On Modeling the Physical World as a Collection of Things: the W3C Thing Description Ontology

Victor Charpenay[1][0000−0002−9210−1583] and Sebastian Käbisch[2][0000−0002−0544−4204]

[1] FAU Erlangen-Nürnberg, victor.charpenay@fau.de
[2] Siemens AG, sebastian.kaebisch@siemens.com

Abstract. This document presents the Thing Description ontology, an axiomatization of the W3C Thing Description model. It also introduces an alignment with the Semantic Sensor Network ontology and evaluates how this alignment contributes to semantic interoperability in the Web of Things.

Keywords: Web of Things · Thing Description · SSN · RDF

1 Introduction

The Web of Things (WoT) is an architectural principle that aims at bringing sensor and actuator data on the Web in order to increase interoperability between connected devices and develop arbitrarily complex mash-ups on that basis [19,4]. The World Wide Web Consortium (W3C) embraced that vision and recently started a standardization activity around WoT with two main outcomes: a set of architectural guidelines [12] and a model to describe 'things' and their interface, the Thing Description (TD) model [8].

As both specifications are about to be officially released, the present paper provides an analysis of the role played by RDF and other Semantic Web technologies in WoT. In particular, RDF shall improve the interoperability across sensors and actuators at the semantic level, such that autonomous Web agents can build their own representation of the physical world from data exposed by various WoT devices.

In practice, semantic interoperability in the TD model translates into an annotation mechanism such that TD documents—instances of the TD model serialized in JSON—link to RDF terms defined in domain-specific vocabularies. To permit it, the TD model was designed on top of a Web ontology, which is being introduced in this paper: the TD ontology. The paper also introduces a formal alignment between the TD ontology and the Semantic Sensor Network (SSN) ontology [6]. It is indeed expected that most relevant vocabularies will be defined as specializations of SSN, as suggested by recent trends in ontology engineering for WoT [2,1].

Both the TD ontology and its alignment with SSN were designed as per the requirement that the TD annotation mechanism should remain as easy to use

as possible, especially for developers with no particular knowledge of Semantic Web technologies. In practice, developers are only asked to provide semantic "tags" at several places of a TD document. As a consequence, the axiomatization we present in this paper favors simplicity over completeness. We provide an evaluation on its effectiveness at the end of the paper, by looking at a collection of TD documents that serves as a test set in the W3C standardization process.

The paper is structured as follows: Sec. 2 contextualizes the TD ontology by providing a short review of the state-of-the-art in WoT ontology engineering, Sec. 3 presents the vocabulary it defines and Sec. 4 introduces an alignment of the TD ontology with SSN. Finally, in Sec. 5, the W3C TD test set is being analyzed in more detail.

2 Ontologies for the Web of Things: State-of-the-Art

Over the last decade, research on WoT systems has moved from pre-defined sensor mash-ups to autonomous agents capable of selecting what sensor measurements to read and what actuator commands to activate to fulfill a global goal [18,11,13,3]. This agent-oriented vision for WoT requires a detailed ontological view on the physical world, such that agents can take informed decisions on what interaction to initiate.

In parallel, a significant effort has been put to providing Web ontologies for WoT [5]. The SSN ontology, recently standardized, is now the pivot to any ontology engineering work in the domain. By analyzing the network formed by alignments between more than 80 WoT ontologies, SSN stands out as the most central point in the network [2,1]. Among others, it aligns with the ontology for units of measure (OM) [17] and the ontology for Quantity Kinds, Units and Datatypes[3] (QUDT), which both provide an extensive list of physical properties in OWL. SSN also aligns with domain-specific ontologies, like the Building Topology Ontology (BOT) [16]. In particular, a range of ontologies derived from the Smart Appliance Reference (SAREF) ontology is currently under construction[4]. All these ontologies are designed with compatibility with SSN in mind [14].

While SSN and the ontologies that align to it model the physical world, the TD ontology shall provide metadata to guide autonomous Web agents in the network of interconnected devices that quantify it. In particular, it formalizes the concept of 'affordance', introduced in the next section.

3 The Thing Description Model

The TD model is a schema to which TD documents must comply when exposing the capabilities of a 'thing' on the Web [8]. We briefly presents its main components on an example. The following TD document, serialized in JSON, describes a lamp:

[3] `http://qudt.org/`
[4] `https://saref.etsi.org/`

```
{
  "@context": "https://www.w3.org/...",
  "id": "http://lamp.local/#it",
  "title": "Some lamp",
  "properties": {
    "state": {
      "type": "boolean",
      "forms": [
        { "href": "http://lamp.local/st" }
      ]
    }
  },
  "actions": {
    "on": {
      "output": { "type": "boolean" },
      "forms": [
        { "href": "http://lamp.local/on" }
      ]
    },
    "off": {
      "output": { "type": "boolean" },
      "forms": [{
        "href": "http://lamp.local/off" }
      ]
    }
  },
  "events": {}
}
```

The mandatory `@context` key makes this document a JSON-LD 1.1 document, such that it complies to the Linked Data principles and can be turned into RDF in a standard way [9]. In order for the 'thing' (here, the physical lamp) to become a Web resource, it can be assigned an IRI with the key `id`, an alias for the JSON-LD keyword `@id`.

The three keys `properties`, `actions` and `events` are the main elements of a TD document. Here, the lamp exposes its on/off state as a property (`state`), which can be changed by calling two actions (`on` and `off`). The on/off state could also have been exposed as an event. Every possible interaction with the lamp starts by submitting a Web form to the 'thing', as specified under the `forms` key. Here, agents must only know the target URI of each form (`href`) to start interacting with the lamp.

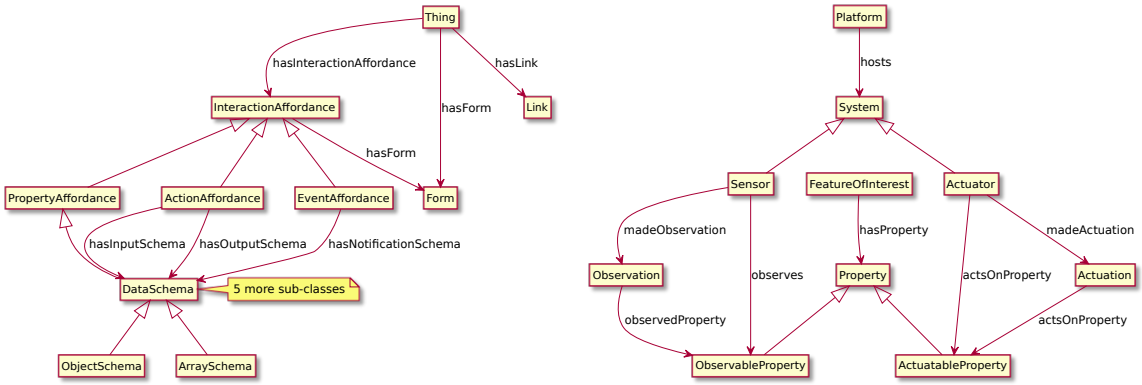
This approach centered around Web forms is inspired by the Representational State Transfer (REST) principles, fundamental to the Web. The values under `property`, `actions` and `events` are 'interaction affordances', which can be seen informally as "invitations" to interact with the 'thing'. The concept of affordance has a well-defined meaning in the context of REST: it relates to hypermedia controls, that is, links and forms embedded in a message [15].

3.1 Requirements

Requirements for the TD model have been collected in the WoT architecture recommendation document [12]. The main requirements can be found in Sec. 6.4 of that document as a series of assertions, which can be turned into formal axioms with little effort. All model elements listed in these assertions (i.e. the TD terminology) were turned into an RDF class. Assertions that include the keyword 'may' have been axiomatized using the lightweight schema.org semantics[5]. Assertions including the keyword 'must' have been turned into RDF shapes, using the SHACL language [10]. The later part is however not presented in this paper. One can indeed argue that it contributes little to semantic interoperability.

It is however important to introduce the RDF classes and properties of the TD model (loosely referred to as the TD ontology), in order to then provide an alignment with SSN, the key element of semantic interoperability. It is indeed

[5] `https://meta.schema.org/`



(a) Thing Description model

(b) Semantic Sensor Network model

Fig. 1: Overview of the TD and SSN ontologies

in SSN terms that the internal state of a 'thing' is to be specified. Contrary to other RESTful systems, the internal state of WoT systems is not purely informational but is instead derived from observing physical world objects. Before developing this aspect in Sec. 4, we first introduce the details of the TD ontology axiomatization.

3.2 Axiomatization

An overview of the classes and properties of the TD ontology is provided on Fig. 1a. The Figure shows sub-class relations and property relations whenever instances of two classes may be related in a TD document. The terms 'property affordance', 'action affordance' and 'event affordance' are respectively the abbreviation of 'affordance to retrieve/update a property', 'affordance to invoke an action' and 'affordance to subscribe to an event'.

As shown on the figure, the TD ontology refers to classes that may be used in another context. In particular, it relies on JSON Schema, a language under standardization[6], and on hypermedia controls (links and forms), which may be used in other RESTful systems, outside the scope of WoT. To encourage reusability, these two aspects were put in their own modules, separate from the TD core module[7]. We present them next.

Core Module The core module defines two main classes: the class `td:Thing`[8] is the entry point of the ontology and its instances are anything that provides

[6] `https://tools.ietf.org/html/draft-handrews-json-schema`

[7] a fourth module for security configurations is included in the W3C standard but not covered in this paper.

[8] all prefixes in the paper can be found on `https://prefix.cc/`.

affordances to interact with it (`td:InteractionAffordance`). The concept of interaction is further refined into three sub-classes: it can be a basic state transfer, i.e. a retrieval or an update of the exposed state of the 'thing', it can be an invocation with input parameters and expected result or it can be an asynchronous notification after subscribing to a particular event. Any of these three interaction patterns translate into a certain kind of affordance, which must include the appropriate metadata for an agent to be able to properly submit the associated form. The three affordance classes are `td:PropertyAffordance`, `td:ActionAffordance` and `td:EventAffordance`.

As per architectural requirements, an action may manipulate the exposed state of a 'thing' but it may as well leave it unchanged. Similarly, event notifications may include parts of the exposed state or not. Conversely, properties (which represent the exposed state of the 'thing') may be retrieved asynchronously if they are 'observable'. As a result, these concepts are *not* mutually exclusive. A state can be retrieved either as properties or via events, while it can be updated by manipulating properties or via action invocation. This versatility shall account for the diversity of communication paradigms that coexist on WoT.

All interaction affordances are composed of two kinds of objects: data schema, which are to be understood as specifications of abstract data structures, and hypermedia controls (forms). They are each described next.

Data Schema Module The data schema module is a port of JSON Schema to RDF. JSON Schema was favored over e.g. schema.org's property/value specification mechanism[9], mostly designed for HTML forms. As its name suggests, JSON Schema is a language whose type system relies on the basic JSON types (object, array, number, string, boolean and null), to which it adds the integer type. In RDF, each type becomes a class: `jsonschema:ObjectSchema`, `jsonschema:ArraySchema`, etc. The language also includes constraints specific to each type, like minimum and maximum values for numbers or a maximum length for strings. Contrary to schema.org's property/value specifications, JSON Schema is a recursive language, via the `jsonschema:properties` and `jsonschema:items` relations, for JSON values of the object and array type.

The goal of this ontological module is less to provide axioms for logical inference than to offer a simple transformation from JSON to RDF (using a JSON-LD standard processor). It is therefore merely a set of RDF terms. An alternative design would have been to embed JSON Schema definitions as literal, leveraging the newly introduced `rdf:JSON` datatype[10].

Yet, the chosen approach has mainly two benefits over literal schemas. First, sub-schemas can be semantically tagged in an individual fashion, which will be later illustrated in Sec. 4. Second, the RDF entities resulting from JSON-LD transformation could then carry both schema information and denote actual properties of physical world objects. An example is given later in Sec. 5.3.

[9] `https://schema.org/PropertyValueSpecification`
[10] `https://w3c.github.io/json-ld-syntax/#the-rdf-json-datatype`

Table 1: Competency questions for autonomous Web agents

Competency Questions	
Q1	How to identify affordances exposed by a 'thing' that have the same effects on the physical world?
Q2	How to identify 'things' that fulfill the same function?
Q3	How to map complex property affordances to simpler representations of physical world objects and their properties?
Q4	How to differentiate between active 'things' (like sensors) and passive 'things' (like feature of interest under observation)?

Hypermedia Controls Module Classical Web applications make use of two kinds of hypermedia controls: links and forms. Both links and forms include a target IRI and a "type". This type is alternatively called a *relation* type for links and an *operation* type for forms. The TD ontology defines operation types specific to WoT: `td:readProperty`, `td:writeProperty`, `td:invokeActtion`, `td:subscribeEvent` and a few others. The hypermedia controls module, however, only includes generic classes and properties that are needed to express links and forms in RDF. A link can also be thought of as a reification of an RDF triple, e.g. to add provenance or temporal metadata.

The JSON-LD context that maps terms from this module to JSON keys was designed in such a way that links have the same format as specified in JSON Hyper-Schema[11].

4 Alignment with the Semantic Sensor Network

4.1 Requirements

The TD model serves primarily communication purposes. The axiomatization presented in the previous section therefore concentrates on the concept of 'interaction affordance'. Contrary to what one may think at first sight, it provides only few axioms on the concept of 'thing'. Yet, as previously mentioned, the peculiarity of WoT sytems is that they do not have a purely informational state but rather maintain a virtual representation of the state of physical world objects (the actual 'things').

According to the state-of-the-art, the state of physical world objects will likely be modeled using ontologies aligned with SSN. It therefore calls for an alignment of the TD ontology itself to SSN, in order to derive an SSN "view" on instances of `td:Thing`.

To this end, we provide four competency questions from the point of view of Web agents that process TD documents (Table 1). Our assumption is that agents are autonomous and rely solely on the RDF statements included in a TD document to select affordances. An example of an affordance selection task is provided later in Sec. 5.3.

[11] `https://tools.ietf.org/html/draft-handrews-json-schema-hyperschema`

4.2 Axiomatization

An overview of SSN is given on Fig. 1b. The ontology is mostly centered around the concepts of 'observations' and 'actuations' performed by sensors and actuators. However, in the context of WoT, the most relevant classes are `sosa:FeatureOfInterest` and `ssn:Property`. These two classes roughly denote physical world objects and their properties (or characteristics). This basic object model is meant to be specialized for concrete domains of application, as is the case in the ontologies mentioned in Sec. 2.

The alignment between TD and SSN consists mostly in existential restrictions on the classes `td:Thing` and `td:InteractionAffordance`. These axioms, like the TD axioms themselves, were not designed for automatic inference but rather as 'may' statements on the SSN entities to include in TD documents. The two main alignment axioms are given below[12] (OWL Manchester syntax [7]):

```
Class: td:Thing
  SubClassOf: ssn:System or sosa:Platform or sosa:FeatureOfInterest
Class: td:InteractionAffordance
  SubClassOf: ssn:forProperty some ssn:Property
```

We review each in the following and then move on to a review of how competency questions can be addressed with SSN.

As provided in our alignment, a 'thing' can be any of the following: a sensor (for illuminance, temperature, air quality, etc.); an actuator (like a binary switch); a composite system; a platform that hosts a system (like a electronic board with pluggable sensors); a feature of interest (like a room). The list is not exhaustive. Yet, it covers all 'things' described in the W3C implementation report we review in Sec. 5.

SSN does not define systems, platforms and features of interest as mutually exclusive. In fact, having systems being themselves features of interest is a common pattern. It is the case e.g. of consumer electronics products like light bulbs or air conditioning units: they are connected devices and thus instances of `ssn:System` but they are neither sensors, nor actuators and they are more than simply a combination of both (because the coupling of sensing and actuation follows some internal logic). They can however be modeled as features of interest with their own properties (on/off status, wind speed, etc.).

The second alignment axiom we reported implies a restriction on what parts of a 'thing' should be exposed via affordances. Indeed, the axiom states that every affordance relates to the property of some physical world object, that is, to part of the physical world. In contrast, some properties like a software version number, a product ID or a writable label only belong to some informational space, without any tangible extent. Despite the fact that neither SSN axioms nor the alignment axioms are restrictive on what instances of `ssn:Property` should be, exposing informational properties as plain RDF statements should be favored over exposing affordances to these properties.

[12] other axioms can be found in the TD ontology documentation, served under its namespace URI: `https://www.w3.org/2019/wot/td`.

```
{
  "properties": {
    "state": {
      "ssn:forProperty": "_:status"
    }
  },
  "actions": {
    "on": {
      "ssn:forProperty": "_:status"
    },
    "off": {
      "ssn:forProperty": "_:status"
    }
  }
}
```

(a) Q1

```
[
  {
    "properties": {
      "light1": {
        "ssn:forProperty": "_:light"
      }
    }
  }, {
    "properties": {
      "light2": {
        "ssn:forProperty": "_:light"
      }
    }
  }
]
```

(b) Q2

```
{
  "properties": {
    "measurement": {
      "type": "object",
      "properties": {
        "temperature": {
          "ssn:forProperty": "_:temp"
        },
        "humidity": {
          "ssn:forProperty": "_:humid"
        }
      }
    }
  }
}
```

(c) Q3

```
[
  {
    "title": "Some sensor",
    "properties": {
      "measure": {
        "ssn:forProperty": "_:light"
      }
    },
    "sosa:observes": "_:light"
  }, {
    },
    "title": "Some room",
    "properties": {
      "measure": {
        "ssn:forProperty": "_:light"
      }
    "ssn:hasProperty": "_:light"
  }
]
```

(d) Q4

Fig. 2: Examples of annotation for each competency question

We now illustrate with examples how the competency questions of Table 1 can be addressed. In each case, the existential restrictions that exist between TD and SSN classes were "instantiated" with blank nodes that have `ssn:forProperty`-relations with TD entities. Portions of TD documents can be found on Fig. 2.

The first competency question (Q1) refers to the fact that the exposed state of a 'thing' may be retrieved via readable properties or events and updated via writable properties or actions, often exposed within the same TD document. The example on Fig. 2a makes the relation explicit between the `state` property affordance and the `on` and `off` action affordances, by pointing at the same actuatable property (an on/off status).

In this example, Web agents must also be able to differentiate between the `on` and `off` actions, as they may have the same signature. For that purpose, a set of basic command types can be found in SAREF ('turn on', 'turn off', 'toggle', 'set level', 'setp up', 'step down', etc.). This aspect is however not directly solvable by an alignment with SSN.

Regarding Q2, it is also possible to use properties of physical world objects as connectors between 'things'. For instance, two sensors may provide measurements for the same observable property, as on Fig. 2b. In this example, two light sensors observe the same illuminance property, e.g. because they are in the same room. This modeling is an approximation, as the two sensors cannot strictly provide measurements for the same illuminated surface but it suffices in most home automation applications.

Regarding Q3, the need to individually characterize parts of a data schema arises from the observation that certain developers expose data of different nature under the same URI. For instance, sensor boards designed to provide environmental data (temperature, humidity, compass direction, etc.) may expose only one complex property in which a value for all physical quantities is provided in a single JSON message.

It is the case in the example of Fig. 2c which offers only one `measurement` property affordance. The `temperature` and `humidity` values in the provided schema however point to SSN properties of different types (type statements are not shown for the sake of brevity).

The last competency question (Q4) is relevant to Web agents insofar as selecting interactions may require knowledge about the underlying sensing or actuation mechanism. In the last example (Fig. 2d), the upper definition describes the sensor that produces measurements itself while in the lower one, the room in which the sensor is located is exposed instead, hiding the sensing device from the agent. Yet, the two definitions include the same affordance.

Exposing (inanimate) physical world objects instead of sensors is relevant in certain cases, though. It is for instance simpler when the object is observed by numerous devices, whose measurements are combined into a single state. In the implementation report presented in the next section, a TD document describes a water tank that embeds three sensor: a water level sensor at its top, another at its bottom and a radar that provides the absolute level.

5 Evaluation

Every W3C standard must be associated with a technical implementation report, which proves interoperability between distinct implementations of the standard. Over the course of the standardization of WoT at W3C, the working group gathered implementation experience by putting existing devices on the Web and thus collected a number of TD documents that were then included in the implementation report for the TD model. In the following, we analyze these TD documents with respect to semantic interoperability and evaluate the role of our axiomatization on that aspect.

In this section, we first give an overview of the set of TD documents that are available. We then report on the semantic tagging approach chosen by implementers and evaluate whether the competency questions of Table 1 are properly addressed by this approach. To this end, we chose a specific task among those

Table 2: List of 'things' under test

Type	Unique	All	Type	Unique	All
Light switch	2	13	Blinds	2	2
Lamp	7	9	Camera	2	2
Illuminance sensor	5	5	Car	1	2
Generic switch	5	5	Pump	1	2
Motion sensor	4	4	Electric meter	1	1
Generic sensor board	4	4	Robot cleaner	1	1
Bulletin board	2	4	Industry automation model	1	1
Temperature sensor	1	3	Water tank	1	1
Buzzer	2	2	Boiler	1	1
Air conditioning unit	2	2	Medical device	1	1
Total				44	65

tested by the W3C working group, which consists in automatically selecting specific affordances from all affordances included in the set of TD documents.

5.1 The W3C Thing Description Implementation Report

The implementation report for the W3C TD specificaton[13] relies on a set of 95 TD documents that each implement specific aspects of the TD model. All examples metioned in this paper come from this test set.

Among the 95 documents, we identified 65 that relate to actual devices (or simulations). The other documents are synthetic and designed for pure testing. Table 2 provides a list of all devices under test. Most of them are small devices (lamp, illuminance sensor, switch, sensor board, electric meter) but the test set includes various other devices, like air conditioning units, blinds, cars and an industrial plant model. Some of the TD documents were generated from other specifications, standardized by other consortia like the Open Connectivity Foundation[14] and ECHONET[15]. One of the two cars exposes data via its on-board diagnostics interface, specified by the International Organization for Standardization.

Some of the TD documents are copies of each other, in which only `id` and `href` values differ. If we discard duplicates, there remains 44 unique sets of interaction affordances. They were designed by 8 distinct organizations (all members of the W3C working group). We could observe notable differences in the data schema definitions, despite referring to similar properties. In particular, the test set includes four different schema definitions for a brightness property: either a number in the intervals $[0, 100]$, $[0, 254]$ or $[-64, 64]$ or an enumeration of the 100 integers in the interval $[0, 100]$[16]. This observation motivates the need for some further input by WoT developers to guarantee interoperability.

Interoperability among these devices was tested in different scenarios by the working group members: home appliances are turned off when the owner leaves

[13] `https://w3c.github.io/wot-thing-description/testing/report.html`
[14] `https://openconnectivity.org/`
[15] `https://echonet.jp/`
[16] this uncommon representation is due to an automatic translation from ECHONET schemas to TD documents.

the house; industrial equipment is put in safety mode when an accident is detected; the electric consumption of devices in a large building is adapted to real-time electric supply. Other ad-hoc interoperability tests have been performed, e.g. between generic switches and various actuators (like a car's honk). For practical reasons, all interoperability tests were performed among specific instances of 'things', involving manual annotation, as opposed to relying on semantic annotations of TD documents. The next two sections address the question of pure semantic interoperability, without human intervention.

Note that all figures in the remainder of the evaluation are based on the 44 TD documents with unique sets of interaction affordances, rather than on all 65 documents.

5.2 Semantic Tagging

Every TD document is a JSON-LD document. Objects that map to RDF entities can therefore be added arbitrary statements in the JSON-LD syntax, as in the examples of Sec. 4. However, this principle seems to be hardly understood by developers with no particular knowledge of Semantic Web technologies. Consequently, the decision has been made among the group that semantic annotations be limited to the tagging of certain model elements with type statements, using the JSON-LD `@type` keyword. Such `@type` tags should be put on instances of three classes: `td:Thing`, `td:InteractionAffordance` and `jsonschema:DataSchema`. One implementation of the WoT standards documents them as "the names of schemas for types of capabilities a device supports,"[17].

Although developers are generally aware of the necessity of semantic tagging to increase interoperability in WoT, the concept of "schema" has remained ambiguous. To ease the tagging effort developers must provide, an initiative to provide a unified vocabulary adapted to TD documents has been launched in parallel to the W3C standardization activity. This initiative, referred to as iot.schema.org, aims at reproducing the success of schema.org in the WoT domain[18]. The upper-level classes of iot.schema.org are aligned with those of the TD model and are meant to later align both with schema.org and with ontologies like SSN and SAREF. It includes a total of 194 classes at the time of writing, some of them having no equivalent class in other WoT ontologies yet (e.g. for individual red/green/blue components of a color).

Some of the TD documents of the W3C implementation report include `@type` tags from iot.schema.org. An overview of how tagging was performed is given in Table 3. What the table first shows is that the majority of TD entities were still not tagged (84% of property affordances and 86% of data schemas). It also shows how many tags can be considered as erroneous because the corresponding URI does not exist in iot.schema.org (resource not found): they represent 39% of the tags.

[17] `https://iot.mozilla.org/wot/#type-member`
[18] `http://iotschema.org/` (incubated domain name for `http://iot.schema.org`)

Table 3: Summary of `@type` tagging in the W3C test set (t: thing, p: property, a:action, e: event, sc: data schema, x: resource not found, ∅: no tag)

	t	p	a	e	sc	x	∅
td:Thing	10	**1**				**12**	**33**
td:PropertyAffordance		29		**2**	11	**6**	**264**
td:ActionAffordance			5			**7**	**41**
td:EventAffordance				1		**1**	**4**
jsonschema:DataSchema		**2**			14	**22**	**258**

Some of the erroneous tags are undoubtedly spelling mistakes (like `iot:PropertyChangeEvent` instead of `iot:PropertyChangedEvent`), which appropriate tooling can mitigate. However, some of these tags seem to result from conceptual discrepancies with the original iot.schema.org vocabulary. For example, the tag `iot:RunModeChanged` does not exist in iot.schema.org although `iot:RunMode` does. Moreover, the former is used as annotation for a data schema, although its name suggests it should apply to event affordances.

This assumption is supported by the fact that the confusion between properties and events can also be observed for a tag that does exist in the vocabulary: `iot:MotionDetected`. This class is defined as a sub-class of `iot:Event` but despite this axiom, two independent contributors to the implementation report used it to tag an instance of `iot:Property`, which is semantically inconsistent. We identified two more cases of semantic inconsistency: a sub-class of `iot:Property` used to tag a 'thing' entity and two other sub-classes of `iot:Property` to annotate data schemas.

One of the original design choices of iot.schema.org was to introduce a certain level of redundancy to ensure developers would find the appropriate tag for a given level of abstraction. As a result, the vocabulary includes e.g. the classes `iot:IlluminanceSensing`, `iot:Illuminance` and `iot:IlluminanceData`, to respectively tag a 'thing', a property affordance and a data schema. What our review suggests, however, is that developers tend not to make the distinction.

To mitigate the risk of confusion, an alternative design would consist in keeping property classes only: `iot:Motion` and `iot:Illuminance`, for example. Our alignment of the TD ontology with SSN becomes then relevant to retain conceptual consistency. These two classes can indeed be both defined as sub-classes of `sosa:ObservableProperty`.

It is worth mentioning two further observations on that topic. First, most entities in TD documents are either data schemas or property affordances (which are also instances of data schemas, as constrained by the TD model's RDF shapes). The modeling effort of iot.schema.org should therefore give priority to properties. Second, although Table 3 shows that 'things' are entities with the highest tagging ratio, all 'things' with tags also include tags at the affordance or the data schema level. It suggests again that properties are the most important entities in a TD document. Existing WoT ontologies already include a number of classes for the physical properties of physical world objects (OM and QUDT, in the first place). Our alignment with SSN details how affordances and properties should relate.

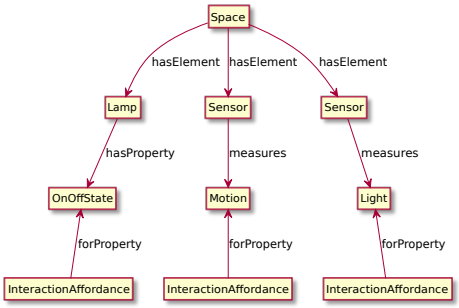


Fig. 3: Query pattern for the affordance selection task

In the next section, we review how SSN statements can be derived from `@type` tags in order to perform an affordance selection task. Tags originate from existing WoT ontologies, as per the above conclusion.

5.3 Affordance Selection Task

The following three-step process was part of a larger home automation scenario in the W3C implementation report: 1. start if motion is detected in some room, 2. turn on the lamp in this room, 3. after some time, end if illuminance is above a certain threshold, 4. otherwise, retry.

This relatively simple mash-up requires affordances of three different kinds to be properly selected. Selection can be reduced to a single query that matches the pattern depicted on Fig. 3, which is formulated using SSN and BOT terms, mostly. This choice is motivated by our review of the state-of-the-art. The goal of the evaluation for this task is to compare a manual affordance selection with an automatic procedure which only takes `@type` tags as input.

We manually annotated the relevant TD documents with SSN statements, assuming all 'things' were located in the same room. The task involves 16 'things', about one third of the whole test set, and 21 affordances in total: 3 for motion, 14 for updating the on/off state of the lamp and 4 for illuminance. We used five classes for the annotation: `saref:Motion`, `saref:LightingDevice`, `saref:OnOffState`, `saref:OnCommand` and `saref:Light`. We assume developers provide the appropriate `@type` tagging for these classes. Given that baseline, we can compare our manual annotation with an inference-based procedure which derives SSN statements from a combination of TD statements and type statements. We further assume that the classes listed above are aligned with SSN.

The first inference rule that is needed for that purpose is given by the following axiom (OWL Manchester syntax):

```
1  Class: :SelfAffordance
2    SubClassOf: (ssn:PropertyAffordance or jsonschema:DataSchema) and
3                ssn:Property
4  Class: :SelfAffordance
5    SubClassOf: ssn:forProperty Self
```

This rule states that whenever a property affordance or a data schema is annotated with a sub-class of `ssn:Property`, it then has an `ssn:forProperty`-relation with itself. In other words, it is as if the SSN property carries its own schema information.

We now look at the competency questions of Table 1 one by one to bridge the gap between `@type` tags and our baseline. Looking at Q1, the challenge is here to find the correspondance between property affordancs on the (writable) on/off state of a lamp and actions affordance to turn the same lamp on. This occurs in 4 TD documents. This equivalence between property and action can be turned into an inference rule that is triggered whenever the tags `saref:OnOffState` and `saref:OnCommand` co-occur in a TD document[19].

Interestingly, 2 TD documents expose two distinct property affordances for the same on/off state. The rule above would also merge them if we add the constraint that lamps can only have one on/off state. However, another TD document exposes a total of 4 affordances tagged with `saref:OnCommand`. Yet, not all these affordances are for the same property. The 'thing' is indeed composed of three LED strips which can be individually controlled. Only one affordance turns all LEDs on. Inference remains sound e.g. if the 'thing' is tagged as a platform instead of a feature of interest (such that the rule is not triggered). It is however a fragile assumption.

As for Q2, the relation between 'things' that is required for our lighting task is a relation stating that devices are in the same building space (to prevent that the process runs into infinite loops). Because this aspect is not addressed in the W3C implementation report, we assume that all devices are in the same room for the sake of this evaluation, without further annotation.

Regarding Q3, `@type` tagging is satisfactory. It does not require further inference. However, `@type` is necessary: 7 TD documents embed the values required for the task in complex schemas.

Finally, regarding Q4, we can observe that most TD documents follow the same pattern: if the 'thing' is a feature of interest, it provides affordances for its own properties; if it is a sensor or an actuator, its affordances refer to the properties it measures or acts on. Again, these rules can be expressed as OWL axioms such that SSN relations be inferred from `td:hasInteractionAffordance`-relations.

In a similar fashion to the inference rule fulfilling Q1, the inferred statements would not hold in the general case. For instance, when a 'thing' is a complex system as are a car or an industrial automation system, if it is tagged as an instance of `sosa:FeatureOfInterest` as well, it would be inferred that it exposes affordances to its own properties. However, it would be more likely that the exposed properties are those of its subsystems. It remains sound in our lighting task, though.

[19] because most inference rules mentioned in this section are tedious to write in OWL, we do not represent them in the paper; they can be found online, along with the set of annotated TD at `https://www.vcharpenay.link/talks/td-sem-interop.html`.

To summarize our comparison between a manual annotation and automatically inferred statements from `@type` annotations, we can state that all 21 affordances could be correctly selected as expected. However, as noted while looking at Q1, 3 more affordances would be wrongly selected as well depending on the value of a single `@type` tag. On a different aspect, to arrive at the expected result, it is worth noting that only a tractable OWL fragment was needed (OWL RL), which provides certain guarantees in terms of scalability.

6 Conclusion

This paper, along with introducing the TD ontology, underlines that the alignment with SSN is essential for autonomous agents to be developed in future WoT systems. Most of the standardization conducted by the W3C was put on interoperability at the protocol level. The TD standard that results from this work includes certain assumptions on how interoperability at the semantic level can also be guaranteed. In particular, it is assumed that relying on the JSON-LD syntax and allowing for `@type` tagging will allow agents to implement arbitrarily complex form selection procedures to drive applications.

This paper provides an initial evaluation of this claim, based on the TD documents included in the W3C implementation report associated with the standard. Under certain conditions, a lighting process involving motion sensors, lamps and illuminance sensors could be executed on the sole basis of `@type` tags, provided that OWL axioms specify how SSN statements can be inferred from these annotations (Sec. 5.3).

Even in this simple task, certain limitations could be shown: substituting one tag with another may threaten the soundness of inference. Mitigating that risk will require appropriate tooling to show WoT developers the potential effects of their tagging when 'things' are combined. In particular, members of the W3C working group on WoT expressed several times the lack of tools that can detect semantic inconsistencies introduced by `@type` tags. A few inconsistencies could indeed be shown in the implementation report, as mentioned in Sec. 5.2. However, we could identify an alternative conceptualization, centered around SSN's features of interests and their properties, that could help limit these inconsistencies.

As the W3C is already preparing for the future version of the TD standard, the Semantic Web community may embrace this question of tooling to improve the quality of input annotations.

Acknowledgments. The authors would like to thank all participants of the W3C WoT Working Group, particularly María Poveda-Villalón and Maxime Lefrançois, for their contribution to the TD ontology. This work was partially funded by the German Federal Ministry of Education and Research through the MOSAIK project (grant no. 01IS18070-A).

References

1. Charpenay, V.: Semantics for the web of things: Modeling the physical world as a collection of things and reasoning with their descriptions (2019)
2. Charpenay, V., Käbisch, S., Kosch, H.: Introducing thing descriptions and interactions: An ontology for the web of things. In: Joint Proceedings of SR and SWIT 2016. vol. 1783, pp. 55–66 (2016)
3. Ciortea, A., Mayer, S., Michahelles, F.: Repurposing manufacturing lines on the fly with multi-agent systems for the web of things. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018. pp. 813–822 (2018)
4. Guinard, D., Trifa, V.: Towards the web of things: Web mashups for embedded devices. In: Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain. vol. 15 (2009)
5. Gyrard, A., Bonnet, C., Boudaoud, K., Serrano, M.: LOV4IoT: A second life for ontology-based domain knowledge to build semantic web of things applications. In: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud). pp. 254–261. IEEE (2016)
6. Haller, A., Janowicz, K., Cox, S., Le Phuoc, D., Taylor, K., Lefrançois, M.: Semantic sensor network ontology, `https://www.w3.org/TR/vocab-ssn/`
7. Horridge, M., Patel-Schneider, P.F.: OWL 2 web ontology language manchester syntax (2009)
8. Kaebisch, S., Kamiya, T., McCool, M., Charpenay, V., Kovatsch, M.: Web of things (WoT) thing description, `https://www.w3.org/TR/wot-thing-description/`
9. Kellogg, G., Champin, P.A., Longley, D.: JSON-LD 1.1, `https://www.w3.org/TR/json-ld11/`
10. Knublauch, H., Kontokostas, D.: Shapes constraint language (SHACL), `https://www.w3.org/TR/shacl/`
11. Kovatsch, M., Hassan, Y.N., Mayer, S.: Practical semantics for the internet of things: Physical states, device mashups, and open questions. In: 2015 5th International Conference on the Internet of Things (IOT). pp. 54–61. IEEE (2015)
12. Kovatsch, M., Matsukura, R., Lagally, M., Kawaguchi, T., Toumura, K., Kajimoto, K.: Web of things (WoT) architecture, `https://www.w3.org/TR/wot-architecture/`
13. Käfer, T., Harth, A.: Rule-based programming of user agents for Linked Data. In: Workshop on Linked Data on the Web. LDOW 2018 (2018)
14. Lefrançois, M.: Planned ETSI SAREF extensions based on the W3C&OGC SOSA/SSN-compatible SEAS ontology patterns. In: Workshop on Semantic Interoperability and Standardization in the IoT, SIS-IoT. p. 11p. Proceedings of Workshop on Semantic Interoperability and Standardization in the IoT, SIS-IoT (2017)
15. Pautasso, C., Wilde, E., Alarcon, R. (eds.): REST: Advanced Research Topics and Practical Applications. Springer New York (2014)
16. Rasmussen, M.H., Lefrançois, M., Pauwels, P., Hviid, C.A., Karlshøj, J.: Managing interrelated project information in AEC knowledge graphs. Automation in Construction **108**, 102956 (2019)
17. Rijgersberg, H., van Assem, M., Top, J.: Ontology of units of measure and related concepts. Semantic Web Journal **4**(1), 3–13 (2013)

18. Verborgh, R., Haerinck, V., Steiner, T., Van Deursen, D., Van Hoecke, S., De Roo, J., Van de Walle, R., Gabarro, J.: Functional composition of sensor web APIs. In: SSN. pp. 65–80 (2012)
19. Wilde, E.: Putting things to REST (2007)