decodes the ID, updates sensor measurements and associates them to the product ID. If a response was requested (when the ID was sent) the backend returns the associated data back to the gateway (scanning device). Returned VEs are visualized on consumer smart phone application presenting data about the products. Another VE service in the backend is related to data processing and it is based on Machine learning to learn about the product lifecycle (when, where a product was used) and about consumers by taking into account input about different context using their VEs. To allow dynamic pricing, during the scanning process the backend extracts the latest price based on the threshold configured for each product and each retailer.

3.3 Domain knowledge

This section brings together the specific domain related knowledge from different UCs, especially in cases where supporting domain knowledge ontologies do not exist in the current state-of-the-art.

3.3.1 Digital Products

When it comes to the use case of Digital Products, it is essential to support different actors within the lifecycle of an item. As each item has a unique lifecycle which involves various actors, services and resources must be able to adapt to this. In principle, when an actor scans a SmartTAG, information delivered to the actor depends on their location and role within the lifecycle, some services are only available to certain users while generic services are available to all users. Same applies to resources, as some users are able to see production data on machinery parameters level but other might only have access to production material resources. In order to provide accurate information to user and to keep user as anonymous as possible, services related to user behaviour are needed, this includes special product catalog for user, where it is possible to register bought items. As this service is personalised for each user, the TagItSmart platform is able to connect requests from one actor (producer) to another actor (consumer) without letting the initiating actor (producer) to know personal information of the receiving actor (consumer).

Manufacturing process and information flow

The production of beer in this use case can be roughly divided into the following processes- brewing, fermentation & conditioning, optional post-treatments (e.g. filtration) and bottling.

Brewing process

In the brewing phase malt and water is mixed together and heated to certain temperatures. The spent grain is removed and the liquid (a.k.a. wort) is boiled together with the hops. From the project perspective, one brewing cycle creates one batch which we later on want to assign to individual products (beer bottle).

Fermentation & conditioning

During fermentation and conditioning beer is stored in tanks. The total amount of time beer is stored depends on the recipe, beer style and brewery conditions. To achieve the best results breweries continuously monitor the conditions in the tank (e.g. temperature). In our case these conditions are made available through the SmartTag. Since batches may be further split into

[25] TagItSmart D1.1 deliverable "Use case descriptions, requirements and market analysis", July 2016.

smaller batches before further post-treatments it is essential for the traceability of the end product to monitor every batch carefully.

Bottling
After fermentation & conditioning, and the optional post-treatment, the finished product is filled into bottles. At this stage batches are no longer split or mixed which allows for the batch in question to be assigned to individual bottles. For the purpose of traceability each bottled is equipped with a unique SmartTag.

Information in delivery chain
From a brewery point of view, delivering bottles to retail stores is a very straight forward process. Bottles are just packed bottles into boxes and delivered into retail store. Those boxes have to be trackable to make sure that all bottles are delivered into the right place. Following this procedure, we track for example, grey imports, because the original target for these bottles is known. In our use case boxes of beer are marked with SmartTags.

Information after consumer has purchased an item
When a consumer purchases one item (a bottle of beer) from a retail store, lifecycle of item is updated accordingly. As in the real world, ownership of the item is changed, it is also now possible for the user to claim ownership of the item. This important change provides valuable information to retail, manufacturer and authorities as it can be used to track how many bottles are in-store, how long the item stayed in shelf, etc... For consumer this change is also important since when a new owner now registers the item to their profile (and allows two-communication) it's possible for manufacturers to send out information to item owners. Also this lifecycle change opens up possibilities for consumer to user item related services via TagItSmart platform, for example local business owners can register via the TagItSmart platform and tell that they have beer related services (parties, recycling, etc.). Then when a user scans a SmartTag they can see services related to beer or other items related to beer.

In recycling
When beer bottles are recycled, it is possible for the recycling machinery to scan a SmartTag and register this information to the TagItSmart platform. In this way, actors involved in the beer lifecycle are able to optimize their processes. For example, retails stores can have appropriate amount of recycling points based on past behaviour of their customers. Authorities can also check how well their local community recycles bottles and manufacturers can see how many bottles end up in recycling.

3.3.2 Recycling Domain

In Europe, recycling is closely linked to EU legislation, but every country can adopt a different approach to its waste management. Within the EU, we can find huge differences among different member countries. The information about what types of waste items are collected and how it is done depends on their national legislation, and it can change depending on the local community inside the country. Having in mind that there are more than 120.000 local communities in the EU, this is a big obstacle to deliver recycling information in Europe at a local level.

To provide precise recycling information in Europe about what can be recyclable and what disposal options are available, there are 2 different levels of information that need to be addressed:

- **Product information**: we need to know in detail the product itself and its packaging, as every country decides how the products can be recycled.
  First, it is very important to know in which components or parts can be the product be divided (jar, lid, label, …), as every component of the product can be recyclled in different ways (some countries separate metal from plastic packaging).
  We need also to know the substances the product contained, as they will determine if the waste is hazardous or not, and that must be collected in a very specific way (for example a battery or a paint can).
  Once we know the parts of the product and if it hazardous, we need to know more about its packaging, in case in has. A way to determine the recyclability of the packaging, is to know the type of packaging (if it is a bottle, a can, a film, …) and the material of the packaging (plastic, metal, …).
  With all this information, we will be able to know that a paint can has 2 components (can and lid), that both contained hazardous products (if they were not water-based) and that are made of metal.
  The problem to get that information is that it depends on different sources, as the producers are the ones who decide what information do they provide to the public and how do they do that. Even if there are international standards to label packaging products, they are not mandatory and they are not applicable in every packaging product.

- **Information about recycling locations**: once we know in detail the product we are going to dispose of; we need to link it to the waste collection system available in the municipality. There are many collection systems in Europe and they are always related to the category of waste collected and the location of the disposal point.
  To provide the available disposal options, we need first to categorize every part or component of the product. Almost every country in Europe has its own categorization (more than 200 in total), but we can determine a simple category for every item in Europe determined by its materials (metal, paper, card, plastic, organic, …) and go a bit further with that categorization in every country (beverage metal can, newspaper, …).
  Second, we must look for what type of collection systems are in the municipality that collect that specific waste category. There are also many collection systems, but in total, there are around 50 systems in Europe (kerbside, recycling banks, recycling centres, reverse vending machines, …).
  With the waste category and the available collection system in the location, we can look for the collection points available around the user. But to do that, we must provide local information about where are those closest collection points where the end-user can get rid of her waste. There are more than 120.000 sources of information, as they are linked to the Councils, but also other type of providers such as national entities that collect waste items. This sources of information are different from country to country, and to find them some national recycling knowledge is needed. Here are some examples of these sources in Spain, but is it very usual to find in every European country the same defragmentation of the information:
  - Pharmacies: http://www.sigre.es/recicla-punto-sigre/buscador/
  - WEEE: http://www.ecotic.es/es/246570/Reciclaje.htm#a3000
  - Batteries: http://www.ecopilas.es/puntos-de-recogida/mapa-de-recogida/
  - Bulbs and fluorescents: http://www.recogidasambilamp.com/index.php
  - Mobile phones: http://tragamovil.es/puntos-de-recogida/
  - Car tyres: http://www.signus.es/es/cadena-de-valor/info/recogida-gratuita-de-neumaticos-1

- Motor oil: http://www.sigaus.es/gestores-de-aceites-usados/gestores-autorizados.aspx
- Plant protection products: http://sigfito.es/puntos-de-recogida/
- Recycling centers: http://www.ocu.org/puntos-limpios/

In the last few years many municipalities and governmental organizations have invested into publishing their internal data to the public under the paradigm of Open Data. Here are some examples of these sources:

- Barcelona Cooking oil street containers: http://www.bcn.cat/tercerlloc/contenidors_oli.xml
- Bologna Organic, Glass and Cans street containers: http://dati.comune.bologna.it/download/file/fid/2293
- Bonn Small appliances street containers: http://opendata.esri-de.opendata.arcgis.com/datasets/6e92ab03fc6546f8889a51322bd51134_0.csv
- Brussels Glass street containers: http://opendata.bruxelles.be/explore/dataset/bruxelles_bulles_a_verre/download/?format=json
- Paris Glass street containers: http://cassini.apur.opendata.arcgis.com/datasets/237baaf5085a4604bcafe74141aca2e6_0.csv
- Toulouse Textile street containers: https://data.toulouse-metropole.fr//explore/dataset/bornes-textiles/download?format=json&timezone=Europe/Berlin&use_labels_for_header=true
- Vienna Plastic street containers: http://data.wien.gv.at/daten/geo?service=WFS&request=GetFeature&version=1.1.0&typeName=ogdwien:ALTSTOFFSAMMLUNGOGD&srsName=EPSG:4326&outputFormat=json

There are many types of data (like spatial and financial data) that those organizations have in their possession, but they publish in many different ways. Here are some examples and references of how data is delivered, using the datasets previously provided:

- geographic coordinate system used: all analyzed datasets came from different zones and sources and were specified for different uses and locations, so it is very logical that they use different geographic coordinate system. They differ in two main aspects: first, the Map projection system, or the numerical system that indicates the location on the Earth surface, mainly Geographical (grades, minutes and seconds or integers) and Universal Transverse Mercator (also called UTM). Second, the Geodetic datum or Geodetic reference system. The ETRS89 (European Terrestrial Reference System 1989) is the recommended frame of reference for geodata for Europe, but for global systems it is more common to use WGS84 (World Geodetic System 84), as it is the default datum used for the Global Positioning System and by Google.
- units used to indicate the available recycling collection points: most analyzed datasets were showing this information in a different way. Some include in the properties of the collection point a field with a number to specify if it exists or not (i.e. to detail the information of every collection point, Vienna includes the fields Fraktion PA, BI, DO, G and KV and the number 1 or 0 to specify if the container is available or not in that specific location), some of them do not

specify units (i.e. Bonn, Toulouse and Paris just include all available recycling collection points, not specifying the number of containers), and some of them specify in properties a field with a number where they indicate the exact number of containers available.

- recycling information included: some datasets explain the recycling information included "outside" the dataset (usually in the metadata: i.e. Glass street containers), while some others include this information in the dataset itself (i.e. Vienna includes this information in the properties of every collection point).
- the language used: it is very common that every entity publishes their datasets in their local language/s, so it makes the creation of a common canonical model more difficult. It can be very challenging to understand what is inside a dataset, mainly the types of containers included (or other collection system used) and the delivery conditions for every collection point provided. Taking just the datasets previously provided we find Spanish, Italian, German and French.
- publishing formats: we can find different formats of how these datasets are published (json, csv, xml, xls, kml, etc.). Brussels and Vienna use json, Bonn and Paris use csv, while Bologna uses a downloadable kml and Barcelona uses xml.

In conclusion, there is a crucial need for all of this information to be homogenized and standardized to a common canonical model, via identifying data sources and integrating them into a common system.

Section 4 - Requirements for Entity, Service and User models

This section describes the requirements identified for the virtual entity, service and user models as they are investigated in the context of the TagItSmart project. The identified requirements include general requirements that guide the design of the relevant semantic models as well as those identified through the UCs. Each identified requirement is also marked as functional or non-functional.

4.1 Virtual Entity Modelling Requirements

ID	Requirement	Type
VE-1	A VE must provide a digital abstraction of an object tagged with a SmartTag	Functional
VE-2	A VE must provide a digital abstraction of an object tagged with a sensor or actuator	Functional
VE-3	A VE must have a unique identification	Non-Functional
VE-4	A VE should contain a list of functionalities provided by the SmartTag/sensor/actuator associated with it.	Functional
VE-5	A VE should contain links to context information, e.g. location	Functional
VE-6	A VE Digital abstraction should include description information that allows its discovery by using particular search criteria.	Non- Functional
VE-7	A VE must provide annotation capabilities for products and their constituent packaging materials	Functional
VE-8	A VE descripiton should contain enough information to enable product recycling decisions	Non- Functional

4.2 Service Modelling Requirements

ID	Requirement	Type
S-1	A Service must provide the abstract description of a service as considered in the TagItSmart context	Functional
S-2	The service model should be independent of any particular service technologies	Non- Functional
S-3	A service must have a unique identification	Non-Functional

S-4	A service must include specification of the offered functionalities	Functional
S-5	A service must allow linking to a VE for cases where the service is a VE function	Functional
S-6	A Service must be complemented by one or more domains	Functional
S-7	A Service must be complemented by a physical or a logical location	Functional
S-8	A service should be associated with at least one user that is characterized as its consumer.	Functional
S-9	A “composite service” must execute particular service logic that describes specific actions that is triggered by particular conditions	Functional
S-10	A Service must be complemented by appropriate information that refers to the domain that supports it and/or it is able to be deployed in.	Functional

4.3 User Modelling Requirements

ID	Requirement	Type
U-1	The user must be associated with particular preferences, policies/ access and/or usage rights with respect to the VEs and the Services	Functional
U-2	The system should enable the representation, storing and retrieval of user preferences, requirements and constraints in user profiles	Functional
U-3	A user instance should have the user’s home location specified in terms of geographical coordinates as well as using the NUTS naming convention	Functional
U-4	The user model should delieanate different roles that the user can assume	Functional

Section 5 - Semantic Model overview

Following the definition and scope delimitation of the virtual entity and service concepts in section 3 and the elicitation of modelling requirements in section 4, this section describes the constituent classes and properties of the virtual entity, user and service models as defined to meet the requirements of the TagItSmart platform architecture and use case scenarios. Since a substantive body of work already exists for these concepts as validated models from various EU IoT projects, the models defined herein are based on the existing state-of-the-art and adapted and extended keeping in mind the specific requirements arising from the project's use cases.

The semantic model has been developed in the OWL-DL (Web Ontology Language – Description Logics) language, which is the second OWL sub-language (the others being OWL-Lite and OWL-Full) and is based on Description Logics, the decidable fragment of first order logic and therefore, enables automated reasoning. It is therefore, possible to automatically compute the classification hierarchy and check for inconsistencies in an ontology written in OWL-DL.

Following the Web of Data's Linked Data principles, the models are not defined in isolation, but include properties that allow linking to each other (where appropriate) and also to external domain ontologies; for instance, the global location URI of an entity could link to the relevant location instance in the GeoNames ontology, where the given location is more fully described.

The following sub-sections follow the notation of symbolising ontology classes and properties in *italics*.

5.1 Virtual Entity Model

Following the ontology design principles of reusing and extending existing ontologies [42], the Virtual Entity ontology makes use of some classes and properties from the following ontologies that are imported into the VE semantic model:

- DUL ontology – the DUL Object class forms the starting node for the VE ontology design. The Object class and some of its sub-classes are extended with classes and properties defined in the VE ontology.
- SSN ontology – the SSN ontology is included into the VE model to allow annotation of sensors that attach to a physical entity to form a VE.
- SAN ontology – for cases, where the ICT object part of a VE is an actuator, the SAN ontology, which itself extends SSN, is included.
- Indoor location ontology – the indoor premises and building ontology, defined in [39], is imported to include indoor location aspects into the VE model
- WGS84 ontology – the WGS84 Point class is used to define the geographical coordinate location specifications of the VEs.

The developed TagItSmart VE model is shown in Figure 7.

The VE class is defined as a sub-class of the *dul:Entity* class. As such, it inherits a number of properties from the *dul:Entity* class that are relevant to the TagItSmart concept of a VE, such as *hasLocation*, *hasConstituent*, *hasPart* and *isDescribedBy*.

In addition, this deliverable defines a number of additional properties and classes to define a VE. A VE has a number of datatype properties that include the specification of its *name*

(specified as a string literal), *type* (URI), unique identifier (*URI* property specified through an URI) and a Boolean property denoting if the underlying physical entity is mobile (*isMobile*). The mechanism for generating the unique identifier for a VE will be defined in deliverable D4.2[26].

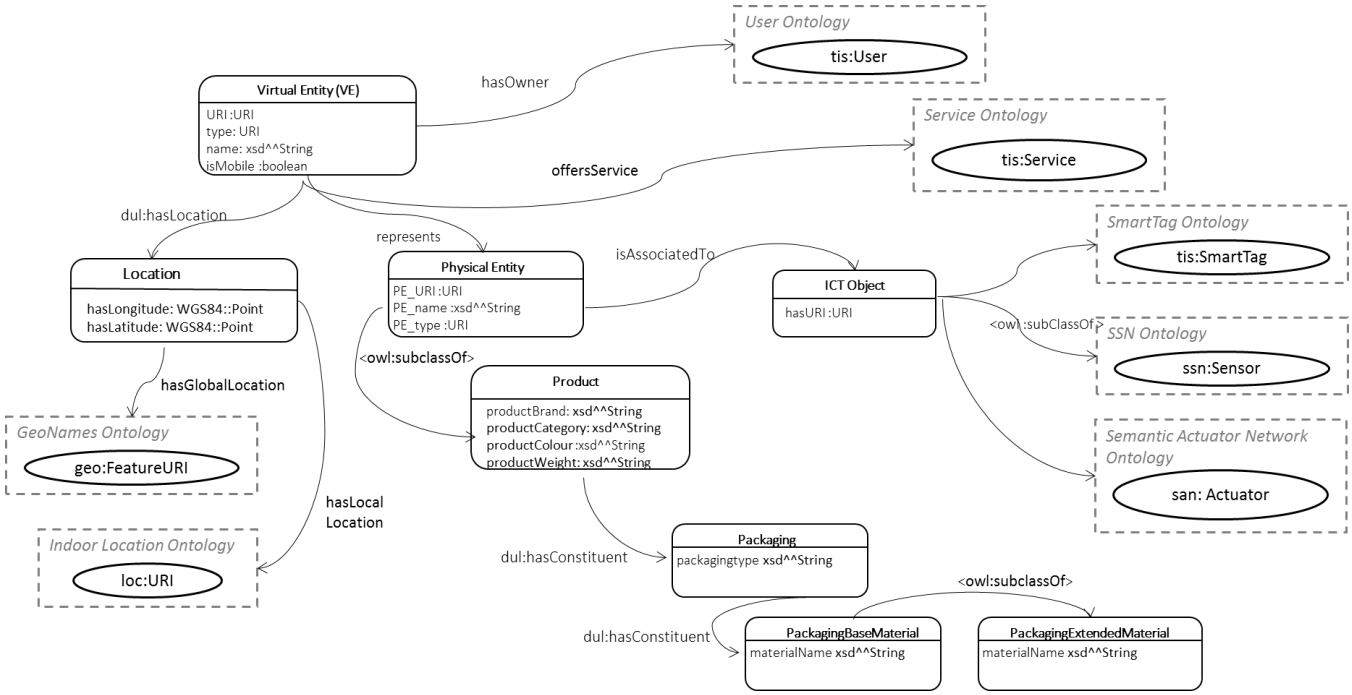

Figure 7: TagItSmart Virtual Entity Model

The VE owner is specified through the *hasOwner* property that links to an instance of the User model.
Since the VE is a digital abstract representation of a physical entity that is associated with an ICT object, this is represented through the *represents* property linking the VE and Physical Entity (PE) classes.
Similar to the VE class, the PE is a sub-class of the *dul:Entity* class and is defined as either a *dul:PhysicalAgent* (e.g. a Person or an Organism) or a *dul:PhysicalArtifact* (e.g. natural objects with a certain function, that can be recycled, or structurally designed objects), expressed through the ∪ (unionOf) operator. This is shown in DL notation in (1) below:

$$Physical_Entity \equiv PhysicalAgent \cup PhysicalArtifact \quad (1)$$

The PE class datatype properties include an identifier (*PE_URI*), name (*PE_name* property) and type (*PE_type,* specified through an URI).

To meet the specific requirements of the TagItSmart lifecycle management UC, a specialised type of PE is defined, i.e. a Product, which is defined as a sub-class of the PE class. Since the lifecycle management UC requires knowledge of the product packaging, a Packaging class is defined, extending the *dul:DesignedArtifact* class. The inherited dul:*hasConstituent* property for the *Product* class, is also refined by placing an existential quantifier restriction (∃) on its range to the *Packaging* class, as shown in (2), thus adding a specialisation to the existential quantifier restriction (∀) to any *dul:Object* class from the DUL ontology.

[26] TagItSmart D4.2 deliverable "Service Composition Platform Description", due in June 2017.

$$Product.dul{:}hasConstituent \exists Packaging \quad (2)$$

The existential quantifier restriction signifies that a Product instance will have some packaging information specified by linking to an instance of the Packaging class, but it may have other *hasConstituent* relationships to other *dul:Object* class instances, thus keeping the DUL ontology constraints in place.
The Product class also has a number of datatype properties, including category, productBrand, productColour and productWeight, all specified as string literals.

The Packaging class is defined by its type (e.g. bottle, film, label etc.) and by constraining the inherited *dul*:*hasConstituent* property to instances of the PackagingBaseMaterial (and its sub-classes) alone, as represented in (3):

$$\begin{aligned} Packaging.dul{:}hasConstituent \exists PackagingBaseMaterial \\ Packaging.dul{:}hasConstituent \forall PackagingBaseMaterial \end{aligned} \quad (3)$$

The combination of the universal and existential quantifiers on the *dul*:*hasConstituent* property ensures that each *Packaging* class instance has at least one *PackagingBaseMaterial* as its constituent by the existential quantifier and also constrains the definition to this class alone by adding a closure axiom by the universal quantifier.

The *PackagingBaseMaterial* class has a sub-class, the *PackagingExtendedMaterial* class, to enable more specialised annotations of the packaging materials. An example could include plastic as the base material and 'grade 5 plastic PP' as the extended packaging material.

The association of the PE to the ICT object is depicted through the *isAssociatedTo* property linking the PE and *ICT_Object* classes. Each constituent ICT object has an URI, with further properties depending upon the type of the ICT device, i.e. whether it is a SmartTag, sensor or actuator. The descriptions of each are then provided by creating and linking to an instance of the relevant model: the SmartTag model (defined in deliverable D2.1), the SSN sensor ontology or the Semantic Actuator Network (SAN) ontology, respectively.

The VE location, is specified through the Location class, to which a VE is linked to using the inherited *dul:hasLocation* property. The Location class is specified through a number of properties which represent locations at different granularities. These properties include latitude (*hasLatitude*) and longitude (*hasLongitude*), which are represented in terms of the WGS84 [9] Point specification. Where available, the VE could also have a global location (*hasGlobalLocation*) specified by linking to an instance in the GeoNames ontology, which could represent certain well-known landmarks or buildings, the city/town or country, as defined in the ontology. Linking to the GeoNames location specification allows semantic reasoning for region containment and other location-specific resolution such as proximity. At a more granular level, the location could also be specified in terms of relative location within an indoor location ontology through the *hasLocalLocation* property.

The functionalities offered by a VE are specified by linking to an instance of the Service model, which is described in the following section.

5.2 Service model

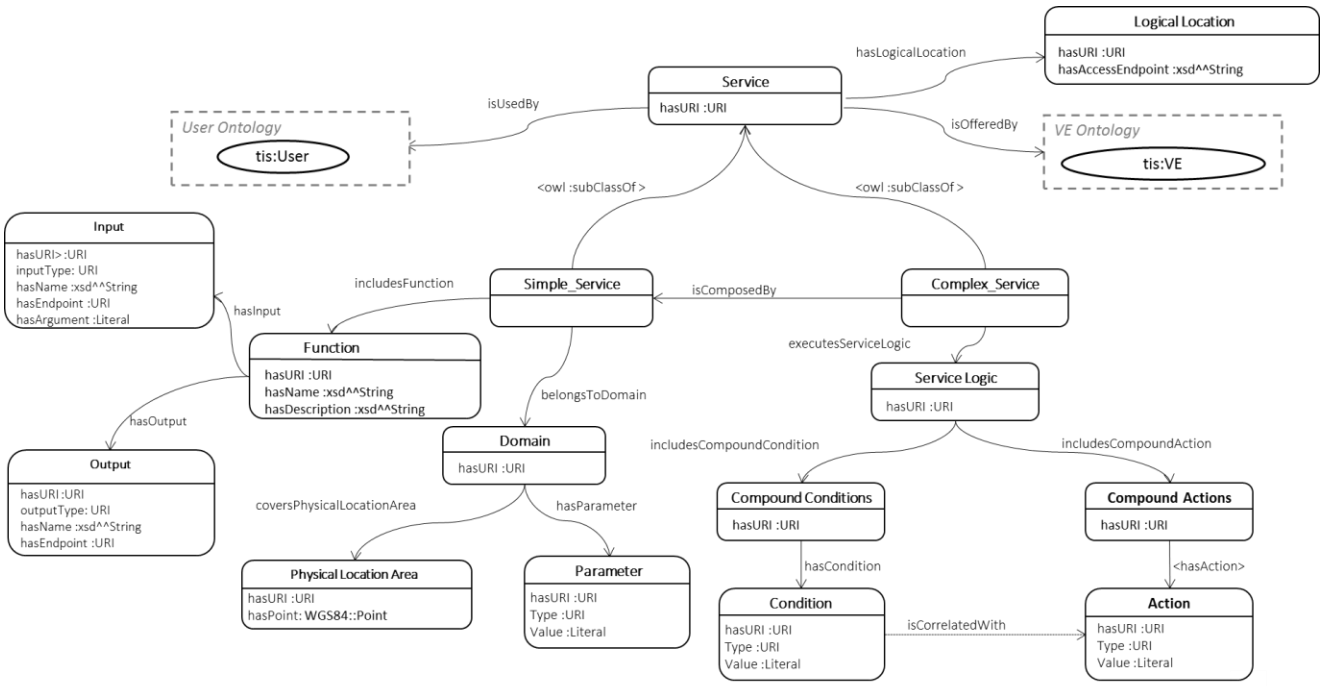

Figure 8. TagItSmart Service Model

The TagItSmart Service Model, shown in Figure 8, constitutes an abstract Semantic Model that can cover the modelling requirements for heterogeneous types of services in the context of the TagItSmart platform. The service model should be expressive enough to represent both VE services (e.g. printing service) and platform, or backend services (e.g. notification service).

The model comprises of one super-class named *Service*, complemented with its first level concepts and properties, as well as by two sub-classes the *Simple_Service* and the *Complex_Service*, defined using the `<owl:subclassOf>` hierarchical representation as shown in (4) in DL notation, that inherit the features of the super-class 'Service' and have some additional specific features.

$$SimpleService \subseteq Service \qquad (4)$$
$$ComplexService \subseteq Service$$

Each service instance has a unique identifier (*hasURI*) specified by a URI and can be part of a VE's functionalities, as specified by linking it through the *isOfferedBy* object property to an instance of the VE model. The user of a service is described through the *isUsedBy* object property associating a service instance to an instance of the User model. The service class also has a property (*hasLogicalLocation*) linking it to the logical location of the service (*LogicalLocation* class), which could be defined as URL of the software implementation of the service, such as some endpoint, complemented by a particular set of properties provided by some API for the service.

For each *Simple_Service*, the service capabilities or offered functionalities are described by the *Function* class, which in addition to an identifier (*hasURI*), textual name (*hasName*) and

short text description (*hasDescription*), also specifies the accepted service inputs and outputs. Both the Input and Output class include an URI, type (e.g. to specify the semantic type), name and endpoint specification (e.g. to specify the endpoint URLs through they are accessible). In addition, the Input class also has an argument property (*hasArgument*) to specify the input arguments that the function accepts.

The domain specification of the service presents an opportunity of associating the service to specific IoT domains such as recycling, smart home etc. This allows the developer to annotate a service as specific to a TagItSmart use case too. The domain specification includes the physical location served by the service, specified in terms of a WGS84 Point instance, which includes latitude, longitude and altitude. The Parameter class allows capturing other domain relevant characteristics in terms of key-value pair attributes.

Since an important aim of WP4 is to develop components to expose composition of SmartTags-based services, the service model enables annotation of such composite services through the *Complex_Service* class. Each complex service is associated with more than one simple service instance that actually compose its features to provide more complex capabilities. Each complex service is defined by a workflow that is described by the *Service_Logic* class, which includes different compound conditions and actions. The compound conditions and actions classes are sets of conditions and actions, respectively, with each condition associated one-to-one with an action. A *Condition* class is uniquely identified by a URI and consists of key-value features that allow description of the properties defining the condition. In practice, the condition type could refer to a Function instance or a threshold. Similar to the condition class, the *Action* class is described by an URI and type-value pairs. The type descriptions can include a Function instance or action filters.

5.3 User model

The user model enables abstract representation of user data. The diagrammatic representation of the model is shown in Figure 9.

The static information about the user is captured in the User_Characteristic class, as key-value property assertions. Captured user characteristics can include first, given or family names, age, gender, preferred language (as a 2 letter code), social network id and even refer to external ontology instances such as FOAF instance.

A user may have one or more different *User_Profiles* that in turn are linked with different *User_Preferences*. Part of a user's profile is the dynamic current user location information and the home location, both of which are specified in terms of a *Location* class instance. The user's current location is relevant for the TagItSmart digital products UC and the lifecycle management UC, while the home location specification is used in the lifecycle management UC. In addition to the geographical coordinates as defined in the VE model, the user model's Location definition also specifies the county and country IDs in terms of the NUTS classification[27].

[27] http://ec.europa.eu/eurostat/web/nuts/overview

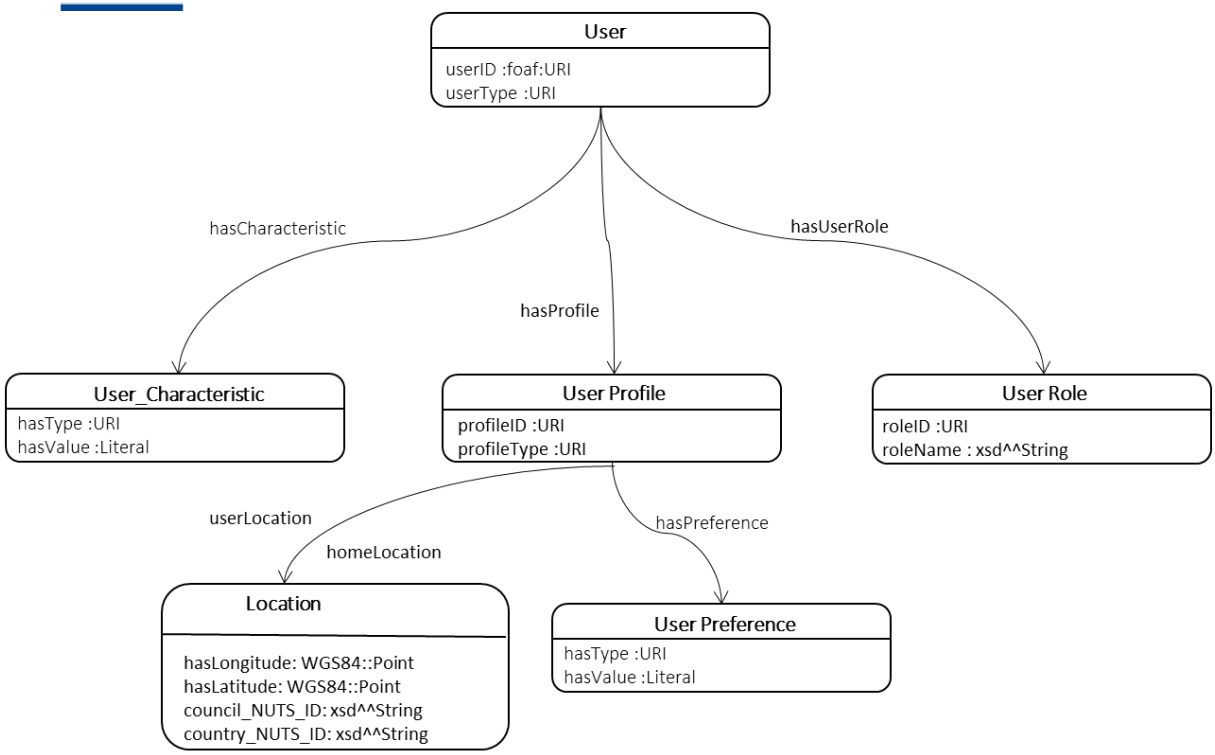

Figure 9. TagItSmart User Model

The *User_Preference* class allows expression of user preferences that relate with user activity in the TagItSmart system. Preferences can guide for instance, customisation of service parameters when the user is using a service. This brings us to the concept of the *User_Role*, since a user can have different roles in the TagItSmart system that are associated with particular access rights. This is applicable for instance in the digital products UC, where the user location and the role specification are used for delivering different levels of information, as well as services to the user, with generic services available to all users and only certain user roles getting access to specialised platform services.

Section 6 - Conclusions and future work

This deliverable has provided a first version of the semantic models for the crucial TagItSmart concepts of a virtual entity, user and service. One of the major advantages of semantic modelling applied to the project's concepts and related entities is that of information enrichment. Entities are not only described using a flat format metadata structure but rather by using the linked data paradigm, allowing references to external concepts and descriptions which can be queried and processed in a uniform way.

The development of the semantic models has been guided by a thorough review of the state-of-the-art in related works in current and past IoT projects as well as standardisation efforts, as presented in section 2.

As outlined in section 3, the core elements of the TagItSmart project are the objects tagged with the SmartTags and the resulting information about the objects and the surrounding environment that can be retrieved, and not the description of the sensing elements of the NFC electronics or the dynamic inks. Another key element is the simple as well as composite services that can be enabled by the capabilities of these SmartTag enabled objects. The project is also developed core platform services such as notification services to user, which needs to be semantically annotated. As such, this deliverable takes the established concepts of physical and virtual entities as well as services, from EU IoT projects such as IoT-A, COSMOS etc. as its starting point. These concepts are revisited and described in section 3.1, with explanations for cases where TagItSmart extends or differs from the definitions from past projects. To support this, examples of services as considered in the different UCs are given in section 3.2.

The specialist domain knowledge, as relevant to some of the project UCs and which is not part of the current state-of-the-art, but impacts the semantic model definition, has been presented in section 3.3. At a first instance, it has resulted in significant enhancements to the VE model beyond the current state-of-the-art and to refinement of the user model. For the next steps in WP4, this domain knowledge will be translated into domain ontologies in cases where these do not exist in the state-of-the-art, but are required to realise the TagItSmart UCs. Moreover, the development of a domain ontology, for instance in the recycling domain, can help to homogenise information representation in an area where there is a lot of information fragmentation, as evident from section 3.3.2. This can then facilitate interoperability of different systems and allow seamless matching of product packaging information with recycling point description.

Next steps, also linked to semantic modelling in WP4, will include generation of model instances, guided by the required VEs, user instances and services from the different UCs. These may result in minor edits of the classes and properties in the existing VE, service and user models. Generation of model instances will be made possible by using a dedicated annotation tool. The tool will be accessible either for manual annotation (through the use of a Web-based form, to be filled in by developers or domain experts) or by programmatic means using a REST API. Efficient storage and retrieval mechanisms will be designed for the generated instances, which will constitute the inputs to the service composition process of WP4.

Section 7 - References

[1] Alessandro Bassi, Martin Bauer, Martin Fiedler, Thorsten Kramp, Rob van Kranenburg, Sebastian Lange, and S. Meissner, *Enabling Things to Talk*: Springer-Verlag, 2013.

[2] S. De, T. Elsaleh, P. Barnaghi, and S. Meissner, "An Internet of Things Platform for Real-World and Digital Objects," *Journal of Scalable Computing: Practice and Experience,* vol. 13, no. 1, pp. 45-57, May 2012.

[3] X. Jin, S. Chun, J. Jung, and K.-H. Lee, "A fast and scalable approach for IoT service selection based on a physical service model," *Information Systems Frontiers,* pp. 1-16, 2016.

[4] S. Chun, S. Seo, B. Oh, and K. H. Lee, "Semantic description, discovery and integration for the Internet of Things," in *Semantic Computing (ICSC), 2015 IEEE International Conference on*, 2015, pp. 272-275.

[5] W. Zhu, G. Zhou, I. L. Yen, and F. Bastani, "A PT-SOA Model for CPS/IoT Services," in *Web Services (ICWS), 2015 IEEE International Conference on*, 2015, pp. 647-654.

[6] D. Kelaidonis, A. Somov, V. Foteinos, G. Poulios, V. Stavroulaki, P. Vlacheas, P. Demestichas, A. Baranov, A. R. Biswas, and R. Giaffreda, "Virtualization and Cognitive Management of Real World Objects in the Internet of Things," in *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, 2012, pp. 187-194.

[7] B. Christophe, V. Verdot, and V. Toubiana, "Searching the 'Web of Things'," in *Proc. Fifth IEEE International Conference on Semantic Computing (ICSC'11)*, Palo Alto, CA, 2011, pp. 308 - 315.

[8] Y. Zhou, S. De, W. Wang, and K. Moessner, "Enabling Query of Frequently Updated Data from Mobile Sensing Sources," in *13th IEEE International Conference on Ubiquitous Computing and Communications (IUCC 2014)*, Chengdu, China, 2014.

[9] "World Geodetic System 1984 (WGS-84)," National Geospatial-Intelligence Agency 2005.

[10] OGC, "Open Geospatial Consortium (OGC) Sensor Web Enablement: Overview and High Level Architecture," OGC white paper, 2007.

[11] M. Compton, P. Barnaghi, L. Bermudez, R. G. Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W. D. Kelsey, D. L. Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. Page, A. Passant, A. Sheth, and K. Taylor, "The SSN Ontology of the Semantic Sensor Networks Incubator Group," *Journal of Web Semantics,* vol. 17, no. pp. 2–32, December 2012.

[12] S. Hachem, "Service-Oriented Middleware for the Large-Scale Mobile Internet of Things," PhD thesis, University of Versailles Saint-Quentin-en-Yvelines, 2014.

[13] Semantic Sensor Network Incubator Group (SSN-XG), "Report Work on the SSN ontology," June 2011. [Online], Available: https://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/

[14] H. Neuhaus and M. Compton, "The Semantic Sensor Network Ontology: A Generic Language to Describe Sensor Assets," presented at the the Pre-Conference Workshop on Challenges in Geospatial Data Harmonisation (AGILE 2009), Hannover, Germany, 2009.

[15] D. J. Russomanno, C. Kothari, and O. Thomas, "Sensor ontologies: from shallow to deep models," presented at the Proceedings of the Thirty-Seventh southeastern Symposium on System Theory, 2005.

[16] OGC, "OpenGIS® Sensor Model Language (SensorML) Implementation Specification," Open Geospatial Consortium, Inc., 2007.

[17] S. Kolozali, M. Bermudez-Edo, D. Puschmann, F. Ganz, and P. Barnaghi, "A Knowledge-Based Approach for Real-Time IoT Data Stream Annotation and Processing," in *Internet of Things (iThings), 2014 IEEE International Conference on,*

and Green Computing and Communications (GreenCom), IEEE and Cyber, Physical and Social Computing(CPSCom), IEEE, 2014, pp. 215-222.

[18] L. Spalazzi, G. Taccari, and A. Bernardini, "An Internet of Things ontology for earthquake emergency evaluation and response," in *Collaboration Technologies and Systems (CTS), 2014 International Conference on*, 2014, pp. 528-534.

[19] F. Wang and K. J. Turner, "An Ontology-Based Actuator Discovery and Invocation Framework in Home Care Systems," in *Ambient Assistive Health and Wellness Management in the Heart of the City*, M. Mokhtari, I. Khalil, J. Bauchet, D. Zhang, and C. Nugent, Eds., ed: Springer Berlin Heidelberg, 2009, pp. 66--73.

[20] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing As a Service Model for Smart Cities Supported by Internet of Things," *Trans. Emerg. Telecommun. Technol.,* vol. 25, no. 1, pp. 81-93, 2014.

[21] W. Wang, S. De, R. Toenjes, E. Reetz, and K. Moessner, "A comprehensive ontology for knowledge representation in the internet of things," in *Proc. IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, Liverpool, UK, 2012, pp. 1793-1798.

[22] S. De, P. Barnaghi, M. Bauer, and S. Meissner, "Service modelling for the Internet of Things," in *Proceedings of Federated Conference on Computer Science and Information Systems (FedCSIS)*, Szczecin, Poland, 2011, pp. 949-955.

[23] W3C, "OWL-S: Semantic Markup for Web Services ", W3C Member Submission 2004.

[24] C. Qu, F. Liu, M. Tao, and D. Deng, "An OWL-S based specification model of dynamic entity services for Internet of Things," *Journal of Ambient Intelligence and Humanized Computing,* vol. 7, no. pp. 73--82, 2016.

[25] P. Gomes, E. Cavalcante, T. Rodrigues, T. Batista, and Deli, "A Federated Discovery Service for the Internet of Things," in *Proceedings of the 2Nd Workshop on Middleware for Context-Aware Applications in the IoT*, Vancouver, BC, Canada, 2015, pp. 25--30.

[26] J. H. Kim, K. Kwon, K. D.-H, and S. J. Lee, "Building a service-oriented ontology for wireless sensor networks," in *Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science*, 2008, pp. 649-654.

[27] M. Ryu, J. Kim, and J. Yun, "Integrated Semantics Service Platform for the Internet of Things: A Case Study of a Smart Office," *Sensors,* vol. 15, no. 1, 2015.

[28] M. Maleshkova, R. G.A., A. Simov, A. B. Renie, and D. Liu, "Service Provisioning Platform Second Prototype," Deliverable D2.1.4 of the project SOA4All2010.

[29] C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data - The Story So Far," *International Journal on Semantic Web and Information Systems (IJSWIS),* no. Special Issue on Linked Data Heath, T., Hepp, M., and Bizer, C. (eds.), 2009.

[30] C. Pedrinaci, D. Liu, M. Maleshkova, D. Lambert, J. Kopecky, and J. Domingue, "iServe: a linked services publishing platform," in *7th Extended Semantic Web Conference, Ontology Repositories for the Semantic Web Workshop*, Heraklion, Greece, 2010.

[31] T. Vitvar, J. Kopecký, J. Viskova, and D. Fensel, "WSMO-Lite Annotations for Web Services," in *Semantic Web: Research and Applications*. vol. 5021, ed: Springer, 2008, pp. 674-689.

[32] J. Kopecky, K. Gomadam, and T. Vitvar, "hRESTS: An HTML Microformat for Describing RESTful Web Services," in *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology," WI-IAT '08*, 2008, pp. 619-625.

[33] K. Skillen, L. Chen, C. D. Nugent, M. P. Donnelly, and I. Solheim, "A user profile ontology based approach for assisting people with dementia in mobile environments," in *Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC*, 2012, pp. 6390-6393.

[34] G. Fischer, "User Modeling in Human Computer Interaction," *User Modeling and User Adapted Interaction,* vol. 11, no. 1, pp. 65-68, 2001.

[35] J. Kay, B. Kummerfeld, and P. Lauder, "Personis: A Server for User Models," in *Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, Malaga, Spain, 2002, pp. 201-212.
[36] D. Zografistou, "Support for context-aware healthcare in Ambient Assisted Living," MSc thesis, University of Crete, Heraklion, 2012.
[37] K.-L. Skillen, L. Chen, C. D. Nugent, M. P. Donnelly, W. Burns, and I. Solheim, "Ontological user modelling and semantic rule-based reasoning for personalisation of Help-On-Demand services in pervasive environments," *Future Generation Computer Systems,* vol. 34, no. pp. 97-109, 2014.
[38] iKaaS, "D2.1: 1st iKaaS Semantic Model, Knowledge Acquisition and Service Provision Toolbox," iKaaS public deliverable, 2016.
[39] S. De, B. Christophe, and K. Moessner, "Semantic Enablers for Dynamic Digital-Physical Object Associations in a Federated Node Architecture for the Internet of Things," *Ad Hoc Networks,* vol. 18, no. C, pp. 102-120, 2014.
[40] W. Wang, S. De, G. Cassar, and K. Moessner, "Knowledge Representation in the Internet of Things: Semantic Modelling and its Applications," *Automatika – Journal for Control, Measurement, Electronics, Computing and Communications,* vol. 54, no. 4, pp. 388-400, 2013.
[41] S. De, Y. Zhou, and M. K., "Ontologies and context modeling for the Web of Things," in *Managing the Web of Things: Linking the Real World to the Web*, Q. Y. Sheng M, Yao L, Benatallah B, Ed., 1 ed Burlington, Massachusetts: Morgan Kaufmann, 2017.
[42] A. Gyrard and C. Bonnet, "Semantic Web Guidelines for domain knowledge interoperability to build the Semantic Web of Things," Eurecom, 7 April 2014.