Semantic Interoperability

Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Bartlomiej Solarz-Niesłuchowski, Paweł Szmeja, Katarzyna Wasielewska

Abstract Interoperability, on the semantic level, deals with shared understanding of data, between IoT artifacts. Positioned on top of the *syntactic* layer, semantic interoperability facilitates solutions to problems that arise after the data is in a common format, with syntax understood by all participants. Provisioning of compatibility between not directly compatible data structures, representations, conventions and standards, falls strictly under the responsibility of semantic methods.This chapter introduces the INTER-IoT perspective on data semantics and summarizes it's achievements in the field of semantic interoperability. Being a generic and far-reaching solution, the syntactic compatibility challenge is also mentioned, as it is a necessary precondition to comprehensive data interoperability suite.

1 Introduction

While the outlook for the Internet of Things (IoT) remains positive, its underlying vision, of an all-encompassing ecosystem, remains unfulfilled.The pervasiveness of so-called *silos* is, in no small part, due to differences in requirements on data semantics introduced, independently, by joining systems. Whether caused by varying domain requirements and perspectives, legacy software, lack of foresight, or deliberate vendor-locking, IoT artifacts usually "speak different languages", understood exclusively "within their cliques". When large(r)-scale IoT ecosystems are developed, different silos that are (physically and/or virtually) close to each other, often require meaningful communication. Here, semantic interoperability layer offers possibility of shared understanding of data. Application of a semantic

Maria Ganzha
Systems Research Institute Polish Academy of Science, Warsaw, Poland e-mail: maria.ganzha@ibspan.waw.pl

solution allows implementation of valid, purposeful, and useful conversation, while protecting internal use of varying data syntax, models, and semantics.[52]

Here, note that it is very rare that *explicit* semantics, using formal representation of the domain, in the form of an *ontology*, is available. Often, the meaning of data is *implicit* and, for instance, buried in system documentation. Fortunately, even in such case it is possible to "make implicit semantics explicit" (i.e. establish an ontology that represents it) and instantiate appropriate translators between the internal data model and its ontology-based, semantically-enriched counterpart. Readers interested in how this preliminary step towards semantic interoperability can be performed are welcome to consult [37, 39], and references provided there.

In the layered model of interoperability [24] semantics is positioned above *syntactic interoperability* [39], which ensures common understanding of the *structure* of data. Since the objective of this chapter is to present the INTER-IoT approach to *semantic* interoperability, and because syntactic interoperability is its prerequisite, in what follows, we shall assume that: (i) (one way or another) IoT artifacts have explicit semantics, represented as an ontology; (ii) artifacts participating in the ecosystem, have already achieved the syntactic interoperability [53].

1.1 Towards semantic interoperability

To provide the context for this chapter, let us assume that a number of IoT artifacts has to be "associated" to instantiate a new IoT ecosystem. Naturally, the case when an existing IoT ecosystem is to be enriched, by adding new artifacts (e.g. sensors/platforms/services) to deliver new functionalities, is also covered.

In general, there are multiple ways of achieving semantic interoperability within the ecosystem. The simplest is to implement the same semantics (use the same ontology) across the ecosystem. However, it would mean that each artifact would have to comply to the common ontology, and possibly change its preexisting semantics to fit the unified vocabulary and schema. In a structure, where no party is privileged, and no one can enforce such far-reaching changes, such approach is not feasible. [49]

Semi-interoperability may be achieved when many artifacts write data to the same storage that has a specific semantics. In this case, they all need to translate the data one-way, on their own, from their own semantics into the selected one. This approach, however, is mainly applicable to data-fusion and does not support inter-artifact communication. Furthermore, the question "which ontology should be selected for the common repository" remains unanswered.

Partial interoperability is (at least theoretically) achievable when artifacts share an upper ontology as a core "module" of their ontologies. This level of interoperability, however, is hardly applicable to IoT ecosystems, mainly because of their vast (internal) heterogeneity. Simply said, in order to be fully functional, highly specialized applications in IoT require higher level of interoperability than what, in most realistic cases, an upper ontology can offer. Here, it should be also noted that the heterogeneity of semantic representations occurring in IoT-related

domains (see, for instance, [35]) further diminishes real-world applicability of this approach.

Even though, it is not strictly necessary from the technical point of view, the INTER-IoT approach assumes that the joining process should be "non-invasive" for the participating artifacts. In particular, it means that no *internal* adjustments should be required in the process of forming/joining the ecosystem. To achieve this goal, INTER-IoT uses *communication channels* which facilitate flow of information between artifacts. Since the artifacts may "speak different languages" it is natural to expect that the communication channels not only form communication "media" but perform *semantic translation* as well.

1.2 IoT Case Study

To illustrate the proposed approach, let us consider a logistic and delivery system, operating in an IoT-enabled city (see, Figure 1). There are several delivery companies (*iotDelivery*), operating in the area, that deliver/pick up goods to/from specific locations. Their goal is to dynamically optimize routes and operation time of their trucks. Therefore, each monitors truck positions and routes and chooses a specific vehicle for selected job. The choice is based on: (i) current traffic information, (ii) availability of parking slots near the delivery/pick-up point. Traffic information is provided by *dTraffic* company. This company analyses and publishes information gathered by drones monitoring city traffic. Note that drones can come from different vendors and, consequently, support different communication standards. Each drone should, periodically, send two types of messages, containing: (i) its position and battery level, and (2) traffic congestion information. *dTraffic* gathers and processes this information, to publish traffic reports. Obviously, *dTraffic* needs to send messages to control drones, e.g. request to return to base for charging (including coordinates of selected base station). Since platform used by *dTraffic* has to operate 24/7, recycling/connecting new devices must be seamless. Finally, parking lots companies (*iParking*) manage parking spaces in specific regions. *iParking*'s publish information about availability of free slots, and receive reservation requests from *iotDelivery* companies. Reservation messages contain, car ID/RFID and arrival time. Upon truck arrival, *iParking* provides slot number. Note that each *iParking* instance can be managed by different entities and use different communication standards.

Here, interoperability is needed for: (i) *dTraffic* company communicating with its drones – drones should be able to send messages in their native format, while *dTraffic* should communicate with them in a common selected format, not having to directly use/support all vendor-specific standards; (ii) *iotDelivery* cooperating with different trucks – should have simplified communication, as in the case of *dTraffic*; (iii) *iotDelivery* gathering information from *dTraffic*, trucks, *iParking*'s, and sending reservations to *iParking*'s. In first two cases, the interoperability mechanism should seamlessly handle addition of new drone/truck vendors. The last example requires

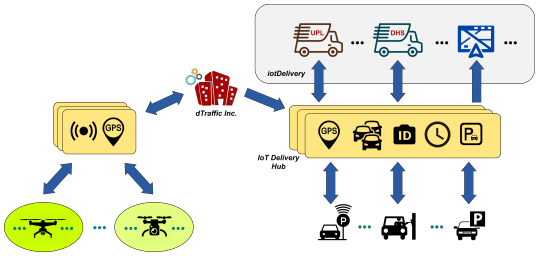

Fig. 1 Smart delivery and logistics interoperability scenario.

creation of an *IoT Delivery Hub*, where information between parking lots, trucks, delivery companies and traffic monitoring can be exchanged and understood.

The *IoT Delivery Hub* is a place where information is exchanged between stakeholders: trucks, *iParking* platforms, *iotDelivery* platforms, and *dTraffic* platform. Interoperability should enable any *iotDelivery* platform to receive information from different parking lots with parking spaces availability in a way that is independent from the original semantics that these platforms use internally. Moreover, when *iotDelivery* platform wants to make a reservation at a parking lot it should be able to send it in its natively supported semantics. Additionally, *IoT Delivery Hub* consumes data from *dTraffic* platform with current city congestion status, and translates data exchanged between trucks and *iotDelivery* platforms. Note that real-life scenario can be even more complex because trucks may cooperate with different delivery companies.

When designing the shared data model, in the *IoT Delivery Hub*, the following concepts should be taken into consideration:

- *Geolocation* – for expressing trucks location, parking lot and parking place location, and to identify areas with different traffic levels,
- *Time* – for defining reservation time slots,
- *Traffic* – for describing traffic congestion level in a given area,
- *Logistics* – for trucks identification, description of delivery orders,
- *Parking* – for describing parking availability.

Clearly, most of data publishing/exchange happening within the use-case, has a *streaming nature*. Therefore, the interoperability solution should enable translation of *streams* of messages exchanged between artifacts as well.

With the scenario in mind, we proceed as follows. First, we outline the semantic interoperability problem, and possible approaches towards solving it. In Section 2, we consider standards and ontologies that are available on the market, including the domain of our example scenario. Next, in Section 4, we discuss how to approach semantic translation, and what is the role of ontology *alignments* in the process. Then, we describe a specific format for persisting mappings/alignments between ontologies, and discuss the Inter-Platform Semantic Mediator (IPSM) – tool for performing alignment-based semantic translation. Finally, in Section 7, we return to the example scenario, to describe its realization using INTER-IoT approach.

2 Ontologies in the Internet of Things

An *ontology* [45], in a broad sense, is a way of representing and describing knowledge, and in more "applied" terms – is a way of representing data together with *metadata*. It can contain information about both concrete instances of data (individuals) and structural information about data, usually confined to a domain of interest. In this chapter, systems are *semantically interoperable* if a sent message can be (in a practical, not theoretical, way) expressed in terms of the ontology of the receiving system.

A formal way of expressing semantics is the Web Ontology Language (OWL)[1]. OWL ontologies are often modular. Here, *Horizontal* modules are subsets of ontologies, often very loosely (or not at all) connected. They describe different areas of the domain of interest, that reside on roughly the same level of abstraction. For instance, in the scope of logistics, a geolocation module is independent from a module describing the cargo. While both modules will likely be used to describe an instance of a physical container, there is no formal connection between the location of a container and its type, weight, or color. Moreover, a geolocation module is not tied to transportation or logistics, as it may be used for other physical entities, and even in entirely different domains. In summary, horizontal modules are not dependent on each other, separable, and sometimes completely orthogonal.

Vertical modules, on the other hand, build "on top of each other" and form a strict top-down hierarchy. The "level of a module" corresponds to its relative level of abstraction. Here, most general ontologies are "on top", and most specific ones "at the bottom". Overall, vertical modularity, based on levels of abstraction, is a backbone of semantic interoperability. High-level ("top") ontologies (or modules) contain general terms that are being "specified" by lower ontologies. An example, central to the IoT, are "device" ontologies that describe, in general terms, IoT devices (sensors, actuators, smart and mobile devices, etc). These ontologies are then extended by domain-specific (or application-specific) ontologies. In the eHealth domain, for instance, lower ontologies may add types of specific devices, e.g. patient monitoring tools, automated medicine applicators, etc. Transportation devices, on the other hand, may include GPS or speed sensors, port crane actuators, etc. A canonical model of semantic interoperability assumes that the same high-level device ontologies are used in both cases. Hence, without additional effort, there is a set of terms understood across domains. For a more detailed explanation of ontology modularity, see [47].

To bring semantic methods to IoT, availability mature/standardized, IoT-specific and domain-specific, ontologies is crucial. Here, note that many ontologies were developed within research projects and remain as prototypes, often incomplete, or abandoned (upon project end). We will thus briefly describe the notable exceptions.

[1] `https://www.w3.org/TR/owl2-overview/`

2.1 IoT core ontologies

Proper semantic treatment of sensors, sensor networks, actuators, and their operations, i.e. observations and actuations, is of fundamental importance for the IoT applications. Many ontologies for describing these concepts/entities have been proposed. Some of them evolved over time to accommodate changing scope, target audience, and technological landscape. Usually, for obvious reasons, the IoT-dedicated ontologies are also combined with/utilize other, high-level ontologies (ontological "modules"), such as ontologies of geolocation (e.g., LinkedGeoData [26], GeoSPARQL [9], or WGS84 [1]), units of measure (e.g., QU[2], OM[3], or SWEET units[4]), time (e.g., Time OWL[5]), or provenance (e.g. PROV-O [18]).

The W3C SSN [25, 29, 51] is an ontology or, actually, a suite of ontologies for describing sensors, their accuracy and capabilities, observations, methods used for sensing, and sensor deployment. The original version of W3C SSN turned out too "heavy-weight" for many smart devices, and did not cover concepts, which became important over time, such as actuators and actuation, or samples, samplers, and sampling. Therefore, recently, a new version of the SSN ontology was proposed. The W3C SOSA/SSN[6] is a modular ontology (see Figure 2), providing the required extensions. After the redesign, (the new version of) SSN became an extension of the kernel module, called SOSA (*Sensor, Observation, Sample, Actuator*).

IoT-Lite [28], is a light-weight ontology for representing IoT specific concepts. It specializes the SSN Device concept, and adds representation for objects, services, and actuators. It deal with geolocation, by using WGS84 ontology. The IoT-Lite focus on key IoT concepts directly supports semantic interoperability between IoT platforms.

Another notable ontology for IoT is the ETSI SAREF[7]. It models the domain of smart appliances and household devices, although there are many extensions that bring this ontology into the domains of energy, environment, and many others.

Many IoT platforms and middleware software solutions are also "riding the semantic wave" and introducing support for ontologies by either (i) adopting existing standards (e.g. FIWARE support for schema.org ontologies in its data models[8]), or (ii) authoring endemic ontologies (like oneM2M Base Ontology[9]). Readers interested in an overview and a more in-depth discussion of the IoT related ontologies can consult [36].

---

[2] https://www.w3.org/2005/Incubator/ssn/ssnx/qu/

[3] https://github.com/HajoRijgersberg/OM

[4] https://github.com/ESIPFed/sweet

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

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

[7] http://ontology.tno.nl/saref/

[8] https://fiware-datamodels.readthedocs.io/en/latest/guidelines/

[9] http://www.onem2m.org/technical/onem2m-ontologies

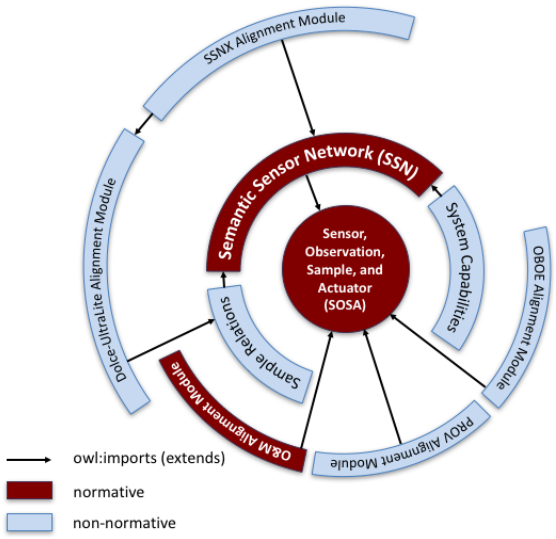

Fig. 2 SOSA/SSN modular structure (source – W3C).

2.2 Domain-specific ontologies

Virtually every IoT-based application/system needs to consider domain-specific ontologies. They accompany IoT ontologies to form a complete view of modeled and exchanged information. We will now briefly discuss some of them, representing several application domains. First, let us focus on transportation and logistics, as related to our sample use case. It was also central to one of pilot deployments of INTER-IoT. Here, ontologies span business perspectives of freight and production companies, transportation hubs (e.g. airports, train stations, ports), mass transit, transport infrastructure, personal and business travel, etc. Interestingly, in logistics in particular, many ontologies cover specific (and narrow) areas and very rarely describe a broad view of the domain. Furthermore, in many cases, work on logistics ontologies ended at the design phase (see, [46, 48]).

Here, one of the most-known ontologies is OTN (Ontology of Transportation Networks; [17, 42]) – a top-level ontology, produced in the REWERSE [20] project. It models general concepts of transportation, traffic networks and locomotion, as well as describes aspects of transportation relevant to, for instance, smart city or smart highway systems. The OTN is a realization and extension of the GDF [8] – Geographic Data Format (ISO specification) – as a formal OWL ontology. It was used in [43], to facilitate ontology-driven interoperability between urban models.

Let us also mention a Logistic Grid Ontology [15] that represents a service-oriented approach to logistics, realizing the idea of combining semantic technologies and cloud services, in order to enable semantically-driven description and application of logistic processes (see, [44]). It was developed within the LOGICAL project [11],

aim of which was to "enhance the interoperability of logistics businesses." A cloud service [12] utilizing the Logistic Grid Ontology was one of the results of this project.

Finally, LogiCO (and it's extension LogiServ) ontologies [13, 14] contain core concepts and properties from the domain. They model business activities (such as Transport, Transshipment, Load, Discharge, Storage, Consolidation, Deconsolidation, etc.) and their properties as the basis for specifying logistic services.

There are also ontologies, which cover other concepts related to transportation and logistics. The *Transport Disruption Ontology* [22], for example, is devoted to events, which can have a disruptive impact on travel planning. It is based on analysis of published disruption information, described in the DATEX II [4] specification.

We will now give a brief overview of other domains with knowledge modeled as ontologies. Some of them are related to INTER-IoT pilot use cases e.g. geospatial data, whereas some were used in INTER-IoT open call projects e.g. weather data.

In the medical domain, biological and biomedical ontologies are within the OBO Foundry [16] and the BioPortal [2]. These are mostly very specialized ontologies or vocabularies, e.g. ICD10, ICD11, SNOMED CT. Standardization bodies, such as HL7, support ontological representation of their standards, e.g. FHIR (Fast Healthcare Interoperability Resources) [6]. In INTER-IoT mHealth pilot we used a set of UniversAAL ontologies [23] as domain ontologies (with extensions). They were originally designed for UniversAAL platform and allow to describe measurements.

Ontologies from weather and meteorology domain include: (i) Linked Earth Ontology [10] that provides a common vocabulary for annotating paleoclimatology data; (ii) SEAS WeatherOntology [21] that allows to describe weather conditions, e.g. temperature, weather, sunrise, sunset; (iii) Climate and Forecast (CP) features [3] that offers representation of generic features defined by Climate and Forecast (CF) standard names vocabulary.

In IoT deployments, concepts related to geolocation and positioning are frequently required. Here, the most advanced ontology and geographic query language seems to be GeoSPARQL [9]. They allow to describe complex geospatial objects and query them using a set of built in functions. Other, simpler but still suitable for many use cases ontologies include W3C Basic Geo Vocabulary [1] and GeoRSS[10].

Overall, ontologies have been designed for various domains – either as simple vocabularies, or as more expressive conceptual models. The decision, which one to use should be based on the requirements analysis and detailed information concerning all data models that are involved. However, it is clear that, for interoperability, abundance of ontology choices may also turn out to be a challenge rather than help.

INTER-IoT, as a generic interoperability solution for IoT, in its scope, covers IoT entities that go far beyond a typical IoT platform, or service. Although, from a semantic perspective, support for sensors and actuators, deployments, physical locations, simple measurements, etc., is common, it is very rare that a given platform considers a broader ecosystem. Hence, the interoperability problem necessarily includes not just multitude of devices, but also multiple platforms, services, middlewares, users, etc. Each of then needs to be treated with the same attention

[10] `http://www.georss.org/`

as devices. Hence, we have developed the *GOIoTP* ontology, described in the next section.

3 Generic Ontology for IoT Platforms

Among the semantic products of INTER-IoT one can find two closely related ontologies: (i) Generic Ontology for IoT Platforms (*GOIoTP*)[11], and its extension (ii) *GOIoTPex*[12].The *GOIoTP* is a modular core ontology for general use in IoT projects. It offers an expanded perspective on IoT, including both top-level concepts related to platforms and users, as well as "devices" that the ecosystem consists of. Therefore, *GOIoTP*, provides common and consistent shared semantics (see, Sections 1.1 and 2), as well as typical core IoT ontology (Section 2.1). *GOIoTP* was built around SOSA/SSN and offers seven horizontal modules (with additional provenance extension points), each focused on a distinct area of an IoT ecosystem (see, Figure 3). Other notable imports include GeoSPARQL and NASA SWEET units ontologies.

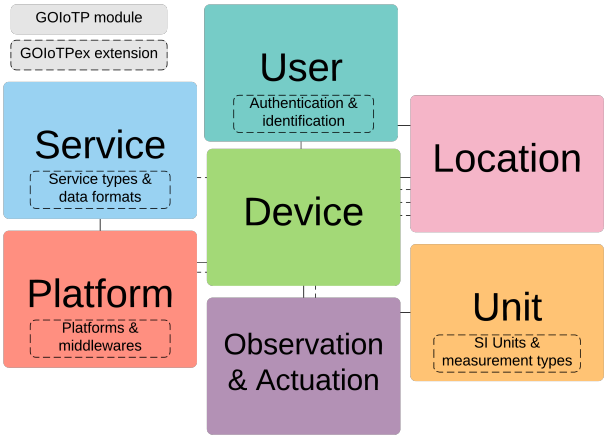

Fig. 3 GOIoTP ontology – main modules

Along a rich *Device* module (as in other IoT ontologies), *GOIoTP*, following SOSA/SSN, proposes a separate module for observations and actuations. This decoupling allows better separation of physical or virtual devices, described by selected properties, and device operation (including obtained or produced data).

Module dedicated to *Platforms* goes beyond the usual scope of platform-specific IoT ontologies, and treats a platform as an entity in an ecosystem, in which it needs to

[11] `http://inter-iot.eu/GOIoTP#`

[12] `http://inter-iot.eu/GOIoTPex#`

cooperate with other, equally important, platforms. This is critical for interoperability scenarios, in which a platform can join or leave a federation/group, delegate the management of some of its roaming devices to other platforms, while no artifact is in a privileged position. Note that the importance of this approach will increase as new IoT ecosystems will federate those already deployed using existing platforms.

The *User* and *Service* modules are largely independent from others. Whereas, usually in IoT ontologies, a service is attached to a platform, *GOIoTP* opts for decoupling them. This allows treatment of services as separate entities, that may, or may not be offered by some platform, or device. Similarly, a user in *GOIoTP* is a separate "sovereign" that may be dynamically associated with a service, device, platform, location, etc. Specifically, user is treated as a physical (e.g. a human) or virtual agent that may have some presence and history with an IoT artifact (e.g. an account and a set of roles registered with a platform). The user description is independent from technical details or requirements of any platform, device, or service.

In addition to generic descriptions of physical locations and their interconnections available in the *Location* module, the geographic information can be annotated with GeoSPARQL descriptions. In this way *GOIoTP* supports detailed descriptions of areas, and geographical functions, such as area intersections, collisions, point inclusion, distance calculation, etc.

In accordance with the ontology engineering best practices, *GOIoTP* has large number of generic descriptions, also called "stubs" that are lightweight, and are designed to be extended with more concrete definitions.

GOIoTPex is an official extension of *GOIoTP* that proposes a number of realizations and concretizations for top-level abstractions, defined in modules vertical with respect to *GOIoTP*, and horizontal to each other. For instance, where *GOIoTP*, across 2 modules, defines general framework for units of measurement, and connects it to observations and actuations, *GOIoTPex* complements it with instances and classes related to SI units. By the virtue of vertical modularization, this has no bearing on *GOIoTP*, which remains compatible with the imperial unit system. In this way, *GOIoTPex* brings semantics closer to concrete implementations of individual IoT systems, while leaving the option to fall-back to the more general *GOIoTP*, and preserving the vertical and horizontal modularity. This is important both for semantic interoperability, and for practicality of use. Both ontologies were successfully used in INTER-IoT deployments as Central Ontologies (see, Section 6).

For more information about history and design of *GOIoTP*, please refer to INTER-IoT deliverables (specifically D4.2 and D4.1). Ontology files, detailed axiom descriptions, outline of modules, updates and more can be found online[13].

[13] `https://inter-iot.github.io/ontology/`

4 From alignments to translation

Since the conceptualization of a domain directly corresponds to its *ontology* (see, [40]), to bridge the "semantic gap" between the artifacts, we have decided to use *alignments* between the corresponding ontologies. An *ontology alignment* [30] is a set of correspondences between two or more ontologies. These correspondences may be simple (between atomic entities) or complex (between groups of entities and sub-structures), but always relate entities from different ontologies and express their *similarity*. Here, *semantic translation* is a process of changing the underlying semantics of a piece of knowledge. Given information described semantically, in terms of a source ontology, it is transformed into information interpretable (understandable) in the scope of target semantics.

We assume that new instances of data may be generated dynamically within an ecosystem. Therefore, we aim to utilize alignments between structural information, rather than individuals, and assume that all exchanged messages are ontologically demarcated and schema-compliant. In the translation process, appropriate alignments are "applied" to messages traveling through *communication channels*. Specifically, application of an alignment substitutes parts of information from incoming messages with their translations (parts mapped by the alignment). Such, updated/translated, messages are available to recipient(s) "at the end of the channel". Here, let us assume that two (or more) platforms operate in the same domain, e.g. logistics, but use different semantics. For instance, one platform uses term *truck* with an attribute *capacity*, while the other uses term *lorry* with an attribute *volume*. Obviously, these terms have the same meaning and can be treated as equivalent. This exemplifies the simplest case of an alignment – equivalence between two atomic terms. Observe also that capacity and volume may be represented in different units, e.g. one uses the metric system (cubic meters), while the other comes from US and uses cubic inches, which introduces a complication into the, otherwise simple, example.

The simplest way in which ontology alignment can be found/defined is to print ontologies, place them next to each other and establish how their concepts relate. However, potential size and complexity of ontologies immediately undermines this approach. In [38], we have summarized the state-of-the-art in the area of tools for ontology alignment/merging/translating, etc. We have identified several tools that can support semantic engineers in preparing alignments. The most interesting among them were LogMap [41][14], COMA [27][15] and Agreement Maker [31][16].

Let us now consider the translation process. In principle, a *semantic-enabled* communication channel is a medium that, when properly configured, can accept messages with data annotated by entities from one (input/source) ontology and produce semantically equivalent messages annotated with another (output/target) ontology. In this way, the translation is entirely *external* to the participating artifacts. This simple idea (see, Figure 4) turns out to be fairly complex to realize in

[14] `https://www.cs.ox.ac.uk/isg/tools/LogMap/`

[15] `http://dbs.uni-leipzig.de/Research/coma.html`

[16] `http://somer.fc.ul.pt/aml.php`

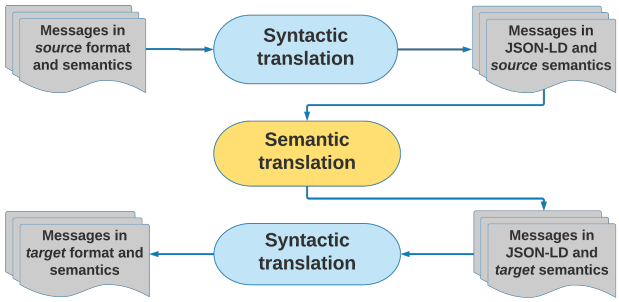

Fig. 4 INTER-IoT translation

practice, and requires harmonization of both syntax and semantics of information, besides actually using the communication infrastructure. Note that the actual number and topology of communication channels depends on the information flow in the ecosystem, and can change dynamically over time (in response to the needs of the applications using them). Additional challenge is to ensure that the communication architecture is capable of handling the volume of messages that can be exchanged between IoT artifacts. We shall come back to this problem in Section 6, but first, we need to describe a format that INTER-IoT solution uses for expressing alignments.

5 Alignment format

As a result of an in-depth analysis, we have decided to use the *Alignment Format* [7] as the basis for alignment representation. We have inspected several other methods of representing alignments, including EDOAL [5], but none of them turned out to be fully satisfactory for our purposes. The Alignment Format is a language independent format dedicated to persisting alignments in a simple and readable way. It was designed with the goal to provide common output format for matching tools. However, we have found that the Alignment Format itself, while working very well in ontology matching competitions, is not going to work well for streaming semantic translations. Therefore, taking the Alignment Format as the base, we have developed a dedicated format, with an RDF [19] representation (in RDF/XML serialization). The *IPSM Alignment Format* (IPSM-AF, in short) is used in the Inter-Platform Semantic Mediator (IPSM) software component (see Section 6). The IPSM-AF allows us to express mappings between contents of any valid RDF graphs. At the same time, alignments persisted in the original Alignment API format can be easily translated to the IPSM-AF, because of similarity in their structures.

Listing 1 shows structure of IPSM-AF expressed in RDF/XML. The format allows persisting uni-directional correspondences between ontologies. Mappings are split into separate alignment *cells*. Each cell is a correspondence between two *entities*

(or compound entity description). Entity can be a class, instance, object or datatype property. Namespace prefixes used in the examples are expanded in Table 1.

Suggested prefix	Namespace
sripas	http://www.inter-iot.eu/sripas\#
var	http://www.inter-iot.eu/sripas:node_
pred	http://www.inter-iot.eu/sripas:pred_
align	http://knowledgeweb.semanticweb.org/heterogeneity/alignment\#
dcelem	http://purl.org/dc/elements/1.1/
exmo	http://exmo.inrialpes.fr/align/ext/1.0/\#

Table 1 Namespaces in IPSM-AF

Elements from the *align* namespace are inherited from the Alignment Format. Table 2 lists the elements elements that are used to persist basic metadata about the alignment. The *sripas:cellFormat* property allows to specify data format used in alignment cells. Currently, RDF/XML and Turtle[17] are supported.

Element/Attribute	Meaning
dcelem:title	Name of the alignment
exmo:version	Version of the alignment e.g. 1.0
dcelem:creator	Author of the alignment
dcelem:description	Comment on what is the scope/aim of the alignment
align:xml	Derived from Alignment API. Default value: yes
align:level	Derived from Alignment API. Default value: 2IPSM
align:time	Derived from Alignment API
align:method	Derived from Alignment API. Default value: manual
align:type	Derived from Alignment API. Default value: **

Table 2 Metadata elements in IPSM-AF

```
<?xml version='1.0' encoding='utf-8' standalone='no'?>
<rdf:RDF xmlns="http://www.inter-iot.eu/sripas#" % other xml namespaces% >
<align:Alignment>
   <dcelem:title> % alignment title % </dcelem:title>
   <exmo:version> % alignment version % </exmo:version>
   <dcelem:creator> % alignment creator % </dcelem:creator>
   <dcelem:description> % alignemnt description % </dcelem:description>
   <align:xml>yes</align:xml>
   <align:level>2IPSM</align:level>
   <align:type>**</align:type>
   <align:method> % method % </align:method>
   <align:time> % time % </align:time>
   <sripas:cellFormat>
      <iiot:DataFormat rdf:about="http://inter-iot.eu/sripas#rdfxml"/>
   </sripas:cellFormat>
   <align:onto1>
      <align:Ontology rdf:about="% source ontology URI %">
         <align:location> % source ontology location % </align:location>
         <align:formalism>
```

[17] https://www.w3.org/TR/turtle/

```
            <align:Formalism
               align:name="% source ontology formalism name %"
               align:uri="% source ontology formalism URI %"/>
         </align:formalism>
      </align:Ontology>
   </align:onto1>
   <align:onto2> % target ontology information % </align:onto2>
   <sripas:steps rdf:parseType="Literal">
      <sripas:step sripas:order="% cell order %" sripas:cell="% cell id %"/>
      % more steps %
   </sripas:steps>
   <align:map> % alignment cell 1 % </align:map>
   % ... %
   <align:map> % alignment cell N % </align:map>
</align:Alignment>
</rdf:RDF>
```

Listing 1 IPSM-AF structure.

In principle, an IPSM-AF file (Listing 1) describes a uni-directional alignment comprised of independent mapping cells, each having "input" and "output" entity descriptions. Elements *onto1* and *onto2* describe the "source" and "target" ontologies of the alignment, by giving their URIs and specifying formalism used for their definition (e.g., OWL 2.0). Here, let us note that when bi-directional translations are needed, separate alignments have to be defined (even if the alignment cells all describe equivalences, and are, therefore, trivially reversible). This is because, in the IPSM-AF, source and target ontologies are explicitly specified (information used for communication channel configuration/creation process, within the IPSM).

The *steps* element specifies the (default) *order*, in which *cells* of the alignment will be subsequently *applied* in the message transformation process. Each *step* refers to a cell identifier as given by the *id* attribute of the *Cell* element. Note that a given cell might be referenced here (hence also applied) more than once. The default order may also be overridden during the channel configuration process.

Every *Cell* element represents an "atomic" RDF graph transformation. Here, content of *entity1* describes the *source*, and content of *entity2* establishes the *target* of the transformation. Both should be valid RDF graphs, possibly containing special-purpose nodes, playing the role of "variables", which are to be bound and referenced within the transformation.

Listing 2 shows an example (unidirectional) alignment between a drone and an ontology used by *dTraffic*. In the sample scenario data coming from drones is integrated and processed by the platform. We also assume that different types of drones use different communication standards. Here, original data is expressed in SAREF ontology. The target ontology is based on SOSA/SSN with extensions. This alignment includes two cells – one defining rules for translation of traffic message from drone to platform, and the other for translation of battery level and location message.

```
<align:Alignment>

   % ... alignment metadata ... %

   <sripas:steps rdf:parseType="Collection">
      <sripas:step sripas:order="1" sripas:cell="http://www.inter-iot.eu/sripas#1_traffic"/>
      <sripas:step sripas:order="2" sripas:cell="http://www.inter-iot.eu/sripas#2_status"/>
```

```
</sripas:steps>

<align:map>
    <align:Cell rdf:about="http://www.inter-iot.eu/sripas#1_traffic">
        <align:entity1 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
            var:Device a saref:Device ;
            saref:makesMeasurement var:Meas, var:Pos1, var:Pos2 .

            var:Meas a saref:Measurement ;
            saref:hasValue var:Val ;
            saref:hasTimestamp var:Tsp ;
            saref:isMeasuredIn sarefInst:Frequency ;
            saref:relatesToProperty sarefInst:Traffic .

            var:Pos1 a saref:Measurement ;
            saref:hasValue var:StartPos ;
            saref:relatesToProperty sarefInst:StartPosition .

            var:Pos2 a saref:Measurement ;
            saref:hasValue var:EndPos ;
            saref:relatesToProperty sarefInst:EndPosition .
        </align:entity1>
        <align:entity2 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
            var:Device a iiot:IoTDevice, sosa:Sensor ;
            sosa:madeObservation var:Meas, var:Pos1, var:Pos2 .

            var:Meas a sosa:Observation ;
            sosa:hasResult [ a sosa:Result ;
            iiot:hasResultValue var:Val ] ;
            sosa:phenomenonTime [ a time:Instant ; time:inXSDDateTime var:Tsp ] ;
            sosa:observedProperty iiotex:Traffic .

            var:Pos1 a sosa:Observation ;
            sosa:hasResult [ a sosa:Result, geosparql:Geometry ;
            iiot:hasResultValue var:GeoStart ] ;
            sosa:observedProperty iiotex:StartPosition .

            var:Pos2 a sosa:Observation ;
            sosa:hasResult [ a sosa:Result, geosparql:Geometry ;
            iiot:hasResultValue var:GeoEnd ] ;
            sosa:observedProperty iiotex:EndPosition .
        </align:entity2>
        <align:relation>=</align:relation>
        <sripas:transformation rdf:parseType="Literal">
            ...
        </sripas:transformation>
        <sripas:filters rdf:parseType="Literal">
            ...
        </sripas:filters>
        <sripas:typings rdf:parseType="Literal">
            ...
        </sripas:typings>
    </align:Cell>
</align:map>

<align:map>
    <align:Cell rdf:about="http://www.inter-iot.eu/sripas#2_status">
        <align:entity1 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
            var:Device a saref:Device ;
            saref:makesMeasurement var:Meas, var:Pos .

            var:Meas a saref:Measurement ;
            saref:hasValue var:Val ;
            saref:hasTimestamp var:Tsp ;
            saref:isMeasuredIn sarefInst:Percentage ;
            saref:relatesToProperty sarefInst:BatteryLevel .
```

```
            var:Pos a saref:Measurement ;
            saref:hasValue var:Position ;
            saref:relatesToProperty sarefInst:Position .
        </align:entity1>
        <align:entity2 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
            var:Device a iiot:IoTDevice, sosa:Sensor ;
            sosa:madeObservation var:Meas, var:Pos .

            var:Meas a sosa:Observation ;
            sosa:hasResult [ a sosa:Result, geosparql:Geometry ;
            iiot:hasResultValue var:Geo ] ;
            sosa:phenomenonTime [ a time:Instant ; time:inXSDDateTime var:Tsp ] ;
            sosa:observedProperty iiotex:BatteryLevel .

            var:Pos a sosa:Observation ;
            sosa:hasResult [ a sosa:Result ;
            iiot:hasResultValue var:Geo ] ;
            sosa:observedProperty iiotex:Position .
        </align:entity2>
        <align:relation>=</align:relation>
        <sripas:transformation rdf:parseType="Literal">
            <sripas:function about="str">
                <param order="1" about="&var;Position"/>
                <return about="&var;sGeo"/>
            </sripas:function>
            <sripas:function about="replace">
                <param order="1" about="&var;sGeo"/>
                <param order="2" val="^(\\d+\\.\\d+)\\s+\\d+\\.\\d+$"/>
                <param order="3" val="$1"/>
                <param order="4" val="i"/>
                <return about="&var;Lat"/>
            </sripas:function>
            <sripas:function about="replace">
                <param order="1" about="&var;sGeo"/>
                <param order="2" val="^\\d+\\.\\d+\\s+(\\d+\\.\\d+)$"/>
                <param order="3" val="$1"/>
                <return about="&var;Long"/>
            </sripas:function>
            <sripas:function about="concat">
                <param order="1" val="Point("/>
                <param order="2" about="&var;Lat"/>
                <param order="3" val=" "/>
                <param order="4" about="&var;Long"/>
                <param order="5" val=")"/>
                <return about="&var;Geo"/>
            </sripas:function>
        </sripas:transformation>
        <sripas:filters rdf:parseType="Literal">
            <sripas:filter about="&var;sGeo" datatype="&xsd;string"/>
            <sripas:filter about="&var;Lat" datatype="&xsd;float"/>
            <sripas:filter about="&var;Long" datatype="&xsd;float"/>
        </sripas:filters>
        <sripas:typings rdf:parseType="Literal">
            <sripas:typing about="&var;Geo"
               datatype="http://www.opengis.net/def/sf/wktLiteral"/>
        </sripas:typings>
      </align:Cell>
   </align:map>
</align:Alignment>
```

Listing 2 IPSM-AF alignment between drone and dTraffic.

6 IPSM semantic translation tool

IPSM (Inter-Platform Semantic Mediator) is a generic streaming semantic translation software developed by INTER-IoT. It realizes the idea of translation through alignments (see, Section 4). IPSM offers highly-scalable, efficient translation architecture applicable in any semantic translation scenario, and configurable with IPSM-AF files, regardless of domain of application.

There are two main interfaces to access IPSM: REST and reactive streaming. The first option offers quick, one-message-at-a-time, service, while the latter is better suited for large asynchronous streams of data. In both cases, translation is done transactionally "per message" by applying rules defined in alignment files. Technically, there is no limit on contents or size of messages, so the software could be used to perform a one-time batch translation of a whole database worth of data. Nevertheless, the design of IPSM makes it better suited for scenarios, where data that needs translation is not known beforehand, and thus cannot be translated in a single operation. This approach lends itself very well to communication between diverse IoT artifacts.

Reactive streaming, in IPSM, rests on semantic translation channels (*channels*, for short), and a *central ontology* (*CO*). A channel is a one way stream that accepts messages described with a given ontology and translates them to a different ontology. Just like in any reactive streaming implementation, inputs and outputs of channels may be read from, written to, connected to other stream processors, and even to each other. In IPSM, translation is performed "on the fly", as messages pass the channel, and is configured by two one-way alignments per channel. The 2 alignment requirement is dictated by the central ontology architecture (described further on). Consequently, to set up a two-way communication one needs to configure 2 channels and 4 one-way alignments (2 per channel).

CO is a shared vocabulary, founded on the concept of core ontology (see, Sections 1.1 and 2) that is going to be used in the process of translation, but does not need to be implemented natively by the communicating artifacts. CO is a special, modular, ontology used as an *intermediary* in translation (see, Figure 5).

Assuming that IoT artifacts A, B that want to communicate, and with the central ontology Ω in place (see, Figure 5), an IPSM channel should be configured with one alignment that defines translation from A semantics into Ω (denoted $A \triangleright \Omega$), and another one from Ω into B (alignment $\Omega \triangleright B$). Messages flowing through are first translated using $A \triangleright \Omega$, and then $\Omega \triangleright B$. In order to set up a different channel, say from A to C, the same alignment $A \triangleright \Omega$ may be used as the "input" configuration, with a new alignment $\Omega \triangleright C$ for the "output". In this way, from the perspective of A, it is enough to learn Ω and provide alignments to and from its own semantics ($A \triangleright \Omega$ and $\Omega \triangleright A$). Once A makes such alignments available publicly by uploading them to an instance of IPSM (i.e. its alignment repository component), anyone may use them to configure communication to and from their own artifact. Technically, IPSM has a built-in "identity" alignment that effectively performs no translation, and is used to relax the two-alignment requirement, in case an artifact natively supports Ω. Using CO as an intermediate step, IPSM facilitates translation by mediating it through CO,

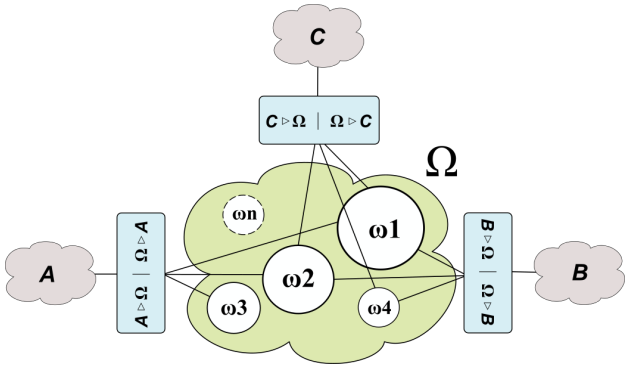

Fig. 5 IPSM configuration with modularized central ontology Ω.

as opposed to pipelining it directly (one-to-one). Therefore, the cost of joining a deployed IPSM ecosystem, regardless of its size, is always just 2 alignments.

IPSM places no technical requirements on the CO (any ontology can be used), but there are features that make some better suited for a CO than others. First of all, a good CO should have a broad range, and cover, in detail, its domain. Because artifacts translate to and from CO, a broad and detailed (highly granular) semantics will allow for translation of full range of messages from all artifacts without loss of specificity. Second, (well documented) modularity of CO lowers the cost of authoring alignments, because an engineer only needs to focus on the modules that are relevant to their artifact, and can disregard others. For instance, if only two artifacts in an ecosystem use descriptions of sea-faring vehicles, the sea module of a CO will not be of interest to any other participant, because their artifacts simply do not support it, i.e. they do not "talk" about sea vehicles, regardless of translation. Overall, while the CO has to cover topics that the participants "want to communicate about" it does not have to cover the whole range of semantics that participating artifacts do.

By design, all alignments in IPSM are "public" (i.e. the alignment repository in an IPSM instance is available to anyone that has the rights to configure channels, in the same instance), so the division of Ω into modules $\omega 1$, $\omega 2$, ... has bearing on security. It is up to the artifact owner to decide what messages to send (and when), what the scope of its own schema that the artifact will use to communicate with others is, and what CO modules will the alignment apply. Technically, IPSM-AF alignments can be used to redact, obfuscate, and anonymize data , but because IPSM instances are de facto external to the data origin, IPSM should not be used as a data security measure, unless run in a closed, trusted network.

More information about IPSM, its architecture, performance, use-case scenarios, and deployment methods can be found in [32, 33, 34], as well as in the INTER-IoT online documentation[18] [19].

7 Use case processing

Thus far we have considered and discussed different facets of our use-case scenario (see, Section 1.2 and Figure 1). The scenario involves actors with various communication needs that can be realized with deployment of a number of IPSM instances.

Let us recall that the *dTraffic* actor needs interoperability in communication of drones with a central "base of operations". Here, we assume that *dTraffic* cooperates with several types of drones that use different data models to represent position, battery level, and traffic congestion level. They can be based on one of the core ontologies extended with domain ontologies e.g. drone type 1 uses SOSA/SSN for drone description and W3C Basic Geo Vocabulary for geopositioning, whereas drone type 2 uses SAREF with GeoSPARQL. Additionally, drones may come from producers using their native data models (not based on any standard). In each case, congestion level is usually represented in a drone-specific way, since there are no publicly available standards. Assuming impossibility of implementing the same semantics across artifacts (now and in the future), the two main approaches to establish interoperability with different drones are to implement: (i) a bi-directional (one-to-one) translation mechanism between *dTraffic* and each drone type, (ii) mediator with a CO, to which messages exchanged in both directions are translated (see, Section 6). The former approach is considerably more difficult and complex to realize in a dynamic ecosystem with multiple IoT artifacts. Moreover, it implies system modification, every time a new type of drone is added (defeating the 24/7 availability). In the latter approach, preexisting artifacts are not affected by the integration. Addition of new device type(s) requires translation to and/or from the CO. Therefore, it is the mediator-based approach that was selected in INTER-IoT to provide interoperability on data and semantics layer, and is recommended in the drones use case.

Similarly, *iotDelivery* needs to exchange data with trucks. This also requires choice of data models. Since there are several publicly available standards in the transportation and logistics domain, we may assume that e.g. truck company 1 uses LogiCO and LogiServ ontologies, whereas truck company 2 might use OTN.

Let us now consider how semantic translation in our use case can be organized. In the presented scenario, we may consider two IPSM instances that serve as communication hubs: *dTraffic Hub* and *iotDelivery Hub*. The former translates messages exchanged between different drone types and *dTraffic* (which can be

[18] `https://inter-iot.readthedocs.io/projects/ipsm/en/latest/`

[19] `https://inter-iot-cookbook.readthedocs.io/en/latest/inter-layer/ds2ds/appendices/products/`

assumed to work with the *dTraffic Hub* data model directly). The latter enables exchange of information between trucks, *iParking* companies, and *dTraffic*, in order to schedule trucks and reserve parking slots for them.

Next, a set of alignments needs to be prepared: (i) bidirectional, between data models of drones of different types and the *dTraffic Hub* central data model, (ii) bidirectional, between data models of *iParking* companies and the *iotDelivery Hub* central data model, (iii) bidirectional, between data models of truck companies and the *iotDelivery Hub*, (iv) unidirectional, between *dTraffic* central data model and the *iotDelivery Hub* central data model, and (v) bidirectional, between data models of *iotDelivery* companies and *iotDelivery Hub* central data model. To illustrate how the proposed approach is going to work, let consider a sample message send by the drone, that should be consumed by *dTraffic*.


```
{
  "@graph" : [ {
    "@graph" : [ {
      "@id" : "InterIoTMsg:meta308c3987-b69e-4672-890b-3f3d6229596d",
      "@type" : [ "InterIoTMsg:meta", "InterIoTMsg:Thing_Update" ],
      "InterIoTMsg:conversationID" : "conv85e0f5d2-cf65-4d23-84b1-ff1381ae01fc",
      "InterIoTMsg:dateTimeStamp" : "2019-02-08T13:48:19.428+00:00",
      "InterIoTMsg:messageID" : "msg204d0405-a6da-4054-a6db-96d20c413746"
    } ],
    "@id" : "InterIoTMsg:metadata"
  }, {
    "@graph" : [
      {
        "@id": "http://www.drone1.eu/devices/Device_1",
        "@type": "saref:Device",
        "saref:hasState": {
          "@id": "saref:Start"
        },
        "saref:makesMeasurement": [
          { "@id": "_:Pos" },
          { "@id": "_:Meas" }
        ]
      },
      {
        "@id": "_:Pos",
        "@type": "saref:Measurement",
        "saref:hasValue": "45.256 -71.92",
        "saref:relatesToProperty": {
          "@id": "sarefInst:Position"
        }
      },
      {
        "@id": "_:Meas",
        "@type": "saref:Measurement",
        "saref:hasTimestamp": {
          "@type" : "http://www.w3.org/2001/XMLSchema#dateTime",
          "@value" : "2019-02-08T13:48:18"
        },
        "saref:hasValue": {
          "@type" : "http://www.w3.org/2001/XMLSchema#float",
          "@value" : "0.75"
        },
        "saref:isMeasuredIn": {
          "@id": "sarefInst:Percentage"
        },
        "saref:relatesToProperty": {
          "@id": "sarefInst:BatteryLevel"
        }
      }
```

```
    ],
    "@id" : "InterIoTMsg:payload"
  } ],
  "@context" : {
    "InterIoTMsg" : "http://inter-iot.eu/message/",
    "InterIoTInst" : "http://inter-iot.eu/instance/",
    "owl" : "http://www.w3.org/2002/07/owl#",
    "rdf" : "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "xsd" : "http://www.w3.org/2001/XMLSchema#",
    "rdfs" : "http://www.w3.org/2000/01/rdf-schema#",
    "InterIoT" : "http://inter-iot.eu/",
    "sosa" : "http://www.w3.org/ns/sosa/",
    "saref" : "https://w3id.org/saref#",
    "sarefInst" : "https://w3id.org/saref/instances/"
  }
}
```


Listing 3 Status message from drone.

Listing 3 shows a status message sent periodically by a drone, stating its current position and battery level. Note that each drone type sends this kind of message utilizing its own data model. This message can be translated with alignment given in Listing 2. It matches pattern in *entity1* of cell 2_*status* that defines how it should be translated to the data model based on SOSA/SSN and GeoSPARQL (*dTraffic Hub* central data model is assumed to use these ontologies).

Listing 4 shows a sample message that can be sent from one of *iotDelivery* companies. Information is expressed in *iotDelivery1* native semantics and contains Car ID and time period for the reservation. Lets assume that reservation should be done by *iParking* company named *iParking1* that also uses its own native semantics. In such case, we need to use two alignments: one from the *iotDelivery1* to the *iotDelivery Hub* central data model (see, Listing 5), the other from the *iotDelivery Hub* to the *iParking1* data model (see, Listing 6).


```
{
  "@graph" : [ {
    "@graph" : [ {
      "@id" : "InterIoTMsg:meta308c3987-b69e-4672-890b-3f3d6229596d",
      "@type" : [ "InterIoTMsg:meta", "InterIoTMsg:Thing_Update" ],
      "InterIoTMsg:conversationID" : "conv85e0f5d2-cf65-4d23-84b1-ff1381ae01fc",
      "InterIoTMsg:dateTimeStamp" : "2019-02-08T13:48:19.428+00:00",
      "InterIoTMsg:messageID" : "msg204d0405-a6da-4054-a6db-96d20c413746"
    } ],
    "@id" : "InterIoTMsg:metadata"
  }, {
    "@graph" : [
      {
        "@id": "iotDeliveryInst:Reservation1",
        "@type": "iotDelivery:Reservation",
        "iotDelivery:hasCarId": "WK 12345",
        "iotDelivery:hasEndTime": {
          "@type" : "http://www.w3.org/2001/XMLSchema#dateTime",
          "@value" : "2019-02-08T13:50:00"
        },
        "iotDelivery:hasStartTime": {
          "@type" : "http://www.w3.org/2001/XMLSchema#dateTime",
          "@value" : "2019-02-08T14:00:00"
        }
      }
    ],
    "@id" : "InterIoTMsg:payload"
  } ],
  "@context" : {
```

```
    "InterIoTMsg" : "http://inter-iot.eu/message/",
    "InterIoTInst" : "http://inter-iot.eu/instance/",
    "owl" : "http://www.w3.org/2002/07/owl#",
    "rdf" : "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "xsd" : "http://www.w3.org/2001/XMLSchema#",
    "rdfs" : "http://www.w3.org/2000/01/rdf-schema#",
    "InterIoT" : "http://inter-iot.eu/",
    "iotDelivery": "https://iotDelivery1.org#",
    "iotDeliveryInst": "https://iotDelivery1.org/instances/"
  }
}
```

Listing 4 Reservation message from *iotDelivery1*.

Listing 5 shows alignment with one cell that matches message from Listing 4.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:sripas="http://www.inter-iot.eu/sripas#"
         xmlns="http://www.inter-iot.eu/sripas#"
         xmlns:align="http://knowledgeweb.semanticweb.org/heterogeneity/alignment#"
         xmlns:dcelem="http://purl.org/dc/elements/1.1/"
         xmlns:exmo="http://exmo.inrialpes.fr/align/ext/1.0/#"
         xmlns:var="http://www.inter-iot.eu/sripas#node_"
         xmlns:sosa="http://www.w3.org/ns/sosa/"
         xmlns:time="http://www.w3.org/2006/time#"
         xmlns:iiot="http://inter-iot.eu/GOIoTP#"
         xmlns:iiotex="http://inter-iot.eu/GOIoTPex#"
         xmlns:ssn="http://www.w3.org/ns/ssn/"
         xmlns:iotDelivery="https://iotDelivery1.org#"
         xmlns:iotDeliveryInst="https://iotDelivery1.org/instances/"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:logico="http://ontology.tno.nl/logico#"
>
   <align:Alignment>

      <dcelem:title>IoTDelivery1_IoTDeliveryCO</dcelem:title>
      <exmo:version>1.0</exmo:version>
      <dcelem:creator>SRIPAS</dcelem:creator>
      <dcelem:description>Between IoT-Delivery1 and IoT-Delivery Hub.</dcelem:description>

      <align:xml>yes</align:xml>
      <align:level>2IPSM</align:level>
      <align:type>**</align:type>
      <align:method>manual</align:method>
      <dcelem:date>13-02-2019</dcelem:date>
      <sripas:cellFormat>
          <iiot:DataFormat rdf:about="http://inter-iot.eu/sripas#turtle" />
      </sripas:cellFormat>

      <align:onto1>
          <align:Ontology rdf:about="https://iotDelivery1.org#">
              <align:formalism>
                  <align:Formalism
                   align:name="OWL2.0" align:uri="http://www.w3.org/2002/07/owl#" />
              </align:formalism>
          </align:Ontology>
      </align:onto1>
      <align:onto2>
          <align:Ontology rdf:about="http://inter-iot.eu/GOIoTPex#">
              <align:formalism>
                  <align:Formalism
                   align:name="OWL2.0" align:uri="http://www.w3.org/2002/07/owl#" />
              </align:formalism>
          </align:Ontology>
      </align:onto2>

      <sripas:steps rdf:parseType="Collection">
```

```
            <sripas:step sripas:order="1"
             sripas:cell="http://www.inter-iot.eu/sripas#1_reservation"/>
        </sripas:steps>

        <align:map>
            <align:Cell rdf:about="http://www.inter-iot.eu/sripas#1_reservation">
                <align:entity1 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                    var:R a iotDelivery:Reservation ;
                    iotDelivery:hasCarId var:CarId ;
                    iotDelivery:hasStartTime var:Time1 ;
                    iotDelivery:hasEndTime var:Time2 .
                </align:entity1>
                <align:entity2 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                    var:R a iiotex:Reservation ;
                    iiotex:hasIssuer [
                        a logico:Truck ;
                        logico:id [ logico:hasIdValue var:CarId ]
                    ] ;
                    iiotex:forRegion [
                        a logico:Region ;
                        logico:id [ logico:hasIdValue var:ParkingId ;
                            logico:hasAgency iiotex:ParkingRegistry
                        ] ;
                        iiotex:hasTimebox [
                            a time:Interval ;
                            time:hasBeginning [
                                a time:Instant; time:inXSDDateTimeStamp var:Time1
                            ] ;
                            time:hasEnd [
                                a time:Instant; time:inXSDDateTimeStamp var:Time2
                            ]
                        ]
                    ] .
                </align:entity2>
                <align:relation>=</align:relation>
            </align:Cell>
        </align:map>

    </align:Alignment>
</rdf:RDF>
```

Listing 5 Alignment between *iotDelivery1* and *iotDelivery Hub* central data model.

Listing 6 shows alignment with one cell that matches output message, after translation of message from Listing 4 with alignment from Listing 5 .

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:sripas="http://www.inter-iot.eu/sripas#"
         xmlns="http://www.inter-iot.eu/sripas#"
         xmlns:align="http://knowledgeweb.semanticweb.org/heterogeneity/alignment#"
         xmlns:dcelem="http://purl.org/dc/elements/1.1/"
         xmlns:exmo="http://exmo.inrialpes.fr/align/ext/1.0/#"
         xmlns:var="http://www.inter-iot.eu/sripas#node_"
         xmlns:sosa="http://www.w3.org/ns/sosa/"
         xmlns:time="http://www.w3.org/2006/time#"
         xmlns:iiot="http://inter-iot.eu/GOIoTP#"
         xmlns:iiotex="http://inter-iot.eu/GOIoTPex#"
         xmlns:ssn="http://www.w3.org/ns/ssn/"
         xmlns:iParking="https://iParking1.org#"
         xmlns:iParkingInst="https://iParking1.org/instances/"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:logico="http://ontology.tno.nl/logico#"
>
    <align:Alignment>

        <dcelem:title>IoTDeliveryCO_iParking1</dcelem:title>
```

```
<exmo:version>1.0</exmo:version>
<dcelem:creator>SRIPAS</dcelem:creator>
<dcelem:description>Between IoT-Delivery Hub and iParking1.</dcelem:description>

<align:xml>yes</align:xml>
<align:level>2IPSM</align:level>
<align:type>**</align:type>
<align:method>manual</align:method>
<dcelem:date>13-02-2019</dcelem:date>
<sripas:cellFormat>
    <iiot:DataFormat rdf:about="http://inter-iot.eu/sripas#turtle" />
</sripas:cellFormat>

<align:onto1>
    <align:Ontology rdf:about="http://inter-iot.eu/GOIoTPex#">
        <align:formalism>
            <align:Formalism align:name="OWL2.0"
             align:uri="http://www.w3.org/2002/07/owl#" />
        </align:formalism>
    </align:Ontology>
</align:onto1>
<align:onto2>
    <align:Ontology rdf:about="https://iParking1.org#">
        <align:formalism>
            <align:Formalism align:name="OWL2.0"
             align:uri="http://www.w3.org/2002/07/owl#" />
        </align:formalism>
    </align:Ontology>
</align:onto2>

<sripas:steps rdf:parseType="Collection">
    <sripas:step sripas:order="1"
     sripas:cell="http://www.inter-iot.eu/sripas#1_reservation"/>
</sripas:steps>

<align:map>
    <align:Cell rdf:about="http://www.inter-iot.eu/sripas#1_reservation">
        <align:entity1 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
            var:R a iiotex:Reservation ;
            iiotex:hasIssuer [
                a logico:Truck ;
                logico:id [ logico:hasIdValue var:CarId ]
            ] ;
            iiotex:forRegion [
                a logico:Region ;
                logico:id [
                    logico:hasIdValue var:ParkingId ;
                    logico:hasAgency iiotex:ParkingRegistry
                ] ;
                iiotex:hasTimebox [
                    a time:Interval ;
                    time:hasBeginning [
                        a time:Instant; time:inXSDDateTimeStamp var:Time1
                    ] ;
                    time:hasEnd [
                        a time:Instant; time:inXSDDateTimeStamp var:Time2
                    ]
                ]
            ] .
        </align:entity1>
        <align:entity2 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
            var:R a iParking:Reservation ;
            iParking:hasCarId var:CarId ;
            iParking:hasStartTime var:Time1 ;
            iParking:hasEndTime var:Time2 .
        </align:entity2>
        <align:relation>=</align:relation>
```

```
            </align:Cell>
        </align:map>

    </align:Alignment>
</rdf:RDF>
```

Listing 6 Alignment between *iotDelivery Hub* central data model and *iParking1*.

8 Concluding remarks

In this chapter we summarized the INTER-IoT approach to semantic interoperability. The simple *dTraffic* mock scenario was deconstructed throughout the text to expose the affluent semantic methods, techniques, and advances weaved into every aspect of design and implementation of semantically interoperable IoT systems that use INTER-IoT. Arisen from the study of ontology alignments, the INTER-IoT pathway is lined with elucidation of semantics into a very explicit and concrete form. Such concretization empowered the novel IPSM-AF format, and paved way to the practical application of ontology alignments as translation rules for real-time dynamic transformation of data. Driven by the needs and wants commonly emerging in the IoT domain, the design of a flexible stream-driven architecture for efficient, scalable, and reactive semantic translation of messages was crowned with the implementation of IPSM software. Additionally, introduction of the idea of a *central ontology* capacitated multi-deployments of IPSM among considerable amount of communicating artifacts with minimal cost of changes in the ecosystem, even during uninterrupted operation.

The generic nature of proposed solutions makes the INTER-IoT approach to semantics feasible for any domain that requires data semantics interoperability. Additionally we developed two modular and extendable ontologies: *GOIoTP* and *GOIoTPex*, ready to be used as central ontologies for any IoT-related sub-domain. [50]

Fruitful usage of the INTER-IoT approach to semantics, including the theory, designs, and software, in INTER-IoT pilot implementations, as well as other IoT projects (e.g. EU ACTIVAGE, EU Pixel) is a testimony to its potential and practicality.

References

1. Basic Geo (WGS84 lat/long) vocabulary. https://www.w3.org/2003/01/geo/.
2. Bioportal ontologies. https://bioportal.bioontology.org/ontologies.
3. Climate and forecast features. https://www.w3.org/2005/Incubator/ssn/ssnx/cf/cf-feature.
4. DATEX II. http://www.datex2.eu/.
5. EDOAL: Expressive and declarative ontology alignment language. http://alignapi.gforge.inria.fr/edoal.html.
6. FHIR OWL Ontology. w3c.github.io/hcls-fhir-rdf/spec/ontology.html.

7. A format for ontology alignment. http://alignapi.gforge.inria.fr/format.html.
8. Geographic data format. http://www.iso.org/iso/iso_catalogue/catalogue_tc /catalogue_detail.htm?csnumber=54610.
9. geoSPARQL. https://www.opengeospatial.org/standards/geosparql.
10. The linked earth ontology. http://linked.earth/ontology/#.
11. Logical – transnational logistics improvement through cloud computing and innovaative cooperative business modelas. http://www.project-logical.eu/.
12. Logical cloud portal. https://logicalcloud.wifa.uni-leipzig.de/portal.
13. Logico Ontology. ontology.tno.nl/logico/.
14. Logiserv Ontology. ontology.tno.nl/logiserv/.
15. Logistic grid ontology. http://www.interloggrid.org/LogisticsOntology.owl.
16. The obo foundry. http://www.obofoundry.org.
17. Ontology of transportation networks. http://www.pms.ifi.lmu.de/rewerse-wga1/otn/OTN.owl.
18. PROV-O: The PROV Ontology. https://www.w3.org/TR/prov-o/.
19. Resource description framework (RDF). https://www.w3.org/RDF/.
20. Rewerse: Reasoning on the web. http://rewerse.net/.
21. Seas-weatherontology ontology. https://ci.mines-stetienne.fr/seas/WeatherOntology.
22. The transport disruption ontology. https://transportdisruption.github.io/transportdisruption.html.
23. UniversAAL Ontologies. https://github.com/universAAL/ontology.
24. Achieving technical interoperability – the ETSI approach. ETSI White Paper No. 3, April 2008.
25. Semantic Sensor Network XG final report, 2011.
26. Sören Auer, Jens Lehmann, and Sebastian Hellmann. LinkedGeoData: Adding a spatial dimension to the web of data. In Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, editors, The Semantic Web – ISWC 2009, pages 731–746, Berlin, Heidelberg, 2009. Springer.
27. David Aumueller, Hong-Hai Do, Sabine Massmann, and Erhard Rahm. Schema and ontology matching with COMA++. In Proc. of the 2005 ACM SIGMOD Int. Conf. on Management of Data, pages 906–908. ACM, 2005.
28. Maria Bermudez-Edo, Tarek Elsaleh, Payam Barnaghi, and Kerry Taylor. IoT-Lite: a lightweight semantic model for the internet of things and its use with dynamic semantics. Personal and Ubiquitous Computing, 21(3):475–487, Jun 2017.
29. Michael Compton, Payam Barnaghi, Luis Bermudez, Raul Garcia-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, Vincent Huang, Krzysztof Janowicz, W. David Kelsey, Danh Le Phuoc, Laurent Lefort, Myriam Leggieri, Holger Neuhaus, Andriy Nikolov, Kevin Page, Alexandre Passant, Amit Sheth, and Kerry Taylor. The SSN ontology of the W3C semantic sensor network incubator group. Web Semantics: Science, Services and Agents on the World Wide Web, 17:25–32, 2012.
30. Jérôme Euzenat and Pavel Shvaiko. Ontology Matching. Springer, 2 edition, 2013.
31. Daniel Faria, Catia Pesquita, Emanuel Santos, Matteo Palmonari, Isabel F Cruz, and Francisco M Couto. The AgreementMakerLight ontology matching system. In On the Move to Meaningful Internet Systems: OTM 2013 Conferences, pages 527–541. Springer, 2013.
32. Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Paweł Szmeja, and Katarzyna Wasielewska. Streaming semantic translations. In 21st International Conference on System Theory, Control and Computing ICSTCC, Proceedings, pages 1–8. Ieee, 2017.
33. M. Ganzha, M. Paprzycki, W. Pawłowski, P. Szmeja and K. Wasielewska, "Alignment-based semantic translation of geospatial data," 2017 3rd International Conference on Advances in Computing,Communication & Automation (ICACCA) (Fall), Dehradun, India, pp. 1-8, 2017.
34. Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Paweł Szmeja, Katarzyna Wasielewska, Bartłomiej Solarz-Niesłuchowski, and Jara Suárez de Puga García. Towards high throughput semantic translation. In Giancarlo Fortino, Carlos E. Palau, Antonio Guerrieri, Nora Cuppens, Frédéric Cuppens, Hakima Chaouchi, and Alban Gabillon, editors, Interoperability, Safety and Security in IoT, pages 67–74, Cham, 2018. Springer International Publishing.

35. Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Paweł Szmeja, and Katarzyna Wasielewska. Semantic technologies for the IoT – an Inter-IoT perspective. In 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI), pages 271–276, Berlin, Germany, April 2016. IEEE.
36. Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Paweł Szmeja, and Katarzyna Wasielewska. Towards common vocabulary for IoT ecosystems—preliminary considerations. In Intelligent Information and Database Systems, 9th Asian Conference, ACIIDS 2017, Kanazawa, Japan, April 3-5, 2017, Proceedings, Part I, volume 10191 of LNCS, pages 35–45. Springer, 2017.
37. Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Paweł Szmeja, and Katarzyna Wasielewska. Towards semantic interoperability between Internet of Things platforms. In Raffaele Gravina, Carlos E. Palau, Marco Manso, Antonio Liotta, and Giancarlo Fortino, editors, Integration, Interconnection, and Interoperability of IoT Systems, pages 103–127. Springer, 2017.
38. Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Paweł Szmeja, Katarzyna Wasielewska, and Giancarlo Fortino. Tools for ontology matching—practical considerations from INTER-IoT perspective. In Proc. of the 8th Int. Conference on Internet and Distributed Computing Systems, volume 9864 of LNCS, pages 296–307. Springer, 2016.
39. Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Paweł Szmeja, Katarzyna Wasielewska, and Carlos E. Palau. From implicit semantics towards ontologies—practical considerations from the INTER-IoT perspective (submitted for publication). In Proc. of 1st edition of Globe-IoT 2017: Towards Global Interoperability among IoT Systems, 2017.
40. Thomas R Gruber. Toward principles for the design of ontologies used for knowledge sharing? International journal of human-computer studies, 43(5):907–928, 1995.
41. Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. LogMap: Logic-based and scalable ontology matching. In International Semantic Web Conference (ISWC), volume 7031 of LNCS, pages 273–288. Springer, 2011.
42. Bernhard Lorenz, Hans Jürgen Ohlbach, and Laibing Yang. Ontology of transportation networks. 2005.
43. Claudine Métral, Roland Billen, Anne-Francoise Cutting-Decelle, and Muriel Van Ruymbeke. Ontology-based approaches for improving the interoperability between 3d urban models. Journal of Information Technology in Construction, 15, 2010.
44. Andreas Scheuermann and Julia Hoxha. Ontologies for intelligent provision of logistics services. In 7th International Conference on Internet and Web Applications and Services (ICIW 2012), Germany, May 2012. XPS.
45. Steffen Staab and Rudi Studer. Handbook on Ontologies. Springer-Verlag, 2 edition, 2009.
46. Stanisław Strzelczak. Core ontology for manufacturing and logistics. Zeszyty Naukowe. Organizacja i Zarządzanie/Politechnika Sląska, (73):603–618, 2014.
47. Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra, editors. Modular Ontologies. Concepts, Theories and Techniques for Knowledge Modularization, volume 5445 of State-of-the-Art Survey, LLNCS. 2009.
48. A.Belsa, D. Sarabia-Jacome, Carlos E. Palau and M.Esteve. "Flow-Based Programming Interoperability Solution for IoT Platform Applications", in 2018 IEEE International Conference on Cloud Engineering (IC2E),pp. 304-309, Orlando (FL), February 2018.
49. A. Broring, A. Zappa, O.Vermesan, K. Främling, A. Zaslavsky, R. Gonzalez-Usach, P. Szmeja, C. Palau, M. Jacoby, I. P. Zarko, S. Sour- sos, C. Schmitt, M. Plociennik, S. Krco, S. Georgoulas, I. Larizgoitia, N. Gligoric, R. García-Castro, F. Serena, V. Orav. Advancing IoT Platform Interoperability, River Publishers, 2018.
50. Giancarlo Fortino, Antonio Liotta, Carlos Palau, Raffaele Gravina and Marco Manso (editors). Integration, Interconnection, and Interoperability of IoT Systems, Springer, February 2017.
51. Salvatore F. Pileggi, Carlos E. Palau and Manuel Esteve. "Building Semantic Sensor Web: Knowledge and Interoperability", in Proceedings of the International Workshop on Semantic Sensor Web - Volume 1: SSW, (IC3K 2010),pp. 15.22, April 2010.
52. Ovidiu Vermesan and Peter Friess (editors). Digitising the Industry Internet of Things Connecting the Physical, Digital and Virtual Worlds, River Publishers, 2016.

53. Giancarlo Fortino, Alfredo Garro, Wilma Russo: Achieving Mobile Agent Systems interoperability through software layering. Inf. Softw. Technol. 50(4): 322-341 (2008)