IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 4, 2019, accepted November 25, 2019, date of publication December 5, 2019,
date of current version December 19, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2957855

Ontology-Based Consistent Specification of Sensor Data Acquisition Plans in Cross-Domain IoT Platforms

STEFANO VALTOLINA, LUCA FERRARI, AND MARCO MESITI
Department of Computer Science, University of Milano, 20133 Milan, Italy
Corresponding author: Stefano Valtolina (valtolin@di.unimi.it)

ABSTRACT Nowadays there is an high number of IoT applications that seldom can interact with each other because developed within different Vertical IoT Platforms that adopt different standards. Several efforts are devoted to the construction of cross-layered frameworks that facilitate the interoperability among cross-domain IoT platforms for the development of *horizontal* applications. Even if their realization poses different challenges across all layers of the network stack, in this paper we focus on the interoperability issues that arise at the data management layer. Specifically, starting from a flexible multi-granular Spatio-Temporal-Thematic data model according to which events generated by different kinds of sensors can be represented, we propose a Semantic Virtualization approach according to which the sensors belonging to different IoT platforms and the schema of the produced event streams are described in a *Domain Ontology*, obtained through the extension of the well-known Semantic Sensor Network ontology. Then, these sensors can be exploited for the creation of Data Acquisition Plans by means of which the streams of events can be filtered, merged, and aggregated in a meaningful way. A notion of consistency is introduced to bind the output streams of the services contained in the Data Acquisition Plan with the Domain Ontology in order to provide a semantic description of its final output. When these plans meet the consistency constraints, it means that the data they handle are well described at the Ontological level and thus the data acquisition process over passed the interoperability barriers occurring in the original sources. The facilities of the StreamLoader prototype are finally presented for supporting the user in the Semantic Virtualization process and for the construction of meaningful Data Acquisition Plans.

INDEX TERMS Interoperability in IoT domain, ETL operations, semantic consistency.

I. INTRODUCTION

According to Gartner Inc. [1] in 2020 more than 20 billion physical or social sensors will be in use worldwide which produce large, complex, heterogeneous, structured and unstructured data that can be profitably used for generating a wide plethora of services. Dealing with the vast amount of data produced by the things, their varying capabilities, and generating useful services, is around the biggest conceptual and technological challenges of our time [2]. This challenge is further exaggerated by the typical ills of early-stage technology that lead to the production of different IoT platforms (more than 600 according to a recent survey [3]) and standard formats (e.g. OMA NGSI 9/10 [4] and ETSI oneM2M [5]) in different contexts of use that rarely are able to collaborate each other. By exploiting these platforms, many ''vertical'' applications can be easily developed in different fields (e.g. environment monitoring [6], healthcare service [7], transportation and logistics [8], smart cities [9], smart homes [10]). However, these solutions are characterized by the use of hardware and software of a specific industry and the interoperability with other platforms is rarely supported. This means that if we have an application for monitoring the energy consumption in a Smart Home platform and one for environment monitoring in a Smart City platform there is no easy way to combine them in the scope of a new added-value application, like for example determining the levels of the internal heater relying on the weather forecast for the next hours. This lack of interoperability prevents the emergence of broadly accepted IoT ecosystems [11]. A recent McKinsey study [12] estimates

The associate editor coordinating the review of this manuscript and approving it for publication was Abdel-Hamid Soliman.

that a 40% share of the potential economic value of the IoT directly depends on interoperability gaps among IoT platforms.

These technological barriers also have negative impact on the business opportunities, especially for small innovative enterprises, which cannot afford to offer solutions across platforms.

To overcome these limitations, a number of European Projects are currently under development with the purpose of generating infrastructures among heterogeneous platforms that facilitate the development of ''horizontal'' IoT applications that exploit sensors, actuators and infrastructures made available by different platforms. Even if their realization poses different challenges across all layers of the network stack, we wish to focus on those that arise at the data management layer. First, data can be represented in different formats (e.g. CSV, XML, JSON) and present different structures for representing the same kind of information. Second, data can be represented at different spatial and/or temporal granularities (e.g. temperatures per hour in a room versus temperatures per day in a geographical area), and according to different thematic (data about traffic jams vs data about pollution). Moreover, as remarked in [13], [14], data can be incomplete and need to be enhanced by considering contextual information that can be acquired by knowledge bases and other information sources. Last but not least, data might have various semantics, including units of measurement (temperatures in Celsius or Fahrenheit degrees), accuracy, mathematical constructs, sensor types and properties and more.

All these issues, that are common in any situation in which we wish to integrate heterogeneous information systems, are further exacerbated from the variability of the available platforms, the variability of the sensors adopted for the development of IoT applications, and the lack of a well accepted IoT standard ontology. Variability refers to the several technologies with which sensors and infrastructures are realized and also to the protocols adopted for the transmission of the events and the operating systems and languages with which applications can be realized in each platform.

Moreover, information sources can change over time or provide vast amount of raw and heterogeneous data, possibly redundant, which need to be managed and processed in an understandable manner. Therefore, developing a common fixed schema or single Ontological descriptions are not feasible.

Only few approaches (like OpenIoT [15] and an extension of Orion Context Broker [16]) exploit ontologies (like the *Semantic Sensor Network* ontology [17] and the *IoT-Lite* ontology [18]) for characterizing the semantics of the data produced by sensors. Recently, ETSI is working on the development of the SAREF Ontology (`w3id.org/saref`) and making it a standard in different contexts of use. However, they provide strategies for describing the semantics of the events generated by sensors but no support is provided for enabling services to semantically integrate and manipulate heterogeneous data. Even if many approaches have been proposed for Ontology extraction and Ontology matching, they are difficult to apply in the context of sensor data where the creation of an ontology for the description of the sensors and its mapping can require much more time than the life of the IoT applications developed on top of the used platforms.

To tackle these issues, in this paper we propose to use a *Domain Ontology* for a given domain and to adopt a *semantic annotation* mechanism according to which the *sensor schema* components are mapped with the concepts of the Domain Ontology. This Ontology is developed by the domain experts that starting from standard IoT Ontologies (e.g. IoT-Lite ontology [18] and the SSN ontology [17], [19]) include concepts and relationships that are usually adopted in a specific IoT domain.

The *sensor schema* is the structure of data as a blueprint of how the sensor is constructed and retrieves data. The schema can be extracted from the data produced by sensors even when these data are represented according to different formats (e.g. JSON, CVS, or XML) and can be formed by different attributes that describe the generated events.

The sensor schema is represented relying on a flexible multi-granular Spatio-Temporal-Thematic (STT) data model.

Key point of the model is that the three dimensions are optional as well as the properties of the thematic in order to handle sensors that only produce single-simple measurements (e.g. temperatures) or structured events (e.g. list of tweets with the associated properties) along with their time-stamps and locations. This design choice has been taken to properly handle situations in which the sensors are not equipped for associating to the observations the spatio-temporal coordinates and the thematic aspects. However, this information and other metadata can be added to the events during the acquisition process. For example, the knowledge of the gateways in charge of the temperature sensors in a given zone of a city can be exploited for aggregating the temperatures relying on the gateway and assigning its location and current time as spatio-temporal dimensions.

The semantic annotation between the sensor schema and the Domain Ontology is partial in the sense that we cannot guarantee that the Ontology is always able to perfectly describe any kind of sensors. However, by means of *Data Acquisition Plans* (*DAP*s), it is possible to integrate different information on the sensors events that make them valid w.r.t. the adopted Domain Ontology (see a simple example in Figure 6). A DAP is a dataflow organized according to a direct acyclic graph where starting nodes represent sensors (either physical and/or social) that should be used for a certain analysis and can belong to different IoT platforms, internal nodes are ETL (Extraction, Transform and Load) *services* that can be exploited for filtering, integrating, and aggregating the streams produced by the sensors according to a certain logic. The final node of the graph represents the output stream that is obtained by the combination of the services on the source streams and can be exploited for conducting an analysis or

for the actuation of a given behavior (e.g. turn off the internal heater).

A DAP is considered sound when the events that it handles are expressed according to the same spatio-temporal granularities. Moreover, the integration with data taken from external sources (e.g. databases, Ontologies) and the ETL services used in the plan are correctly specified. When these conditions are met, the plan can be executed without errors.

We cannot however guarantee that the semantics of the produced events are well-defined. For this purpose a *Semantic Virtualization* process is executed.

This process consists in the annotation of the sensor schema and the output of each data acquisition service contained in the DAP with the concepts and properties occurring in the Domain Ontology. In this way, the consistency of the output stream produced by the services w.r.t. the Domain Ontology can be checked. Since the application of the services can alter the schema of the incoming streams (e.g. by introducing new properties or modifying the STT dimensions or transforming its properties) it is possible that non consistent streams become consistent. Therefore, it is important to check the consistency property for each service contained in a Data Acquisition Plan and, especially for the final output stream. Starting from the identified mapping between the schema of the sensors and the properties of the Domain concepts, it is possible to generate individuals of the Ontology that provide a semantic representation of the original sensors. Moreover, also individuals can be generated for representing the schema generated by the application of the services within a DAP. These individual can be considered instances of the Ontology and a common JSON representation can be provided for the events generated by the sensor.

When no mismatches are identified between the schema of the final output stream and the Ontology, populated with instances for the description of the DAP, we argue that the plan is ''*consistent*'' w.r.t. the Ontology, that is, it is well described at the semantic level. However, we also give the chance to work with DAPs that are not consistent. Indeed, the semantics of the final output stream can be fixed afterwards (when events are stored in a Datawarehouse or in the Cloud) or asserted by the experts that develop the plan. The instances introduced in the Domain Ontology (for modeling the sensors and the data acquisition services) can be exploited for generating transformation rules from the external representation to the internal one. When the mapping is consistent, the transformation lead to an uniform representation of the sensor events which are exposed to applications according to a well-defined semantics

A Web application, named StreamLoader, has been developed for the specification of the Data Acquisition Plans, for the verification of their consistency w.r.t. a Domain Ontology and their execution in a cluster of machines where the data produced by cross-domain sensors are collected by means of the Apache Kafka communication protocol. The DAPs are automatically translated in a Apache Spark Streaming script for being executed in a distributed environment in order to easily scale to the number of sensors and the events they generate. This language has been chosen because it permits to handle both stream and stored data. However, the DAP can be easily translated also in other frameworks. StreamLoader is equipped with an interactive environment that supports the user in charge of handling events to discover the sensors useful in a given situation, specify the adequate Data Acquisition Plan for extracting, filtering, integrating, aggregating, (eventually) storing, analyzing the events coming from the identified sensors, and visualizing the results. Once, the user is satisfied of his work, he can check the consistency w.r.t. the Domain Ontology to verify that the semantics of the outcome is well described. Then, the DAP is automatically translated in a Apache Spark Streaming script that becomes a client of Apache Kafka for the acquisition of streams of heterogeneous sensor data.

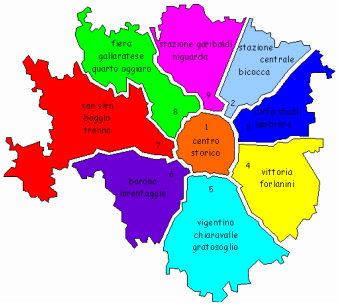
The contributions of the paper are thus:

- Decoupling of the STT model adopted by a specific platform for the representation of the data produced by its sensors and a Domain Ontology. In this way, different semantic characterizations are possible depending on the context of use of the sensors.
- Definition of a Semantic Virtualization process according to which the sensor schema is semantically characterized with the concepts of a given Domain Ontology. This characterization can be partial and can be enhanced by means of a DAP.
- Semantic characterization of an entire data acquisition process in order to verify whether the semantics of the information manipulated by means of a DAP is consistent w.r.t. the adopted Domain Ontology. When the DAP is consistent we can argue that the produced data are well described by the Ontology.
- Development of a tool that supports the user in the creation and validation of DAPs w.r.t. the adopted Domain Ontology. The obtained DAP is then translated in Spark streaming and executed in a cluster of machines.

The reminder of the paper is organized as follow. Section II introduces a motivating scenario and the high level architecture that is exploited for the Semantic Virtualization of the sensors and the specification of Data Acquisition Plans. Section III introduces the multi-granular STT data model that is used for modeling the information coming from the sensors and the different kinds of services that can be exploited for the manipulation of the sensor event streams. Moreover, the Data Acquisition Plan for our running example is presented along with the conditions for stating its soundness. In Section IV the characteristics of the Domain Ontology are introduced pointing out the representation of the STT dimensions and of the spatio-temporal granularities. The consistency of an event stream that adheres to the STT data model w.r.t. the Domain Ontology is discussed in Section V. Then, this notion is extended in Section VI to an entire Data Acquisition Plan and an algorithm is presented for the semantic description of the plan. Section VII describes the main characteristics of the

zone	LHD
1	3.0
2	2.5
3	1.0
4	1.5
5	2.5
6	2.0
7	0.5
8	2.0
9	3.0

(a)



(b)

threshold	event
$HD < 27$	wellbeing
$27 \leq HD < 40$	precaution
$40 \leq HD < 55$	dangerous
$HD \geq 55$	very dangerous

(c)

FIGURE 1. The LHD factor, the zones of the city of Milano, and the thresholds for HD.

StreamLoader system and provides some details on the GUI facilities developed for supporting the users in the verification of the soundness and consistency of a plan. Some experimental results are reported in Section VIII and Section IX discusses related work in sensor data acquisition techniques, ETL visual facilities, and IoT Ontologies. Finally, Section X concludes.

II. MOTIVATING SCENARIO AND ARCHITECTURE

In this section we outline a scenario that points out the issues that we wish to face with our tool. Then, the overall architecture of our system is described.

A. MOTIVATING SCENARIO

Suppose that the major of Milano wishes to evaluate the *Human Discomfort* (*HD*) in the different zones of the city (depicted in Figure 1(b)) due to excessive heat and humidity. His meteorologists started from the Humidex Factor [20] and proposed a formula that takes into account also the mood of people gathered by considering the tweets exchanged in the city zones. The formula is:

$$HD = [T + A + (0.555 \cdot (H - 10))] \cdot STweet + LHD$$

where, A is the accuracy of the identified temperature T (both expressed in $^{\circ}C$), H is the humidity degree (expressed in *hPa*), *LHD* is a local correction factor that takes into account the position of each zone within the area of Milano (the *LHD* factors are depicted in Figure 1(a)), and *STweet* is the percentage of tweets containing the words *hot*, *heat*, and *sweat* among those exchanged in a given zone. This formula should be evaluated every hour when the maximal temperature in the area of Milano is greater than $20^{o}C$ and the values are used to evaluate the level of alert (Figure 1(c)).

Suppose now that in the area of Milano there are many sensors that can be exploited for the computation of HD, but the sensors belong to different IoT platforms and return the events using different formats, and with different spatio-temporal granularities. A first platform contains sensors of type T_1 to gather the temperatures every 10 minutes in Celsius degree with an accuracy of the retrieved value of more or less 3 degrees (their JSON format is reported in Figure 2(a)).

A
```
{
 "metadata": {
   "thematic": "temperature",
   "location": {
    "latitude": "45464161",
    "longitude": "9.190336"
   },
 "data": [
    {
     "parameter": "timestamp",
     "dataType": "date",
     "value": "201605-15 12:10"
    },
    {
     "parameter": "measurement",
     "dataType": "real",
     "value": 21.41
    }
  ]
 }
```

B
```
<event>
 <temperature>71.24</temperature>
 <time>2016/05/15 12:10</time>
</event>

<event>
 <temperature>71.96</temperature>
 <time>2016/05/15 12:30</time>
</event>
```

C
```
15-05-2016 13:00, {45.441524, 9.085241},
'humidity', 32.3

15-05-2016 13:30, {45.441524 9.085241} ,
'humidity', 32.1

15-05-2016 14:00, {45441524 9.085241} ,
'humidity', 31.9
```

D
```
2016-05-15 12:00, z1, ['today is very hot', 'In
my home there is much heat'], 2

2016-05-15 12:01, z1, ['a nice day for me',
'hot,hot,hot', 'I am walking'], 3

2016-05-15 12:02, z1, ['I am sweat', 'It's hot in
here'], 2
```

FIGURE 2. Formats of the events generated by the sensors in the zones of Milano.

A second platform contains: *i*) sensors of type T_2 for gathering the temperatures every 20 minutes in Fahrenheit degree with the XML format in Figure 2(b) (but no geo-spatial location is provided and they act only during spring 2019); and, *ii*) sensors of type H_1 generate the events about humidity every 30 minutes with the CSV format in Figure 2(c). Finally, a third platform is equipped with sensors of type TW_1 that generate every minutes the list and the total number of tweets that are exchanged in a given zone of the city in the CSV format in Figure 2(d).

The information needed for computing *HD* is thus present in the three platforms (and in the metadata associated with their sensors) but need to be acquired, normalized, integrated and elaborated before being ready for the computation. For example the events of sensors of type T_1 presented in Figure 2 has to be enriched with information about the accuracy of the gathered temperature, as well as the events of other sensors need to be integrated with other information about their spatial, temporal and thematic dimensions. With this goal, a Data Acquisition Plan should be defined that can be applied to the streams of events that are generated by the different sensors in

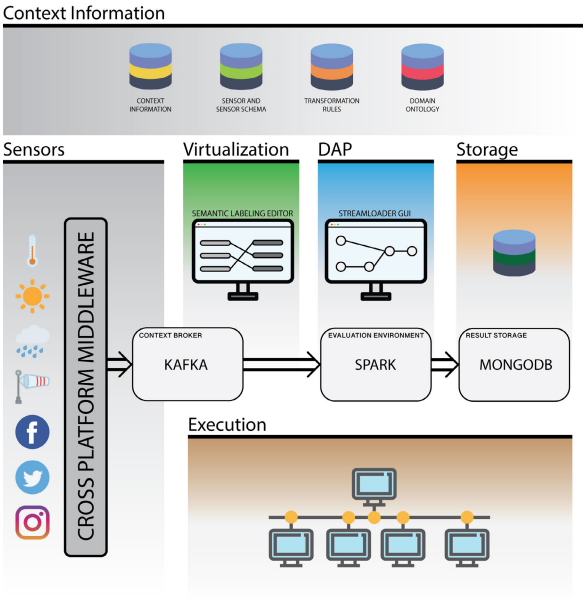


FIGURE 3. StreamLoader architecture.

order to produce the information required for the computation of *HD*. First, the schema of the sensors needs to be mapped to an internal data model in which, when available, the temporal, spatial and thematic dimensions are pointed out. This guarantees the adoption of a common model within our system (though at this level we cannot guarantee the adoption of the same semantics). Then, some services are composed for filtering, combining, aggregating and enhancing the sensors events in order to lead to the specific calculus interested in the analysis.

B. ARCHITECTURE

In order to guarantee the behavior illustrated in the motivating scenario, we will exploit the general architecture depicted in Figure 3. In our architecture we assume the presence of a cross-platform middleware that exposes the sensors made available by the different architectures. The middleware offers facilities for sensor discovery and maintenance of queues of currently available sensors with their schema structure and information about spatio-temporal granularity.

Semantic Virtualization is the process through which the physical and social sensors are associated with the StreamLoader services and described by means of the adopted Domain Ontology. Specific wrappers are then realized for transforming the input formats (e.g. CSV, XML, JSON) of the sensors of a given type in a JSON representation conforming to our STT data model. Our service is able to detect the type of format used by available/registered sensors of a gateway and handles them through a Publish-Subscribe system that is present in the cross-platform middleware. For each sensor, when available, the middleware reports the frequency at which events are generated, the spatial coordinates covered by the sensors, and the produced thematic. This information is used for the retrieval of the most adequate sensors for a given kind of analysis and for establishing the kinds of data sources that can be exploited for enriching the sensors events.

StreamLoader offers a set of different ETL (Extraction, Transform and Load) services for the data acquisition process that can be exploited by the user in the definition of the Data Acquisition Plan (DAP). These services can be composed in order to work on the events made available from the sensors through the virtualization process and leads to establish the flow of events that need to be stored or analyzed. The composition process is guided by means of the Ontological model that gives feedback on the quality of the obtained stream of events.

In our architecture new services can be added quite easily. The designer has to decide whether the new service will work on each single event (non-blocking service) or on windows of events (blocking service) and produce the corresponding code and applicability conditions. The new service will be made available/invokable for the data acquisition process. The whole system is executed on a cluster of machines and results are stored on a database in order to be subsequently analyzed.

III. OVERVIEW

In this section we detail our model for the representation of the sensor event according to the STT dimensions and discuss the developed data acquisition services. We remark that each service should produce values according to our model. This is an important feature for the verification of the consistency of the service output w.r.t. the Domain Ontology (that will be discussed in the remainder of the paper).

A. EVENT DATA MODEL

Relying on the concepts of temporal and spatial granularities, we exploit the concept of *event*, that is an instance of a thematic aspect associated with a spatio-temporal granularity [21]. Granularities are used for identifying correlations among events produced by different sensors and for imposing consistency constraints in the composition of sensor events produced by heterogeneous devices. Since some sensors might not be adequately equipped for associating all the required contextual information of a given concept of the Domain Ontology (e.g. a sensor can only provide a real number representing the temperature), or the time/location where the event has been generated, all the dimensions of a given event are considered optional in our data model as well as the properties of the thematic.

Further metadata (e.g. unit of measure, precision) can be associated with the event during the acquisition process. The possibility offered through this model to represent the STT dimensions generated by the sensors is somehow a ''semantic'' information extracted from data. For example a gateway in charge of a set of temperature sensors disseminated in a given zone of a city can calculate the average temperature and assign the time/location this observation refers to. However, this kind of semantic can be directly desumed by the events generated from the data and their formats and needs to be

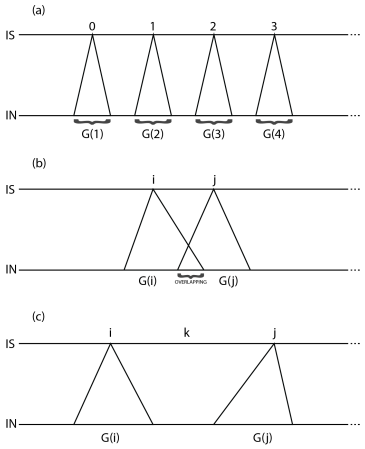


FIGURE 4. Graphical representation of temporal granularity.

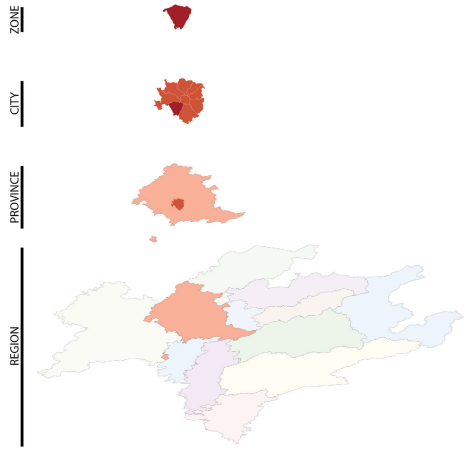


FIGURE 5. Graphical representation of spatial granularity.

validated from the domain experts (as we will discuss in the following sections).

1) SPATIAL, TEMPORAL AND THEMATIC DIMENSIONS

In our work, we consider the definition of temporal and spatial granularities adopted in [21], [22]. Temporal and spatial granularities are mapping functions from an index set $\mathcal{IS}$ to the power sets of the temporal and spatial domains, respectively. The temporal domain is represented as a pair $(\mathbb{N},\leq)$, where $\mathbb{N}$ is the set of natural numbers representing the set of *time instants* and $\leq$ is a relation order on $\mathbb{N}$. The spatial domain contains geometric objects represented in one or two dimensions (e.g. points, lines, and regions).[1] Examples of temporal granularity include `second`, `minute`, `day` with the usual meaning adopted in the Gregorian calendar, whereas, `meter`, `kilometer`, `feet`, `yard`, `zone` and `city` are examples of spatial granularities. Let $\mathcal{G}_\mathcal{T}$ and $\mathcal{G}_\mathcal{S}$ denote a set of time and spatial granularities.

A *granule* is a sub set of a domain corresponding to a single granularity mapping, that is, given a granularity G and an index $i \in \mathcal{IS}$, $G(i)$ is a granule of G that identifies a subset of the corresponding domain. Figure 4(a) shows this property in case of temporal granularities. Granules of the same granularity are disjoint, so there are no granule overlapping. Figure 4(b) shows a violation of this definition while Figure 4(c) presents another violation: non-empty temporal granules preserve the order of the temporal domains. Each non-empty granule of a granularity G is represented by means of the "textual representation" as a *label* (e.g. a label for `day` can be in the form `mm/dd/yyyy`). Granules are used to specify the valid spatio-temporal bounds on attribute values.

Different granularities provide different partitions of their domains because of the diverse relationships that can exist among granularities, depending on the inclusion and the overlapping of granules [21], [22]. A granularity G is said to be *finer than* a granularity H, denoted $G \preceq H$, if for each index i, there exists an index j s.t. $G(i) \subseteq H(j)$ [23].

For example, temporal granularity `second` is *finer than* `minute`, and granularity `month` is finer than `year`. Likewise, spatial granularity `zone` is finer than `city`. Figure 5 shows different four granularities that are in the *finer-than* relationship. Indeed, `zone` $\preceq$ `city` $\preceq$ `province` $\preceq$ `region`. In the example, the zone/granule z_1 is a part the city/granule `Milan` which in turn is a part of the province/granule `MilanProvince` and of the region/granule `Lombardy`. Besides spatial and temporal information, a semantically rich sensor network would provide thematic information for discovering and analyzing sensor events. Thematics represent the type of event that is generated by a sensor.

Examples of thematic are `temperature`, `humidity`, `wind speed`, etc. In the following, $\mathcal{TH}$ represents the set of thematics.

2) TEMPORAL AND SPATIAL TYPES AND VALUES

Starting from basic domains (like `int`, `real`, `boolean`), denoted by D_i, we consider structured types, like records and lists, represented according to the JSON format. The set of types is denoted by $\mathcal{T}$ and is used for the representation of the thematic properties. Let $\mathcal{A}$ be a set of labels, the set of legal values $\mathcal{V}$ for the types $\mathcal{T}$ is inductively defined as follows:

- if $d \in D_i$ and $a \in \mathcal{A}$, then $(a : d) \in \mathcal{V}$;
- if $d_1 \in D_1, \ldots, d_n \in D_n$ and $a \in \mathcal{A}, (a : [d_1, \ldots, d_n]) \in \mathcal{V}$;
- if $\{d_1, \ldots, d_n\} \subseteq \mathcal{V}$ and $a \in \mathcal{A}$, $(a : [d_1, \ldots, d_n]) \in \mathcal{V}$;
- if $(a_1 : d_1) \in \mathcal{V}, \ldots, (a_n : d_n) \in \mathcal{V}$ with $a_i \neq a_j$ $(i \neq j)$, then $(a_1 : d_1, \ldots, a_n : d_n) \in \mathcal{V}$.

Given a type $\tau \in \mathcal{T}$, a spatial granularity $G_S \in \mathcal{G}_\mathcal{S}$, and a temporal granularity $G_T \in \mathcal{G}_\mathcal{T}$, we denote with: $Spatial_{G_S}(\tau)$, a spatial type; $Temporal_{G_T}(\tau)$, a temporal type, and $Spatio\text{-}Temporal_{(G_T,G_S)}(\tau)$ a spatio-temporal type [21].

Their legal values are defined as partial functions that map each granule $i \in \mathcal{IS}$ (or pair of granules (i, j) for spatio-temporal types) to the legal values for τ.

[1] Points are modeled by two coordinates; lines by a list of points, regions by their boundary lines.

3) STT EVENTS AND STREAM DATA MODEL

Relying on the temporal and spatial granularities, we are now ready to present the concept of *event*. Our definition covers different kinds of events that can be associated with the STT dimensions. In the definition $\perp$ denotes a missing component.

Definition 1: (Event Type). Let $\tau \in \mathcal{T}$ be a type, $G_T \in \mathcal{G}_\mathcal{T} \cup \{\perp\}$ a temporal granularity, $G_S \in \mathcal{G}_\mathcal{S} \cup \{\perp\}$ a spatial granularity, and $th \in \mathcal{TH} \cup \{\perp\}$ a thematic. An Event Type, denoted $Event^{th}_{\langle G_T, G_S\rangle}(\tau)$, can be:

- if $G_T = G_S = \perp$, a list of pairs $\langle th, \tau\rangle$.
- if $G_T = \perp$ and $G_S \neq \perp$, a partial function that maps G_S-granules (referred to by their indices) to pair $\langle th, \tau\rangle$ $(Spatial_{G_S}(\langle th, \tau\rangle))$.
- if $G_T \neq \perp$ and $G_S = \perp$, a partial function that maps G_T-granules to pair $\langle th, \tau\rangle$ $(Temporal_{G_T}(\langle th, \tau\rangle))$.
- if $G_T \neq \perp$ and $G_S \neq \perp$, a partial function that maps (G_T, G_S)-granules to pair $\langle th, \tau\rangle(Spatio\text{-}Temporal_{\langle G_T, G_S\rangle}(\langle th, \tau\rangle))$. □

Example 1: Consider the scenario introduced in Section II-A. In this scenario we can identify four types of events generated by sensors as follows:

- T_1: $Event^{\{\texttt{temperature}\}}_{\langle \texttt{10 minute,point}\rangle}(\{\texttt{temperatureVal} : \texttt{real}\})$.
- T_2: $Event^{\{\texttt{temperature}\}}_{\langle \texttt{20 minute},\perp\rangle}(\{\texttt{temperatureVal} : \texttt{real}\})$.
- H_1: $Event^{\{\texttt{humidity}\}}_{\langle \texttt{30 minute,point}\rangle}(\{\texttt{humidityVal} : \texttt{real}\})$.
- TW_1: $Event^{\{\texttt{tweet}\}}_{\langle \texttt{minute,zone}\rangle}(\{\texttt{tweets} : \texttt{list(string)}, \texttt{numTweets} : \texttt{int}\})$.

This representation of the event type of each sensor is exploited for modeling the events generated by the sensors in our uniform JSON format that is used for simplifying their processing. Note that the temperatures in T_1 are expressed in Celsius, where those in T_2 are expressed in Fahrenheit, and the notation `point` indicates the event geo-position (latitude, longitude). □

Relying on the events, we can characterize an event stream.

Definition 2: (Event Stream). Let $\tau \in \mathcal{T}$ be a type, $G_T \in \mathcal{G}_\mathcal{T} \cup \{\perp\}$ a temporal granularity, $G_S \in \mathcal{G}_\mathcal{T} \cup \{\perp\}$ a spatial granularity, $th \in \mathcal{TH} \cup \{\perp\}$ a thematic, $[t_s, t_e]$ a temporal interval, and S a set of spatial values. An event stream is a 6-tuple

$$\langle G_T, G_S, th, [t_s, t_e], S, Event^{th}_{\langle G_T, G_S\rangle}(\tau)\rangle$$

where: if $G_S \neq \perp$ then $G_S \preceq Gran_S(S)$, and if $G_T \neq \perp$ then $G_T = Gran_T(t_s) = Gran_T(t_e)$. □

An event stream describes thus a sequence of events that is associated with meta information related to the interval in which the flow is acquired and the STT dimensions (when available). The meta information is exploited for imposing the integrity constraints on the produced events.

Example 2: Consider the event type associated with the sensors of type TW_1.

The event streams of each zone of Milano produced in the year 2019 can be characterized by $\langle$`minute, zone, tweet`, [“1/1/2019-00:00″,“1/1/2020-00:00″], {`z1, ..., z9`}, $Event^{\{\texttt{tweet}\}}_{\langle \texttt{minute,zone}\rangle}(\{\texttt{tweets:list(string)}, \texttt{num-Tweets:int}\})\rangle$, where {⟨“15/05/2019-12:00″, `z1, tweet`, {`tweets`:[’today is very hot’, ’In my home there is much heat’], `numTweets`:2}⟩, ⟨“15/05/2019-12:01″, `z1, tweet`, {`tweets`:[’a nice day forme’,’hot,hot,hot’, ’I am walking’], `numTweets`:3}⟩, ⟨“15/05/2019-12:02″, `z1, tweet`, {`tweets`:[’I amsweat’,’It’s hot inhere’], `numTweets`:2}⟩ is a stream of legal values. □

B. DATA ACQUISITION SERVICES

Several services can be devised for processing and combining the streams produced by the sensors that adopt the event stream model. Other services are provided for manipulating the streams (for filtering, transforming, aggregating, composing and for event detection). Stream manipulation services are classified in *non-blocking* and *blocking* services. In the remainder of the section we discuss and detail the developed services whose formal specification is sketched in Table 1.

In the presentation we use the following notations. $\mathcal{P}rop(o)$ denotes the set of properties occurring in o, whereas $Gran_T(o)/Gran_S(o)$ represents the temporal/spatial granularity of o and $\mathcal{T}ype(a, s)$ denotes the type domain of property a in the event stream s. Other services are provided for manipulating the streams (for filtering, transformation, aggregation and composition, and event detection). Each service is characterized by several parameters that are exploited for the verification of the DAP soundness and consistency.

1) NON-BLOCKING SERVICES

These services are applied on each single event and thus do not require to maintain caches.

The `filter` service allows one to remove events that do not adhere to a condition *cond* expressed on the values of the event stream s. *cond* is a boolean expression on the properties specified in the schema of the stream s.

The `enrich` service allows the inclusion of extra information to the stream according to a knowledge base *KB* or the Domain Ontology or static information sources. The binding between the current event and one of these sources is realized through the join predicate *pred*. This service is particularly important for enriching the events with contextual information that are local to where the event is generated (for example for associating the local correction factor *LHD* in the computation of *HD*). The spatio-temporal granularity of s needs to be compliant with the one adopted in *KB* for a sound enrichment of the event stream. The `virtual-property` service allows the inclusion of a new property p to the schema of s according to the specification *spec*. *spec* is an arithmetic expression for determining the value of p relying on the values of the properties of s. The `transform` service applies the transformation function *trans* on the properties $a_1, \ldots, a_n$ of events in s. At the current stage the following transformation functions have been considered: *i*) for changing the unit of measure (e.g. from yards to meters) or geographical coordinates (from one standard to another one); *ii*) for checking that

TABLE 1. Data acquisition services.

Service	Symbol	Condition
Filter	$\sigma(s, cond)$	$\mathcal{P}rop(cond) \subseteq \mathcal{P}rop(s)$
Enrich	$\propto_{pred}^{KB} s$	$Gran_S(s)$ and $Gran_T(s)$ compliant with the one adopted in KB, $pred \equiv a_1 = b_1 \wedge ... \wedge a_n = b_n$ with $\{a_1, .., a_n\} \subseteq \mathcal{P}rop(s)$ and $\{b_1 .., b_n\} \subseteq \mathcal{P}rop(KB)$
Virtual property	$\uplus_s \langle p, spec \rangle$	$p \notin \mathcal{P}rop(s), \mathcal{P}rop(spec) \subseteq \mathcal{P}rop(s)$
Transform	$\Diamond_{trans}^{\{a_1,\ldots,a_n\}} s$	$\{a_1, .., a_n\} \subseteq \mathcal{P}rop(s)$
Aggregation (B)	$@_{op}^{t,\{a_1,\ldots,a_n\}}(s)$	$Gran_T(s) \preceq Gran_T(t), \{a_1, .., a_n\} \subseteq \mathcal{P}rop(s), op \in \{\texttt{count}, \texttt{avg}, \texttt{sum}, \texttt{min}, \texttt{max}\}$
Union (B)	$\cup^{t,th'}(\{s_1, .., s_n\})$	$s_i \neq s_j (i \neq j), Gran_T(s_i) = Gran_T(s_j), Gran_S(s_i) = Gran_S(s_j), \forall a \in \mathcal{P}rop(s_1) \cap \mathcal{P}rop(s_2) : \mathcal{T}ype(a, s_1) = \mathcal{T}ype(a, s_2)$
Join (B)	$s_1 \bowtie_{pred}^{t,th'} s_2$	$s_1 \neq s_2, Gran_T(s_1) = Gran_T(s_2), Gran_S(s_1) = Gran_S(s_2), pred \equiv a_1 = b_1 \wedge ... \wedge a_n = b_n$ with $\{a_1, .., a_n\} \subseteq \mathcal{P}rop(s_1), \mathcal{T}ype(a_i, s_1) = \mathcal{T}ype(b_i, s_2)$, and $\{b_1 .., b_n\} \subseteq \mathcal{P}rop(s_2)$
Trigger On (B)	$\oplus^{\texttt{ON},t,th'}(s, \{s_1\}, cond)$	$Gran_T(s) \preceq Gran_T(t), \mathcal{P}rop(cond) \subseteq \mathcal{P}rop(s)$
Trigger Off (B)	$\oplus^{\texttt{OFF},t,th'}(s, \{s_1\}, cond)$	$Gran_T(s) \preceq Gran_T(t), \mathcal{P}rop(cond) \subseteq \mathcal{P}rop(s)$
Convert (B)	$\ominus_{tg,sg}^{t}(s, cf)$	$Gran_T(s) \preceq tg, Gran_S(s) \preceq sg, cf \in \{\texttt{first}, \texttt{last}, \texttt{proj(index)}, \texttt{main}, \texttt{all}, \texttt{min}, \texttt{max}, \texttt{avg}, \texttt{sum}\} \cup \mathcal{UDF}$

data conform to given validation rules (e.g. dates conforming to given patterns); *iii*) for changing the case of letters. However, further functions can be easily integrated in our framework.

2) BLOCKING SERVICES

Blocking operators require to maintain a cache of events for a temporal interval t. At the end of the interval, the events collected in the cache are processed by the operator and the obtained result produced to the upcoming operators. The blocking operators are marked with a `B` in Table 1.

The `aggregation` service aggregates the events of s on the properties $a_1, \ldots, a_n$ and applies the aggregation function $op \in \{\texttt{count}, \texttt{avg}, \texttt{sum}, \texttt{min}, \texttt{max}\}$ on the other properties. The temporal granularity of t needs to be compatible with the one of s. Moreover, the properties $a_1, \ldots, a_n$ need to be included in those appearing in s. The `union` service allows the union of the events produced by different sensors in the same time window and produces a new stream of events of thematic th'. Indeed, in this case the user can specify whether the generated sequence of events has a new thematics or maintain one of those of the incoming streams. The `join` service creates a correspondence among the events of two streams when their temporal and spatial granularities are identical and the join predicate *pred* is verified. The evaluation of the join is window-based, that is, it is executed on the events collected from the two streams in the temporal interval t and produce events of thematic th'. The `trigger-on/off` service activates/deactivates the production of events from a stream $\{s_1\}$ when a condition *cond* is verified on the events collected from s in the time interval t. We remark that the stream s_1 can be the result of the applications of different data acquisition services whose thematics is th'.

The `convert` service applies a coercion function *cf* for changing the spatio-temporal granularity of the stream s. The temporal/spatial granularities *tg* and *sg* should be coarser than the one used in s. Coercion functions [22] can be classified into three categories: *selective*, *aggregate*, and *user-defined* coercion functions. Selective coercion functions are `first`, `last`, `proj(index)`, `main`, and `all`. Coercion function `proj(index)`, for each granule in the coarser granularity, returns the value corresponding to the granule of position `index` at the finer granularity. Coercion function `first` and `last` are the obvious specializations of the previous one. Coercion function `main`, for each granule in the coarser granularity, returns the value which appears most frequently in the included granules at the finer granularity. Coercion function `all`, for each granule in the coarser granularity, returns the value which always appears in the included granules at the finer granularity if this value exists, the null value otherwise. Functions are `min`, `max`, `avg`, and `sum` corresponding to the well-known SQL aggregate functions. User-defined coercion functions (named $\mathcal{UDF}$) are services that can be dynamically loaded and preserve the relationships among granularities.

C. DATA ACQUISITION PLAN

A Data Acquisition Plan is a directed acyclic graph that represents the flow of operations on sensor data. This representation helps the user on composing services in order to transform, aggregate, filter and store information. For depicting the concepts connected with the DAP, we use graphical blocks whose meaning in reported in Table 2.

Definition 3: (A Data Acquisition Plan). A Data Acquisition Plan is a direct acyclic graph $DAP = (V, E)$, where V is a set of vertices representing sensors (V_s), data acquisition services (V_{op}), and destination (V_d), and E represents the flow of events that adheres to our model. □

We are now ready for the construction of the DAP for computing the human discomfort of our motivating example.

Example 3: First, all the temperatures measured by the sensors of type T_1 and T_2 are taken. Before computing the union of the observed temperatures, we need to convert them to the same spatio-temporal granularity, the same unit of measure, and include the accuracy. Thus, on the streams produced by sensors of type T_1 the following services are invoked:

$$(enrich) \quad t_1^1 = \propto_{\texttt{accuracy}}^{O} t_1$$
$$(convert) \quad t_1^2 \ominus_{hour,zone}^{1\ hour} (t_1^1, avg)$$

TABLE 2. Sensors, services, nodes, and edges representation.

Symbol	Description
●	Source node. It can have only 1 outgoing edge and no incoming edge.
▭	`filter`, `enrich`, `virtual property`, `transform`, `aggregation` and `convert` service. All these services have a single incoming and outgoing edge.
◉	Destination node. No outgoing edges are allowed and it can receives data from a single node.
‖	`union` service. It has two incoming edges and a single outgoing edge.
▮	`join` service. It presents the same shape of the `union` because it has 2 incoming edges and a outgoing edge.
⋈	`trigger` service. It represents the trigger event and action (2 incoming edges and 1 outgoing edge).
→	Edge that connects each node and service and expresses the flow direction.

By exploiting the Domain Ontology the accuracy and zone of the sensors are included in the stream. Then, by exploiting the relationship between the `point` and `zone` spatial granularities and the relationship between the `10 minute` and 1 `hour` temporal granularities, the average temperature is computed for each hour and zone. Therefore, the event type stream of t_1^2 is `(hour, zone, temperature, {temperatureVal:real, accuracy:real})`.

Similar behavior is followed for the streams produced by sensors of type T_2 with the exceptions that: *i*) the temperature values are expressed in Fahrenheit and need to be transformed; *ii*) the spatial dimension is missing and can be included by means of the data contained in the Domain Ontology; and *iii*) temperatures are collected every 20 minutes.

$$(transform) \quad t_2^1 \Diamond_{far2celsius}^{temperature} t_2$$
$$(enrich) \quad t_2^2 = \propto_{zone}^{O} t_2^1$$
$$(convert) \quad t_2^3 \ominus_{hour,zone}^{1\ hour} (t_2^2, avg)$$

`(hour,zone,temperature, {temperatureVal:real})` is the event type stream of t_2^3. Since the STT dimensions of t_2^3 and t_1^2 are the same, the union of the generated events is possible by the invocation $t_3^1 = \cup^{1\ hour,temperature}(\{t_1^2, t_2^3\})$. We remark that the types produced by the two kinds of sensors are different (`accuracy` is missing in t_2^3). Therefore the value `0.0` is associated for this property to the events of t_2^3 because its type is `real`. At this point we can enrich the stream t_3^1 with the local correction factor *LHD* (that are specified for each zone of Milan – see Figure 1(a)) by means of the service $t_3^2 = \propto_{zone}^{table(lhd)} t_3^1$. The resulting type is `(hour,zone,temperature, {temperatureVal:real, accuracy:real, lhd:real})`.

For what concern the humidity, we need to exploit the relationship between the `point` and `zone` spatial granularities and the relationship between the `30 minute` and 1 `hour` temporal granularities to enhance the stream with the zone and compute the average humidity per hour and per zone. Similar behavior should be adopted for associating the zone to the streams about humidity an aggregating at the `hour granularity`. For what concern the humidity, we need to exploit the relationship between the `point` and `zone` spatial granularities and we need to enrich it with information about the timestamp. Then, we need to compute the average humidity per zone. By contrast, for the tweets we need to select those containing the terms hot, heat and sweat and add a virtual property containing the number of selected tweets.

$$(convert) \quad h_1^1 \ominus_{hour,zone}^{1\ hour} (h_1, avg)$$
$$(enrich) \quad h_1^1 = \propto_{hour}^{O} h_1$$
$$(filter) \quad tw_1^1 = \sigma(tw_1, like(tweets, ".*hot*.")||\ like(tweets, ".*heat*.")||\ like(tweets, ".*sweat*."))$$
$$(virtual\ property) \quad tw_1^2 = \uplus_{tw_1^1}\langle numPos, COUNT(tweets)\rangle$$

`(hour,zone,humidity,{humidityVal:real})` is the event type stream of h_1^1, whereas the event type stream of tw_1^2 is `(minute,zone,tweet, {tweets:list(string), numTweets:int, numPosTweets:int})`.

At this point the information about temperature, humidity and tweets are organized according to the same spatio-temporal granularities and can be joined to obtain the terms of the formula and a virtual property can be included in the resulting stream for storing the value of the human discomfort for each hour.

$$(join) \quad hd_1^1 = h_1^1 \bowtie_{h_1^1.zone=tw_1^2.zone}^{1\ hour,humanDisc} tw_1^2$$
$$(join) \quad hd_1^2 = hd_1^1 \bowtie_{hd_1^1.zone=t_3^2.zone}^{1\ hour,humanDisc} t_3^2$$
$$(virtual\ property) \quad hd_1^3 = \uplus_{hd_1^2}\langle hd, (numPos/numTweets) * (temperature + accuracy + (0.555 * (humidity - 10))) + lhd\rangle$$

`(hour,zone,humanDisc,{temperatureVal:real, accuracy:real,lhd:real, humidityVal:real, tweets: list(string), numTweets:int, hd: real}, numPosTweets:int)` is the event type stream of hd_1^3.

The process for the calculation of the human discomfort is triggered only when the maximal temperature in Milan in the last hour is greater than 20^o Celsius. Supposing to use only the sensors of type T_1 for the trigger, the events produced by these sensors need to be converted to the `hour` time granularity and `city` spatial granularity. The following two services are thus invoked:

$$(convert) t_3^1 = \ominus_{hour,city}^{1\ hour}(t_1, max)$$

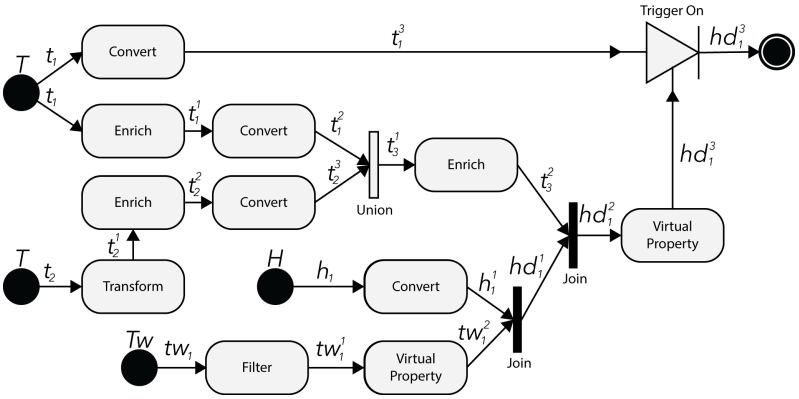


FIGURE 6. Graph representation of the running example's data acquisition plan.

$$(trigger\ on)hd = \oplus^{\mathrm{ON},1\ hour,humanDisc}(t_3^1, \{hd_1^3\},)$$
$$temperature > 20$$

Therefore, the process for the calculation of the stream hd_1^3 is activated only when the trigger condition is verified in the last hour. The event type of the final stream is the same of hd_1^3. A graphical representation of this Data Acquisition Plan is reported in Figure 6. The figure points out the flows of events that are generated by the physical and social sensors and those produced by the application of the different services. Different symbols are used for representing physical and social sensors (bold circles), services (rectangles with rounded corners, the triangle for representing the event trigger and the corresponding action, and the thin rectangles for representing the union/join services) and the destination node (bold double line circle). □

In the graph representation of a DAP, we can specify constraints for obtaining sound execution. Given a $DAP = (V, E)$, the following functions are used: $deg^-(v)$ represents the number or edges incoming in $v \in V$, whereas $deg^+(v)$ is the number of outgoing edges from v.

Definition 4: (A Sound Data Acquisition Plan). Let $op_1, \ldots, op_n$ be the services used in a Data Acquisition Plan $DAP = (V, E)$.[2] DAP is sound when: *i*)$V_s \neq \emptyset$, $|V_d| = 1$; *ii*)$\forall v \in V_s$, $deg^-(v) = 0$, $\forall v \in V_d, deg^+(v) = 0$; and *iii*) for each op_i, $v_i \in V_{op}$, $deg^-(v_i)$ corresponds to the number of input streams for op_i, each parameter required by op_i is correctly specified, and the conditions reported in Table 1 are verified on the events of the incoming edges. □

IV. DOMAIN ONTOLOGY

The STT data model so far presented allows one to produce events whose correctness is left to the user in charge of its creation. To support the user in this activity, a Domain Ontology is included within our system for each supported domain. The purpose of the adopted ontology is to guarantee that the properties specified for a given concept (in a certain domain) actually occur in the events produced by the sensors and that the final events (i.e. those generated at the end of the Data Acquisition Plan) are compliant with the spatial, temporal and thematic concepts made available in the adopted Domain Ontology. When this requirement is met we can say that the generated events are consistent w.r.t. the Domain Ontology.

For the design of the Domain Ontology we take into account two of the most notable works in this field, the IoT-Lite ontology [18] and the SSN ontology [17], [19] from which IoT-Lite ontology is derived.

Then, these ontologies are aligned with other foundational ontologies in order to make spatial, temporal and thematic commitments explicit by using further concepts and relations for better explaining their intended meaning. They have been chosen as the upper ontologies because they have ontological frameworks and concepts (e.g. qualities, temporal entities, units of measurements, geospatial positions info) that are needed for making our Domain Ontology consistent w.r.t. the STT model. Figure 7 depicts the concepts of the ontologies integrated in our Domain Ontology and the main relationships between them. For example, DOLCE-UltraLite (DUL) [24] introduces the concept of `DUL:Amount` that is used to identify and specify the type of a value (real, integer, listString, etc). `qu:QuantityKind`, and `qu:Unit` derives from QU[3] and they provides, respectively, information about instances for thematic values and measurement units (Celsius, Fahrenheit, kilometers, etc). Time[4] is used to label temporal instants or interval while Geo[5] introduces the class `Geo:Point` that is necessary in order to define latitude and longitude. Two new classes, used to represent instances coming from social sensors and characteized by the prefix `do:` in the Domain Ontology, have been introduced without taking into consideration any other existing ontologies: `do:Facebook` and `do:Twitter`. The ontology obtained by the integration of these components can be further extended by domain experts with concepts and

[2]For the sake of simplicity, the service op_i corresponds to vertex v_i.

[3]http://purl.oclc.org/NET/ssnx/qu/qu-rec20
[4]http://motools.sourceforge.net/timeline/timeline.html
[5]http://www.w3.org/2005/Incubator/geo/XGR-geo-ont/

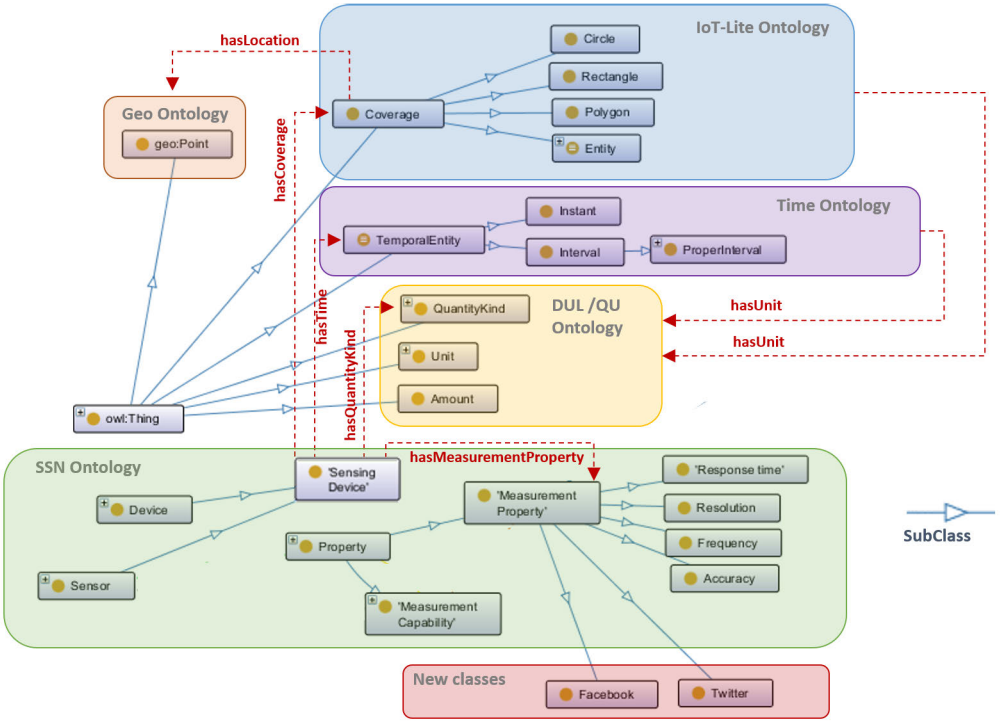


FIGURE 7. Relationships between the ontologies that compose our Domain Ontology.

relationships specifically tailored for representing peculiar characteristics of a given domain of interest. In our graphics we use standard notations for representing our Domain Ontology and its instances. Specifically, dashed rectangles represent instances, whereas dashed lines are used for linking related instances according to a given relation. The dashed circle is used for modeling the data value linked to an instance and created by the data property assertion `hasDataValue`. $\mathcal{I}(\mathtt{C})$ denotes the set of instances of class `C`, and $\mathtt{C}_1 \sqsubseteq \mathtt{C}_2$ denotes that $\mathtt{C}_1$ is subclass of $\mathtt{C}_2$.

In the following we discuss the ontological representation of the STT dimensions included in our Domain Ontology.

A. SPATIAL DIMENSION

The spatial dimension provides information regarding the sensor location, in terms of either a geographical reference system or named location. According to the specification of classes of the IoT-Lite ontology, the spatial information can be modeled using the class `iot-lite:Coverage` that acknowledges that a location can be related to the coverage of an IoT device (i.e. a temperature sensor inside a room has a coverage of that room).

The property `hasPoint` of the `iot-lite:Coverage` class states its location by using the `geo:Point` class and its latitude and longitude properties. By contrast, to specify that a location is a country, a region, a province, a city, etc new subclasses of the class `iot-lite:Entity` (subclass of `iot-lite:Coverage`) are inserted in the Ontology. The spatial granularity G_S of a location is specified through a relation of order among the individuals of the subclasses of the class `iot-lite:Entity` and is guaranteed by the unary association `isPartOf` that, for instance, can be used for describing that a province is a part of a region that, in turn, is a part of a country. The spatial granularity G_S can be also specified by using the association `hasSystemReference` between the class `iot-lite:Entity` and the class `Geo-SubdivisionStandard` that we defined for instantiating concepts concerning Standard Geographical Administrative Subdivisions. A possible instance of this class can be used for referring to the International Standard for country codes and the ISO 3166[6] codes for their subdivisions. This standard defines the codes for the names of countries, dependent territories, special areas of geographical interest, and their principal subdivisions (e.g., provinces or states).

In the IoT-Lite ontology, the class `iot-lite:Coverage` is then associated to the classes: `iot-lite:Rectangle`, `iot-lite:Polygon`, and `iot-lite:Circle` in order to represent the coverage area. Instances of the class `iot-lite:Rectangle` are used for describing that the coverage is made up by giving two geographical points (instances of the class `geo:Point`) which are the opposite corners of a rectangle. Instances of the class `iot-lite:-Polygon` describe that the coverage is made up by linking several geographical points by straight lines. Finally, instances of the class `iot-lite:Circle` specify that the coverage of a sensor is a circle with the center in a

[6] `www.iso.org/iso/home/standards/country_codes.htm`

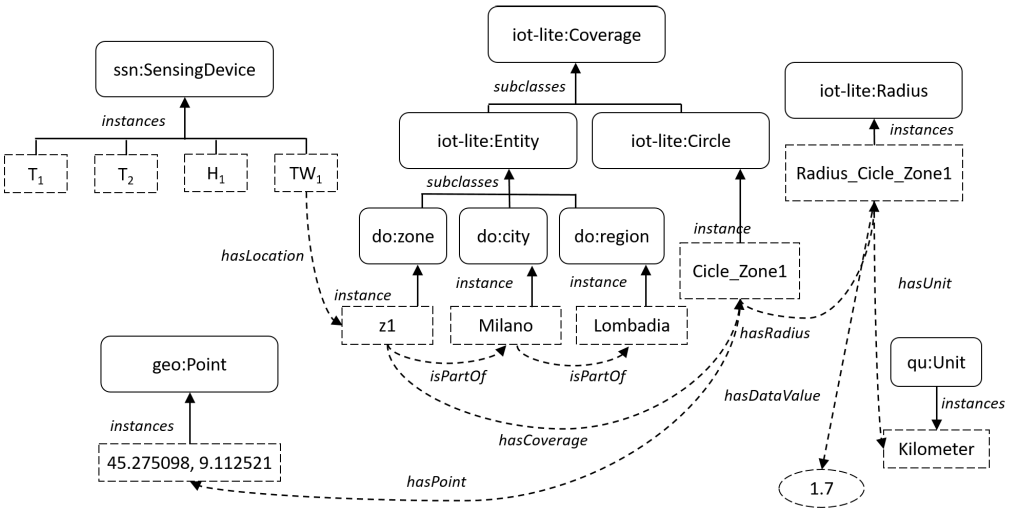


FIGURE 8. Twitter sensor of type $\mathtt{TW}_1$.

geographical point and with a given radius. The radius is specified by using the class `iot-lite:Radius` that has to be then associated with an unit of measure (instance of the class `qu:Unit` for indicating that the value of the radius is expressed in `meter`, `kilometer`, `feet`, `yard`, etc.).

For specifying the relation of order among different units of measurement (i.e. meters vs kilometers or minutes vs hours), the class `qu:Unit` can be linked to the class `qu:SystemOfUnits` that represents the concept of "system of units". This concept is defined as set of base units and derived units, together with their multiples and sub-multiples, defined in accordance with given rules, for a given system of quantities. For example, the most widely used systems of quantities and system of units are the International System of Quantities (ISQ) and the International System of Units (SI).

Example 4: In our running example we wish to represent a Domain Ontology for evaluating the human discomfort in differ zones of Milano. With this aim, domain experts have specified a Domain Ontology with a set of classes and relationships useful to detect human discomfort events. For modeling the spatial dimension, we wish to represent the single points where the sensors are located, the zones of Milano and the entire city. Moreover, we wish to model the finer-than relationships existing among these granularities.

Figure 8 reports an example of spatial dimension instantiation for a sensor of type $\mathtt{TW}_1$ of our running example.

In this case the Twitter sensor is located in zone 1 of Milano (instance of the new class `do:zone` we inserted as subclass of the class `iot-lite:Entity`) whose shape is rounded by using a circle (instance of the class `iot-lite:Circle`). The Circle coverage is made up by giving the location of the sensor as the center of the circle (e.g. a geo point instance of the class `geo:Point`) and the radius as a `DataProperty`. The granularity is specified though the properties `isPartOf` that connect the instances `z1`, `Milano` and `Lombardia` of the new classes `do:zone`, `do:city` and `do:region` we introduced in the Domain Ontology. The other zones in Milano are represented in a similar way through the class `iot-lite:Polygon` for modeling the vertices delimiting each area. □

B. TEMPORAL DIMENSION

The temporal dimension in sensor events and its observation and/or measurement data are used for describing attributes such as time zone and measurement timestamp.

For modeling such concepts, the ontology is integrated with the Timeline Ontology [25] that extends OWL-Time with various temporal concepts such as `Instant`, `Interval`, and `Interval relationships`. In detail, we are interested in two main subclasses of Temporal Entity: `tl:Instant` and `tl:Interval`. The instances of the class `tl:Instant` are used for describing instants of time, and the instances of the class `tl:Interval` are used for specifying intervals by means of which we describe that a sensor gathers events from time $\mathtt{t}_1$ to time $\mathtt{t}_2$ by means of the properties `Interval starts` and `Interval finishes` of the class `tl:Proper Interval` subclass of the class `tl:Interval`.

As with the `iot-lite:Coverage`, through the association between the temporal entities and the class `qu:Unit`, it is possible to specify the granularity of the detected time (`day`, `hour`, `minute`, `second`) and evaluate their relationships. The temporal dimension $t \in T$ of events coming from a sensor at a granularity G_T, is strictly related to the instant (the timestamp) of gathering of the feature of interest we want to monitor in an event such as temperature, humidity, etc. This information is modeled by using an instance of the class `tl:Instant`.

Example 5: Figure 9 describes an instance of the temporal dimension associated with a sensor. In this case, the sensor of type $\mathtt{T}_2$ is linked to a timestamp used for describing the instant of time to which the specified sensor refers (instance

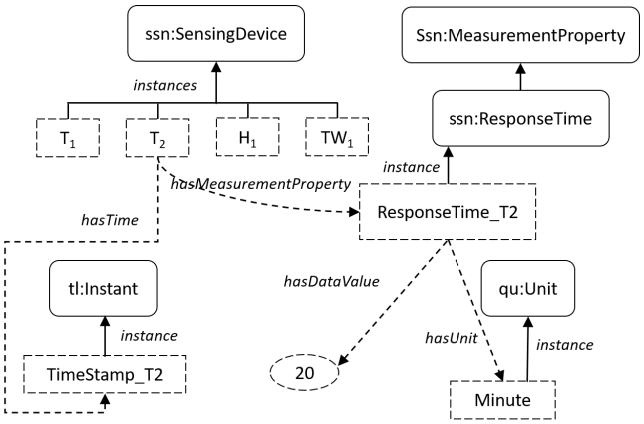


FIGURE 9. Temporal dimension of sensor of type T_2 .

of the class `ttl:Instant`). The other sensors of our motivating scenario are linked to instances of the same class for describing their timestamps. □

Through the property `hasMeasurementProperty` we can link a sensor to the class `ssn:ResponseTime`, subclass `ssn:MeasurementProperty` ($\sqsubseteq$ `iot-lite:Property`) of the SSN ontology for specifying the time of sampling. Therefore, the concept "response time" is used for describing the granularity at which events are generated.

Example 6: In our motivating example, a sensor of type T_2 gathers temperature every 20 minutes. This situation is described in Figure 9 by introducing an instance of the class `ssn:ResponseTime`. An instance of the class `qu:Unit` is then used for specifying the time granularity `minute`. □

C. THEMATIC DIMENSION

This dimension refers to the type of events that are observed and is described by a record of property-values pairs. We use this representation because the single observation can be enriched by other information that can be directly generated by the sensor or added during the data acquisition process.

In our ontology a thematic is an instance of the class `qu:QuantityKind` of the IoT-Lite ontology, and is used for describing the meaning of the values dispatched by a sensor. The abstract classifier `qu:QuantityKind` represents the concept of "kind of quantity" that is defined as "*aspect common to mutually comparable quantities*" [26]. A quantity is defined as a characteristic of a phenomenon, where it has a magnitude that can be expressed as a number (i.e. the degree of a thermometer) and a reference (i.e. the temperature). Through the instances of this class we are able to represent the kind of values gathered by a sensor such as `temperature`, `humidity`, `wind speed`, etc. Quantities of the same kind (e.g. the values gathered by two thermometers) have the same quantity dimension. However, quantities of the same dimension are not necessarily of the same kind (e.g. sensors T_1 and T_2 in our running example gather the temperatures using two different units of measurement: `Celsius` and `Fahrenheit`). For this reason a sensor associated with a given theme (e.g. `temperature`) can be linked to an instance of the class `qu:Unit` for specifying the unit of measure of the detected values (e.g. `Celsius` for a `temperature`).

In order to model the event type of a sensor, the class `iot:Metadata` is used. Its instances model the data type of the entity whose thematic is observed. As an example, for the representation of a temperature value, an instance of the class `iot:Metadata` is linked to the instance of the sensor by means of a `hasMetadata` link. The instance of the class `iot:Metadata` is then linked to an instance of the class `DUL:Amount` for expressing its domain.

In some contexts of use, the events gathered by sensors are coupled with some measurement properties that characterize their thematic. Measurement properties (e.g. `accuracy`, `range`, `precision` of `ssn:Property`) identify observable characteristics of a sensor's events or ability to make observations. Specifically, these properties can refer to: *i*) the observed characteristics of the measurement or; *ii*) other information that can be used for a given kind of analysis (e.g. the number of tweets related to high temperature).

The first kind of properties can be modeled in the SSN ontology by means of the classes: `ssn:Accuracy`, `ssn:DetectionLimit`, `ssn:Frequency`, `ssn:Latency`, `ssn:MeasurementRange`, `ssn:Precision`, `ssn:Resolution`, `ssn:Drift`, `ssn:Sensitivity` and `ssn:Selectivity` $\sqsubseteq$ `ssn:MeasurementProperty`. The second kind of properties can be modeled by introducing new classes in the Domain Ontology.

Example 7: Figure 10 shows how sensors of type T_1 are semantically described by the class `ssn:SensingDevice` and whose thematic areas are modeled by the instance `temperature` of the class `qu:QuantityKind`. A sensor gathers temperature values in Celsius (instance of the class `qu:Unit`) and it is then linked to an instance of the class `iot:Metadata` for modeling the temperature data type. Moreover the sensor of type T_1 presents a measurement property for reporting the accuracy of the retrieved value (more or less 3 Celsius degrees), which is expressed by means of an instance of the class `ssn:Accuracy`. The instances of the class `qu:Unit` are used for specifying the unit of measurement (Celsius) of the accuracy.

In the Ontology we describe sensors of type T_1 after the application of operations of the Data Acquisition Plan as described in Section III-C.

As indicated in Figure 11 the sensors of type TW_1 deal with the Twitter thematic. This concept is modeled by using the `Tweets` instance of the class `qu:QuantityKind`. The data type associated to these sensors is a list of strings (tweets) modeled by an instance of the class `iot:Metadata`. The set of properties of the retrieved tweets, that is, the total number of tweets, and the number of positive tweets (i.e. those that contain the terms *hot*, *heat*, and *sweat*) is modeled by the instances of the subclasses `do:NumTweets` and `do:NumPosTweets` of a new class `do:TwitterProp`$\sqsubseteq$ `ssn:MeasurementProperty`. Two properties are linked

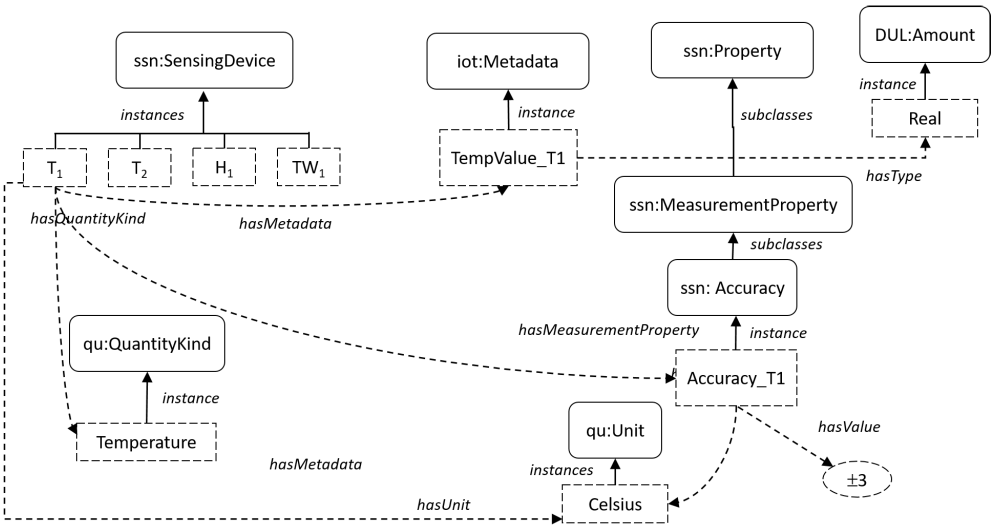


FIGURE 10. Thematic dimension of sensor of type T_1.

to the sensor of type TW_1 that are used for modeling the number of tweets and the number of positive tweets. In a similar way, we can introduce the instance `HumanDisc` for representing the thematic *Human Discomfort* and its properties required for the computation of the formula. □

In this Section we defined a Domain Ontology able to describe each STT dimension discussed before. In this phase we are able to semantically characterize the information produced by every sensor into the dimensions provided by our Syntactic Data Model in order to generate transformation rules that generate instances and links in the Domain Ontology for the representation of the sensor and of its schema.

V. EVENT STREAM CONSISTENCY W.R.T. DOMAIN ONTOLOGY

To guarantee that the event stream

$$\mathcal{M} = \langle G_T, G_S, th, [t_s, t_e], S, Event^{th}_{\langle G_T, G_S \rangle}(\tau) \rangle$$

gathered by a sensor specified according to Definition 2 is consistent w.r.t. a given Domain Ontology $\mathcal{O}$, we need to verify the exact match of its spatial, temporal and thematic dimensions with the related concepts expressed in $\mathcal{O}$. We remark that a Domain Ontology contains a set of classes and relationships considered valid by the experts of such domain.

For what concern the thematic dimension, this means that, in order to be consistent, the properties specified in the streams should be conceptualized as classes and links in advance in the Domain Ontology by the experts.

The general idea is to enable domain experts to include different kinds of constraints into the Domain Ontology in order to represent only concepts that are relative in their context of use. To do it, we can include restrictions on the mandatory occurrence of properties and on the verification of logical formulas on the Ontological instances. For instance, we can require that the unit of measure used by a thermometer should be present in the stream and that when more than one observation occurs, information about the accuracy of the gathered temperature is also reported. These restrictions have effects only on the consistency of the produced events.

At the ontology level, the existential restriction, denoted with ∃, can be used for this purpose with the meaning that 'some values from' (or 'at least one') respect the condition.

An existential restriction describes the class of instances that have at least one kind of relationship along a specified association to an instance of a specific class. For example, in order to impose the presence of the accuracy for each gathered temperature, the following constraint should be specified: ∀ instance of the class `ssn:SensingDevice` having an association `qu:QuantityKind` with an instance of the class `Temperature`, an association `ssn:MeasurementProperty` exists with an instance of the class `ssn:Accuracy`. In other words,the experts require that sensors of type T_1 for being consistent should have a measurement property `ssn:Accuracy`. Analogously, they can require that sensors of type TW_1 need to be bound with an instance of `TwitterProp` that contains the total number of tweets and positive ones.

A sensor is "not consistent" w.r.t. the Domain Ontology if it is not consistent with any of the STT dimensions according to the following definitions.

Definition 5: (Spatial Consistency). Given a sensor s instance of the class `ssn:SensingDevice` that generates a event stream $\mathcal{M}$, $\mathcal{M}$ is spatially consistent w.r.t. $\mathcal{DO}$ if:

1) G_S corresponds to an instance i_{sp} of the classes `geo:Point`, `iot-lite:Rectangle`, `iot-lite:Polygon`, or `iot-lite:Circle`, and an association exists between s and i_{sp}; or,
2) G_S corresponds to an instance i_s of the class `iot-lite:Entity` ⊑ `iot-lite:Coverage` at the granularity G_S. □

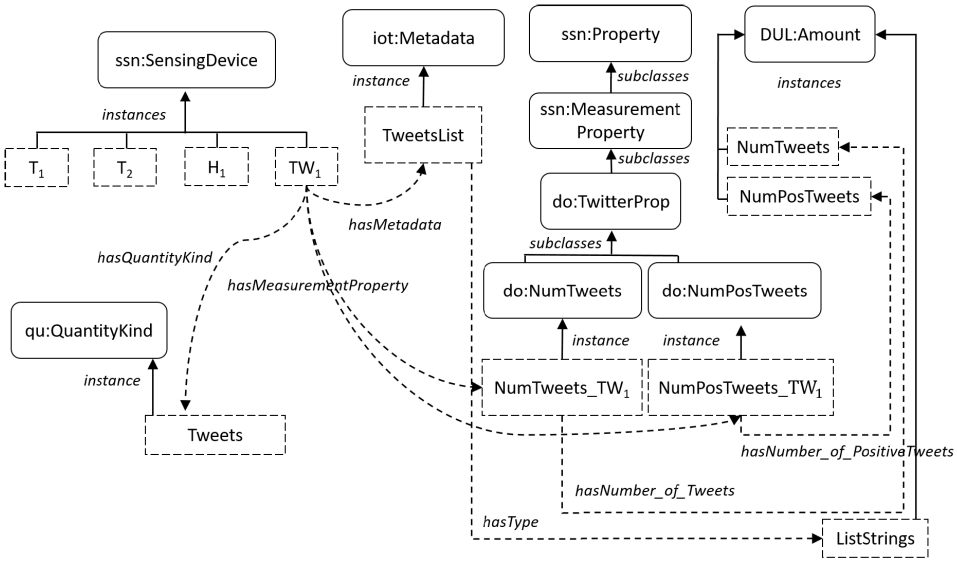


FIGURE 11. Thematic dimension of sensor of type TW_1.

Example 8: Consider the situation described in Example 4 and depicted in Figure 8. A sensor of type `TW1` is spatially consistent because its location is described by an instance of the class `iot-lite:Entity` that is a part of a city (Milano) thus fulfilling the relation of order defined by the granularity G_S. □

Definition 6: (Temporal Consistency). Given a sensor s instance of the class `ssn:SensingDevice` that generates a event stream $\mathcal{M}$, $\mathcal{M}$ is temporally consistent w.r.t. $\mathcal{DO}$ if:

1) G_T is specified in $\mathcal{M}$ and s is linked to a timestamp described by means of an instance of the class `tl:Instant` at the temporal granularity G_T and s is linked to an instance of the class `ssn:Response-Time` that produces the stream at the granularity G_T;
2) the time interval $[t_s, t_e]$ is specified in $\mathcal{M}$ and an instance of the class `tl:Proper Interval` exists in $\mathcal{DO}$ such that the interval described by the properties `Interval starts` and `Interval finishes` contains $[t_s, t_e]$;
3) G_T is specified through a link between the instance of the class `ssn:ResponseTime` with an instance of the class `qu:Unit` for specifying the unit of measure and by the subsequential link to the instance of the class `qu:SystemOfUnits`. □

Example 9: Consider the situation described in Example 5 and Example 6 and depicted in Figure 9. A sensor of type `T2` is temporally consistent because the timestamp of the data coming from the sensor is described by an instance of the class `tl:Instant`. Moreover, the sensor is also linked to an instance of the class `ssn:ResponseTime` that specifies that the temperature is acquired every 20 minutes. The instance of `qu:Unit` specifies the time granularity `Minute`. □

Definition 7: (Thematic Consistency). Given a sensor s instance of the class `ssn:SensingDevice` that generates a event stream $\mathcal{M}$, $\mathcal{M}$ is thematic consistent w.r.t. $\mathcal{DO}$ if:

1) s is linked to an instance i_{th} of the class `qu:QuantityKind` through the link `hasQuantityKind` that corresponds to th;
2) for each property p belonging to the data type τ associated to the sensor s, p is described by means of an instance of the class `iot:Metadata` which is in turn associated at an instance of the class `DUL:Amount` for expressing its domain, or p is linked to an instance of the class `ssn:MeasurementProperty` by means of an association `hasMeasurementProperty`. □

Example 10: Consider the situation depicted in Figure 10 and in Figure 11. `T1` and `TW1` are thematically consistent w.r.t. the adopted Domain Ontology because these instances are linked to instances of the `qu:QuantityKind` class for representing their thematic, they are linked to instances of the class `iot:Metadata` for modeling the data type and in turn are linked to instances of the class `DUL: Amount` for describing the corresponding domains. `T1` is also linked to the unit of measurement Celsius as foreseen in the $\mathcal{DO}$. Moreover, both sensors provide a complete semantic description of the properties as specified in the Domain Ontology. In fact, `T1` and `TW1` present the connections with proper measurement properties `ssn:Accuracy` and `do:NumTweets` and `do:NumPosTweets`. □

Finally, we can specify that an event stream is consistent only if it is temporally, spatially and thematically consistent according to the adopted Ontology.

Definition 8: (Consistent Specification). The event stream $\mathcal{M}$ is consistent if it is temporally, spatially and thematically consistent w.r.t. $\mathcal{DO}$. $\mathcal{M}$ is partially consistent if at least one of the dimensions is consistent. □

Example 11: Let $\mathcal{DO}$ be the Domain Ontology. Consider the situation described in Example 2, and their ontology characterization partially described in Examples 4,5, 6 and 7, and depicted in Figures 8, 9, 10 and 11. Sensors of type $\mathtt{T}_1$ and $\mathtt{TW}_1$ are spatially, temporally and thematic consistent w.r.t. $\mathcal{DO}$ because for each retrieved event, the streams report data about the position, the timestamp, the temperature plus the accuracy in case of $\mathtt{T}_1$ or the list of tweets and the related number of tweets and number of positive tweets in case of $\mathtt{TW}_1$, as specified by domain experts in the $\mathcal{DO}$. Sensors of type $\mathtt{T}_2$ are temporally consistent w.r.t. $\mathcal{O}$ but they are not spatially and thematic consistent because spatial coordinates are missing and the thematic dimension does not report information about the unit of measurement and the accuracy. Finally, Sensors of type H_1 are thematic and spatially consistent w.r.t. $\mathcal{DO}$, but temporally inconsistent because temporal coordinates are missing. □

This definition of consistency allows the generation of a mapping between the temporal and spatial dimensions adopted in the event streams and the Domain Ontology and also of a mapping between the properties specified for a given thematic and the concepts of the Domain Ontology. These mappings are then exploited in the Semantic Virtualization process of the physical and social sensors for transforming the events from the sensor specific formats to an internal representation. This internal representation, when is consistent w.r.t. the Domain Ontology, allows interoperability among the events generated by sensors of different IoT platforms.

VI. VERIFICATION OF DAPS CONSISTENCY

In the previous sections we presented the data acquisition services and we identified an approach for verifying the consistency of their output w.r.t. the adopted Domain Ontology.

A Data Acquisition Plan is a composition of such services and can be represented as a graph that specifies the flow of operations on sensor data. This representation helps the user on composing services in order to transform, aggregate, filter and store information. According to the Definition 4, a Data Acquisition Plan is considered *sound* if: *i*) the number of input streams is equal to `inputS`; *ii*) the required parameters are specified; and *iii*) its conditions `conds` can be evaluated on the input stream *s* and are verified.

We are now able to acquire streams for which someone of the Spatio-Temporal-Thematic dimensions are not specified or are specified but are not consistent according to Definition 8. This is particular useful in order to flexibly adapt to different situations and to post-pone the inclusion of the STT dimensions according to extra knowledge that is present in the Domain Ontology or in other sources. However, we provide here the notion of a consistent Data Acquisition Plan which is a sound plan where the ontology O and the event model associated with the destination node of the graph are consistent according to Definition 8.

Definition 9: (A Consistent Data Acquisition Plan). Let $G = (V, E)$ be a sound Data Acquisition Plan according to Definition 4, $\mathcal{M}$ and O be the event model and ontology generated for the destination node. G is consistent if: $G_T \neq \perp$, $G_S \neq \perp$, and $\mathcal{M}$ is consistent w.r.t. $\mathcal{O}$. □

This definition allows the applications of operations that are not consistent w.r.t. the Domain Ontology, but can produce final results that are consistent with it. Therefore, the data acquisition services can change the semantics of the data and make them compliant with the adopted Ontology.

Example 12: The Data Acquisition Plan described in example represented on Figure 6 is sound according to Definition 4. Moreover, the event type of the last instruction (*hd*) is consistent w.r.t. our Domain Ontology. □

Internal nodes of a Data Acquisition Plan can thus be not consistent (or only partially consistent) w.r.t. the Domain Ontology. This allows the generation of a flexible tool that is able to handle heterogeneous streams that do not perfectly match the constraints imposed by the Domain Ontology.

In the remainder of the section we first introduce the operations required for the description of the event schema produced by each data acquisition service at the ontological level. Then, an algorithm is presented that starting from a sound Data Acquisition Plan, populates the instances of the Domain Ontology for its description, and determines whether its output is consistent.

A. AUXILIARY SENSORS IN THE DOMAIN ONTOLOGY

Once a Data Acquisition Plan is considered sound (according to Definition 4), we wish yo check its consistency w.r.t. the Domain Ontology (Definition 8).

With this aim, at the Ontological level we need to describe each service (that can be applied to the streams produced by the sensors) as an *auxiliary sensor*, instance of the class `ssn:SensingDevice`. An auxiliary sensor receives one or more incoming streams and produces a single stream whose schema, that is compliant with the STT model, can be made in correspondence with the concepts and relationships available in the Domain Ontology. Also for these auxiliary sensors we can check the consistency w.r.t. the Domain Ontology.

Example 13: Consider the situation depicted in Figure 10 and in Figure 11, $\mathtt{T}_1^1$ and $\mathtt{TW}_1^2$, described in Example 3, present ontological schema similar to those depicted in the figures. In this case, the two sensors are examples of auxiliary sensors generated as result of the Data Acquisition Plan, which properties depend on the applied operator and on the streams of the incoming sensors (either physical or social). Instances are included in our Domain Ontology to represent such data acquisition services and then check the consistency on the resulting Ontology. □

An auxiliary sensor is obtained in two steps. First, a new sensor *j* is generated (either by cloning the incoming sensor *i* or by generating a new one). Then, the new instance and its links are modified or other links are added in order to comply to the operator specification. These operations are specified by means of the simple primitives reported in Table 3. Starting from an instance *i* of the class `ssn:SensingDevice`,

TABLE 3. Primitives for updating Ontology instances.

Operation	Meaning
i=clone(j)	Create a copy of the instance j with all the links that involve j
i=new("CLASS")	Create a new instance of the class CLASS
i.addLink(j,"REL")	Include a link between i and j instance of the relationship REL
i.delLink(j,"REL")	Remove the link between i and j instance of the relationship REL

these operations introduce new instances and links in the Ontology (when the conceptualization allows it).

The only exception to this general rule is the treatment of the filter and trigger services. In the first case, we simply need to clone the incoming sensor and no further operations are required, while in the other case we need to clone and check if a trigger condition is verified. In both cases the schema of the incoming sensor is the same produced as output.

By contrast, for the other services (transform, enrich, virtual property, aggregation, union, join and convert) the operations reported in Table 4 need to be applied.

For the services enrich and virtual property, new instances of the class a are added in the ontology $\mathcal{O}$ in order to model the new properties that are inherited by other sensors (by using the enrich service) or created as new virtual properties (by using the virtual property service). The service transform enables to change the units of measurement of the sensor properties a according to the function *trans* or it applies a transformation by executing a specific code. The service aggregation creates a new auxiliary sensor whose values are gathered according to a new time value defined by a new instance of the class ssn:ResponseTime with a new unit of measure. The service join creates a new auxiliary sensor whose properties are the union of the properties of the two input sensors $\mathtt{a} \in \mathcal{P}rop(s_1) \cup \mathcal{P}rop(s_2)$ whereas the service union allows us to union the events produced by different sensors and generates a new thematic (instance of the class qu : QuantityKind) and collect all properties of the input sensors. Finally, the service convert simply changes the temporal or spatial dimension according to the target granularity specified in input.

Example 14: Example 3 presents two auxiliary sensors $\mathtt{T}_1^1$ and $\mathtt{TW}_1^2$ generated by sensors of type $\mathtt{T}_1$ and $\mathtt{TW}_1$. $\mathtt{T}_1^1$ is created by applying an enrich service. As result of the application of this service, the instance $\mathtt{T}_1$ is cloned in $\mathtt{T}_1^1$ and a new link hasMeasurementProperty is added for connecting the auxiliary sensor to an instance of the class ssn : Accuracy for modeling the accuracy property. In the same way, $\mathtt{TW}_1^2$ is cloned by $\mathtt{TW}_1$ as result of the application of a virtualProperty service. Two new links hasMeasurementProperty are added for connecting the new auxiliary sensor to new instances of the class do: NumTweets and do:NumPosTweets used for reporting the number of tweets and positive tweets.

Now, consider the situation described in Example 3. The final formula used for calculating the human discomfort is based on the creation of the auxiliary sensor $\mathtt{HD}_1^3$ which thematic HumanDisc is an instance of the class qu:QuantityKind. The auxiliary sensor is in turn linked to instances of the class iot:Metadata for modeling the associated data type temperatureVal, humidityVal, TweetsList and hd, and to instances of the class ssn:MeasurementProperty for modeling the measurement properties that characterize the thematic (i.e. accuracy, numTweets, numPos and lhd). Once the final auxiliary sensor $\mathtt{HD}_1^3$ is generated, its consistency w.r.t Definition 8 is checked. As result we can say that the sensor of type $\mathtt{HD}_1^3$ is consistent w.r.t. the Domain Ontology $\mathcal{O}$ that we have described in the previous examples. □

B. A CONSISTENT DATA ACQUISITION PLAN

Starting from the operations reported in Table 3 for the representation of each single service at the ontological level, we here discuss the algorithm for populating the instances of the Domain Ontology with the information about a Data Acquisition Plan and for verifying whether it is also consistent.

Given a sound Data Acquisition Plan $DAP = (V, E)$ and a Domain Ontology O, we need to create a binding between the schema of the sensors occurring in DAP and the concepts of O. The user carries out this activity by means of a Web tool for specifying a mapping between the properties of the sensor schema and the concepts of the Ontology (details of the GUI are in Section VII).

Once the sensors have been mapped to the Domain Ontology, the Data Acquisition Plan is visited in post-order starting from the destination node, and for each service the operations reported in Table 4 are applied to provide their representation at the ontological level. These activities are automatically executed and do not require any human interaction and lead to the introduction of the ontology instances for representing the Data Acquisition Plan. The user can check the consistency of each service by means of the graphical interface. At this point it is possible to check the consistency of the entire Data Acquisition Plan DAP according to the following definition.

Definition 10: (A Consistent Data Acquisition Plan). Let $DAP = (V, E)$ be a sound Data Acquisition Plan according to Definition 4, $\mathcal{M}$ and O be the event model and Ontology generated for the destination node. DAP is consistent if: $G_T \neq \perp$, $G_S \neq \perp$, and $\mathcal{M}$ is consistent w.r.t. $\mathcal{O}$. □

Example 15: Figure 12 reports the graph representation of the Data Acquisition Plan discussed in Example 3 annotated with the consistency of the initial flows produced by the sensors and of the streams produced by the application of the internal services. The triple (Y/N, Y/N, Y/N) is used to denote whether the stream is spatially, temporally, and thematically consistent w.r.t. the adopted Domain Ontology. The triple is reported within the rounded rectangles denoting services for representing that the consistency of the stream produced by the service. Note that, in this case, even if some services are not consistent, the entire DAP is consistent

TABLE 4. Instructions for modifying the Ontology Instances according to the service.

Service(Symbol)	InsOntology

Enrich – $\propto_{pred}^{KB} s$	j=clone(i)
	for $a \in \mathcal{P}rop(KB) \setminus \mathcal{P}rop(s)$ **do**
	j.addLink(new("a"), "t") (where t is the type of a)
	end for

Virtual property – $\uplus_s \langle p, spec \rangle$	j=clone(i) Let t be the type of p j.addLink(new("p"), "t")

Transform – $\Diamond_{trans}^{\{a_1,\ldots,a_n\}} s$	j=clone(i)
	for $a \in \{a_1, \ldots, a_n\}$ **do**
	if $trans$ = CHANGE_UNIT **then**
	Consider $i_a \in \mathcal{I}$(qu : Unit) corresponding to a for instance i
	j.delLink(i_a, "hasUnit")
	j.addLink(new("a"), "hasUnit")
	end if
	if $trans = X$ **then**
	exec specific code for the function X
	end if
	end for

Aggregation – $@_{op}^{t,\{a_1,\ldots,a_n\}}(s)$	j=clone(i) Let $i_t \in \mathcal{I}$(ssn : ResponseTime) s.t. a link $(i$, hasMeasurementProperty, $i_t)$ exists j.delLink(i_t, "hasMeasurementProperty") j_t = new("ssn : ResponseTime") j.addLink(j_t, "hasMeasurementProperty") Consider $j_u \in \mathcal{I}$(qu : Unit) corresponding to the temporal granularity of t j_t.addLink(j_u, "hasUnit")

Join – $s_1 \bowtie_{pred}^{t,th'} s_2$	j = new("ssn : SensingDevice")
	$j_{th'}$ = new("qu : QuantityKind")
	$j_{th'}$.namedIndividual = th'
	j.addLink($j_{th'}$, "hasQuantityKind")
	for $a \in \mathcal{P}rop(s_1)$ **do**
	Let $(i_1$, "t", $i_a)$ be a link to the instance of class a
	j.addLink(j_a, "t")
	end for
	for $a \in \mathcal{P}rop(s_2) \setminus \{b_1, \ldots, b_n\}$ **do**
	Let $(i_2$, "t", $i_a)$ be a link to the instance of class a
	j.addLink(j_a, "t")
	end for

Union – $\cup^{t,th'}(\{s_1, s_2\})$	j = new("ssn : SensingDevice")
	$j_{th'}$ = new("qu : QuantityKind")
	$j_{th'}$.namedIndividual = th'
	j.addLink($j_{th'}$, "hasQuantityKind")
	for $k \in \{1, \ldots, n\}$ **do**
	for $a \in \mathcal{P}rop(s_k)$ **do**
	Let $(i_k$, "t", $i_a)$ be a link to the instance of class a
	j.addLink(j_a, "t")
	end for
	end for

Convert – $\ominus_{tg,sg}^{t}(s, cf)$	j=clone(i)
	if $tg \neq Gran_T(s)$ **then**
	Let $(i$, "hasMeasurementProperty", $i_t)$ be a link s.t. $i_t \in \mathcal{I}$(ssn : ResponseTime)
	j.delLink(i_t, "hasMeasurementProperty")
	j_t = new("ssn : ResponseTime")
	j.addLink(j_t, "hasMeasurementProperty")
	Consider $j_u \in \mathcal{I}$(qu : Unit) corresponding to the temporal granularity of tg
	j_t.addLink(j_u, "hasUnit")
	end if
	if $sg \neq Gran_S(s)$ **then**
	Let $(i$, "hasLocation", $i_s)$ be a link s.t. $i_s \in \mathcal{I}$(iot-lite : Entity) or its subclasses
	j.delLink(i_s, "hasLocation")
	j_s = new("ssn : hasLocation")
	j.addLink(j_s, "hasLocation")
	Consider $j_u \in \mathcal{I}$(qu : Unit) corresponding to the spatial granularity of sg
	j_t.addLink(j_u, "hasUnit")
	end if

because the last stream (*hd*) is consistent w.r.t. our Domain Ontology. □

Starting from the consistency of the sensors adopted in a Data Acquisition Plan, it is possible to determine the consistency of its services and of the produced final stream without the need to verify the consistency conditions as specified by the following lemma.

Lemma 1: Let $\mathcal{M}_{b_1}, \ldots, \mathcal{M}_{b_n}$ be the event data models of the incoming streams to a service *op* and $\mathcal{O}_b$ be the ontology before the application of *op*. Let $\mathcal{M}_a$ be an event data model and $\mathcal{O}_a$ be the ontology after the application of *op*. The following statements hold:

- if op = filter, then $n = 1$ and $\mathcal{O}_a = \mathcal{O}_b$, and the consistency w.r.t $\mathcal{O}_a$ is the same of $\mathcal{O}_b$.

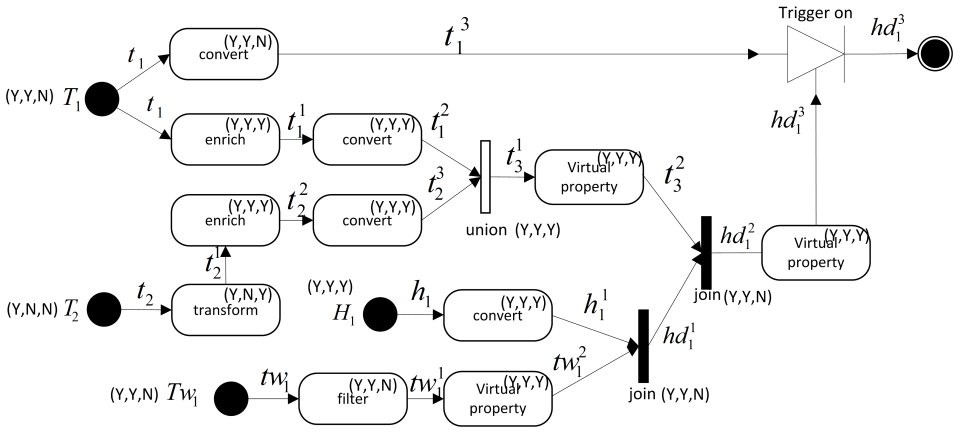


FIGURE 12. Evaluation of consistency of the running example's data acquisition plan.

- if $op \in \{\texttt{enrich}, \texttt{virtualProperty}\}$ and $\mathcal{M}_{b_1}$ is consistent w.r.t $\mathcal{O}_b$, then $\mathcal{M}_a$ loses the thematic consistency w.r.t. $\mathcal{O}_a$.
- if $op = \texttt{convert}$ and the thematic of s is t, the consistency w.r.t $\mathcal{O}_a$ is the same of $\mathcal{O}_b$.
- if $op = \texttt{union}$ and the thematics of s_1 and s_2 are th', the consistency w.r.t $\mathcal{O}_a$ is the same of $\mathcal{O}_b$. □

This lemma can be easily demonstrated for induction by taking into account the specification of each single data acquisition service.

VII. THE StreamLoader PROTOTYPE

The facilities so far described have been implemented in the context of the StreamLoader system. In [27] we have shown the algorithm for translating the obtained DAP in Spark streaming and reported some experiments on the scalability and efficiency of the obtained script. This system is specifically tailored for domain experts that need to develop different kinds of analysis on streams of events produced by sensors belonging to cross-domain platforms. Since domain experts are usually not computer experts, specific attentions have been devoted to create easy to use interfaces that support them in monitoring and manipulating the flow of events coming from heterogeneous sensors by means of Data Acquisition Plans that easily adapt to their mental model. In the design of the interface we adopted the participation design principles [28], [29] that require to involve users with different backgrounds (in our case meteorologists, computer scientists, sociologists) to identify the critical components to be included in the graphical interfaces for the creation of sound and complete Data Acquisition Plans. The key concept is to involve domain experts in activities for mapping and translating their professional knowledge into proper vocabularies, notations, and suitable visual structures of navigation among interactive systems interface elements. In our context of use, the purpose is to obtain a system that allows the domain experts to focus on the analysis of data-flows rather than on technical details related to the configuration of software and hardware components and on the development of code.

By following these principles a group of 3 independent people have been enrolled with whom we organized different meeting for the design and development of the StreamLoader interface.

The main goal of these meetings was to evaluate the validity-in-practice of the systems features in order to extend, amend, or eventually recommend improvements. In the remainder of the section we discuss the user interfaces obtained through this work for the Semantic Virtualization of the sensors and for the verification of the consistency of the Data Acquisition Plans.

A. SEMANTIC VIRTUALIZATION

The GUI interface supports the user in the Semantic Virtualization of a sensor. This service is particularly important because it handles different formats according to which each sensor produces events and for mapping its schema to the concepts of the Ontology.

We remark that our flexible model allows us to handle also streams that do not completely adhere to STT organization of the information. Specifically, the GUI supports users to manually establish relations between the sensor schema and the classes of the Domain Ontology. The web mapping editor offers three distinctive features:

- a function for reading and visually presenting the sensor schema regardless of their formats (e.g. JSON, CVS, or XML);
- a point-and-click interface for reducing the mapping activities efforts;
- an Ontology-driven mapping approach, where the user can link each entry of the sensor schema to an instance of the ontology class. The user can select an existing instance, by means of a drop-down menu, or can create a new instance.

The user-defined mapping provides for each sensor a semantic description at the Ontological level through which its consistency can be evaluated according to Definition 8. At this stage, the user can describe at a semantic level the data

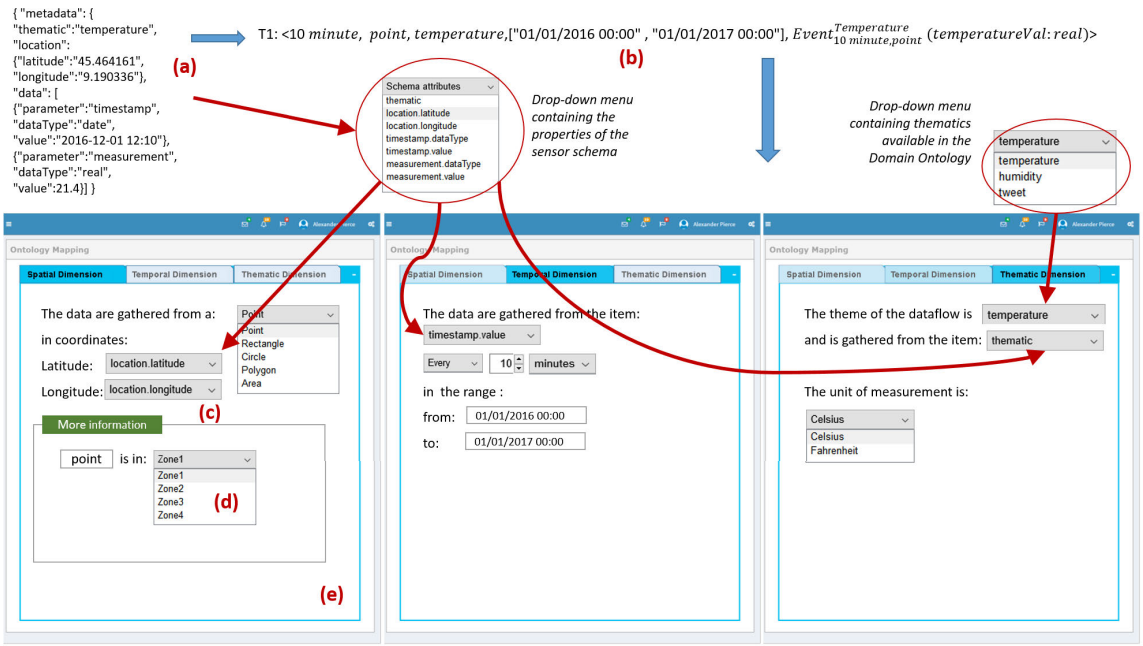


FIGURE 13. StreamLoader GUI used for mapping the spatial dimension of the event stream of the sensor of type T_1 to the corresponding concepts in the domain ontology.

coming from a sensor and can map its attributes to the concepts of the Domain Ontology. For example, if the sensor provides a simple timestamp in a traditional format "DD/MM/YY HH:MM:SS:MS" or in any other format, by using the interface in Figure 13, the user detects the attribute and maps it to the proper class retrieved by the ontology.

Example 16: Figure 13 shows a screen-shot of the GUI used for mapping the STT dimensions of a sensor schema to the instances of the ontology classes. Specifically in Figure 13 a), the JSON representation of the sensor schema of type T_1 (whose event type is depicted in Figure 13 b) is mapped to instances of our Domain Ontology. In Figure 13 e), the user has specified the mapping about the spatial dimension, the result of which corresponds to the one presented in Figure 8. The interface is dynamically created relying on the information gathered from the sensor schema (e.g. as reported in Figure 13 c)) or from the Ontology (e.g. see Figure 13 d)). □

When the virtualization process is concluded the service is included in a publish-subscribe system and made available to be used for the specification and monitoring of new Data Acquisition Plans, as discussed in the remainder. Moreover, the provided mapping is stored and considered when other sensors of the same type need to be semantically virtualized. In this way, we simplify this process.

B. DATA ACQUISITION PLAN SPECIFICATION

As shown in Figure 14 the interface for the Data Acquisition Plan specification is composed of a left sidebar menu. The menu offers the following functionalities: *project*, for the management of the data acquisition projects (create, delete, save, and open projects); *source*, for selecting and discovering sensors from which event streams can be acquired; *service*, for selecting one of the services in Table 1; *destination* for specifying the repository where the resulting stream should be stored. By clicking on each of them, a modal menu appears through which it is possible to select icons to be included in the main canvas. Moreover, connections can be drawn in the canvas among the services, the sensors and the destination nodes.

When the icon is placed in the canvas its border is colored in red to represent the fact that the user needs to specify parameters to obtain a sound specification. Once the parameters are specified, the associated conditions are evaluated and, when it is sound, the border of the icon is colored in blue. The border color becomes green when also the consistency constraints are met.

At the bottom of the canvas, a horizontal tab menu area appears when icons (e.g. services or connections) are double-clicked. Relying on the type of element, it shows different buttons and information and is used to define the name of the services, the type of sensors, the filtering conditions, and the trigger condition. It also provides the event types that are processed and the conditions for being sound and consistent. The consistency of the STT dimensions of each node is provided by the 3 icons (a point for the spatial dimension, a clock for the temporal dimension, a tag for the thematic dimension) on the right hand side of the horizontal tab menu header. When the consistency is met, the icon is colored green, otherwise is colored red. The consistency of the entire Data Acquisition Plan is shown in the top right border of Figure 14 by means of a traffic light (green is consistent, yellow is partially consistent, red is not consistent).

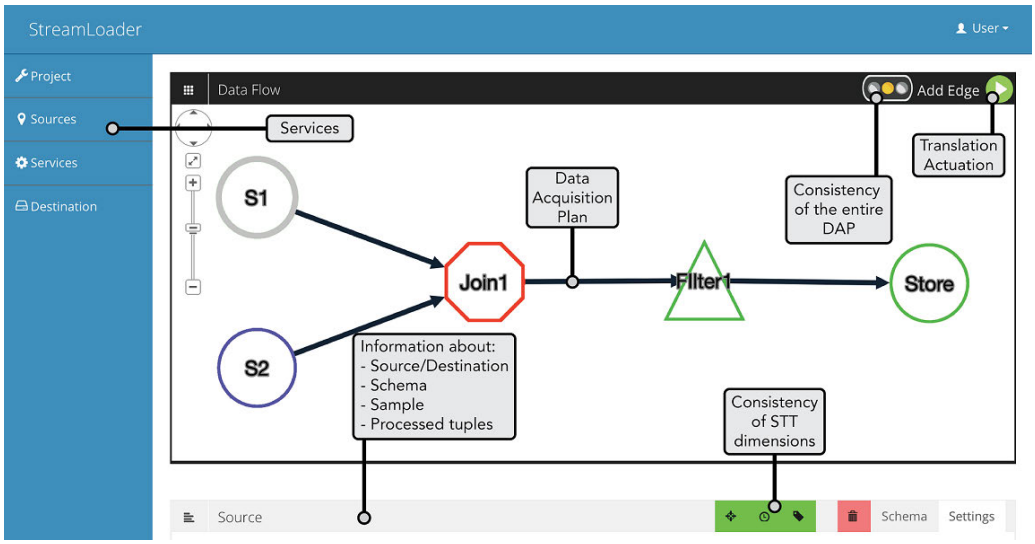


FIGURE 14. Main screen of the web application.

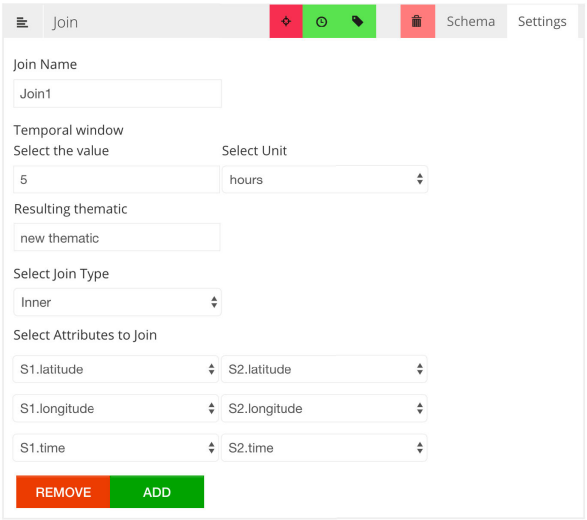


FIGURE 15. Join tab menu.

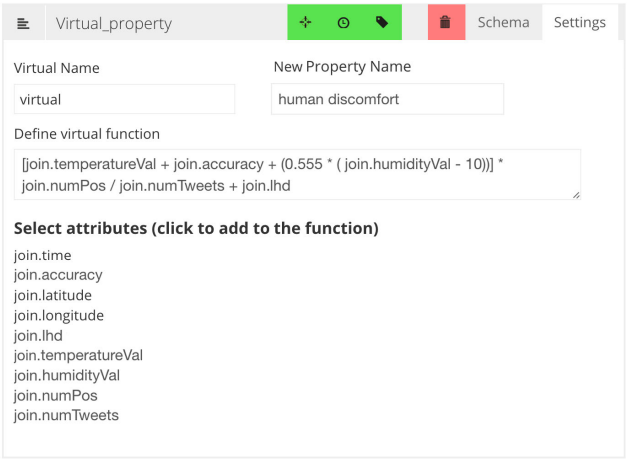


FIGURE 16. Virtual property tab menu.

As specified in Section III-B several services are provided by the interface for processing and combining the stream produced by the sensors. During the creation of the Data Acquisition Plan, the interface implements the control of soundness and shows the identified errors. These messages support the user in the development of sound and consistent Data Acquisition Plans. In the remainder we describe the icons and the horizontal tab menus for the `Join` and `virtualProperty` services. The `Join` service corresponds to the `SQL JOIN` and the associated icon has two incoming edges and a single outgoing edge. It easily allows the join between the different attributes of the schema coming from the ingoing streams (see for example the join service named `join1` in Figure 14). By clicking on the icon the horizontal tab menu in Figure 15 is shown in the bottom of the canvas that consists of two sections: Settings and Schema. The former contains the name associated with the join service, the type of join (inner, left, right, full join), the temporal windows according to which the join is evaluated, the generated thematic, and the join predicate. By means of the `add` and `remove` buttons an user can add/remove conditions to the join predicate.

The latter provides the stream event type that is produced as the result of the join execution. We remark that the domain type of the join result is composed by the union of the properties of the incoming streams (even if the user can decide which properties should be made available to the outgoing service). Property names are disambiguated relying on the names of the incoming services (when needed).

The `virtualProperty` service allows the user to create a new property based on arithmetic expression.

As shown in Figure 16 at the bottom of horizontal tab (*virtualProperty tab*) a list of all the properties (from all the ingoing sensors) is provided. In a text area the user can create a function using the mathematical operators (+, -, *, /), using

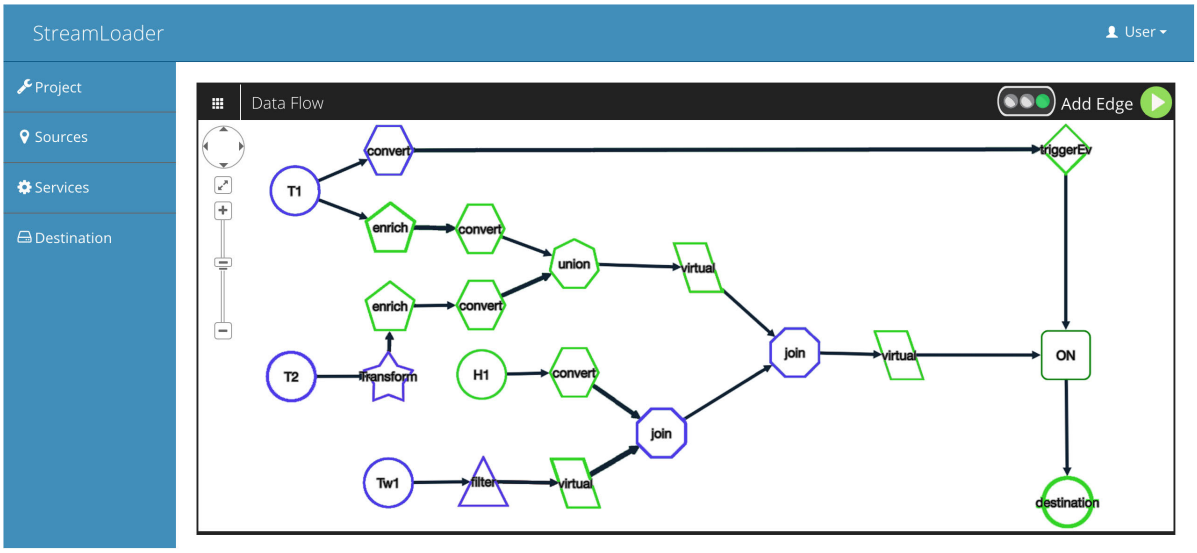


FIGURE 17. Data Acquisition Plan for the running example realized with the provided GUI.

brackets and by adding the properties of the incoming streams by just clicking on the property names shown in the list. Before applying the calculation, the user is asked to give a name to the new (virtual) property. The function calculated in the text area is the Human Discomfort formula of our motivating scenario (see Section II-A).

The second tab (*Schema tab*) is used to show all the event type that is generated by the service. In this tab the user can check whether the included virtual property is also described at the Ontological level.

Example 17: Figure 17 shows the Data Acquisition Plan that is created for our running example and corresponds to the plan of Example 3. Note that the shapes of the icons are colored in blue or green depending on the consistency of the operator w.r.t. the Domain Ontology and that the traffic light in the top right corner is green because the Data Acquisition Plan is consistent w.r.t. Definition 10. □

VIII. EXPERIMENTAL VALIDATION

We conducted a set of experiments for evaluating the impact of the use of an Ontology in the validation activities. Since the DAPs are translated in Spark Streaming scripts we have considered their executions locally on a single machine and on clusters of 5 nodes. We use virtual machines that reside on an **Ovirt** Datacenter (`https://www.ovirt.org/`) that are equipped with Ubuntu 16.04 LTS (GNU/Linux 4.4.0-96-generic x86_64), 8GB RAM, 2 core processor, 250 GB HDD and 2799.202 MHz CPU clock speed. In each cluster, a node acts as master and the others as slaves.

The first experiment was devoted to investigate the effects of validation w.r.t. the size of the Domain Ontology. Indeed, during validation new instances are introduced in the Domain Ontology and we have measure the time required for checking the validity of the DAP that takes into account the time required for introducing in the Domain Ontology the instances representing the virtual sensors. In the second experiment we have exploited the semantic characterization of sensors for specifying translation rules that allow us to map the sensor data available in a specific IoT platform in our data model (based on JSON) that is semantically labeled with the concepts of the Domain Ontology. Finally, we have considered some scenarios and tried to model the corresponding Domain Ontology. Then, some DAPs have been realized and their consistency checked w.r.t. the developed Ontology. In the remainder of the section we discuss such experiments.

A. EXECUTION TIME FOR VALIDATION AND EFFECT ON THE SIZE OF THE DOMAIN ONTOLOGY

By considering the Domain Ontology that we have used throughout the paper, we have considered a set of DAPs of different sizes and calculate the average time for checking their validity w.r.t. the Domain Ontology. The required time is reasonable for a web application and grows linearly with the side of the DAP. Moreover, since new instances are included in the Domain Ontology we have evaluated the size of the Ontology (in terms of bytes) for the considered DAPs. The size of the Ontology increase of an average of 600 byte for each virtual sensor present in the DAP. For the DAP with 50 nodes, the Ontology size has increased of 30 KB.

B. APPLICATION OF TRANSLATION RULES FOR THE CONVERSION OF SENSOR DATA INTO THE STT DATA MODEL ANNOTATED WITH SEMANTIC CONCEPTS

When data produced by sensors of a given platform do not adhere to our STT data model, a transformation process should be applied that takes into account the Semantic Virtualization described in Section VII-A. This process can be executed *producer-side (PS)*, i.e. in the environment of the platform before sending the data to our platform, or *consumer-side (CS)*, i.e. data are transmitted through the Kafka client in the same format as they are generated and then translated in the internal format directly on the Apache

Spark cluster. A set of different experiments have been conducted in order to understand how the system behaves with the introduction of the transformation phase. The data set for this experiments have been downloaded in csv format from the *(ARPA)* (the Lombardy Regional Meteorological Agency).[7] Specifically, we get temperature and humidity data from sensors in Milan.

The DAPs contain both non-blocking and blocking services at increasing complexity. The first one (named `store`) is a simple DAP that reads data from a sensor and store the obtained value in a file every 3 sec. The second one (named `aggregation`) applies an aggregation on the considered events and the aggregated events are stored in the file every 6 sec. The last one (named `join`) computes the join between two streams every 6 sec. and is the most complicated experiment because tuples from different sources need to be collected and different transformation should be applied before performing the Join according to the specific time window. For the first two DAPs we have considered a single flow of 10 million events, whereas for the last one we have considered two flows of 10 millions events each. In our tests the Apache Kafka Server runs in one of the worker of our cluster of machines. In the `store` and `aggregate` experiments the Kafka clients run on the same machine of the Kafka Server while in the `join` experiment, we need to have two different Kafka Producers that generate data on two different topics. For this reason one Kafka Producer runs on the same machine of the Apache Server and the second one runs on one of the cluster worker.

These experiments aim at discovering the cost of performing the transformation before or after the transmission of the data. By means of the experimental activity we can point out some interesting observations. Before discussing the results on Processing Time, Scheduling Delay and Total Delay of each test we need to give a look to the diagram on Figure 18. This diagram reports the number of tuples that are transmitted every second in the PS and CS environments by Apache Kafka. From the diagram is quite evident that in our setting the application of transformations ''producer'' side has negative effectiveness on the number of records produced every second. The average rate in the PS approach is around 24.000 tuples while in the CS we have an average size of 55.000 records. This means that the CS approach generates more than the double of tuples per second. This is probably caused by the introduction of computation in order to transform the raw data into the internal data model.

For what concern the `join` experiment, the average number of data per second is exactly the double in both approaches: 48.000 tuples per second for the PS approach and 110.000 tuples per second in the CS approach.

1) PROCESSING TIME

The Processing Time is the time required to compute all the jobs of a given batch of data, end to end.

[7]http://www.arpalombardia.it/siti/arpalombardia/meteo/richiesta-dati-misurati/Pagine/RichiestaDatiMisurati.aspx

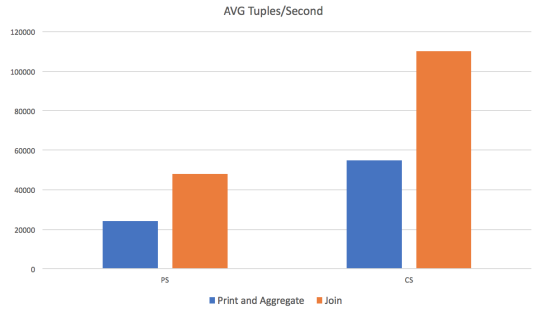
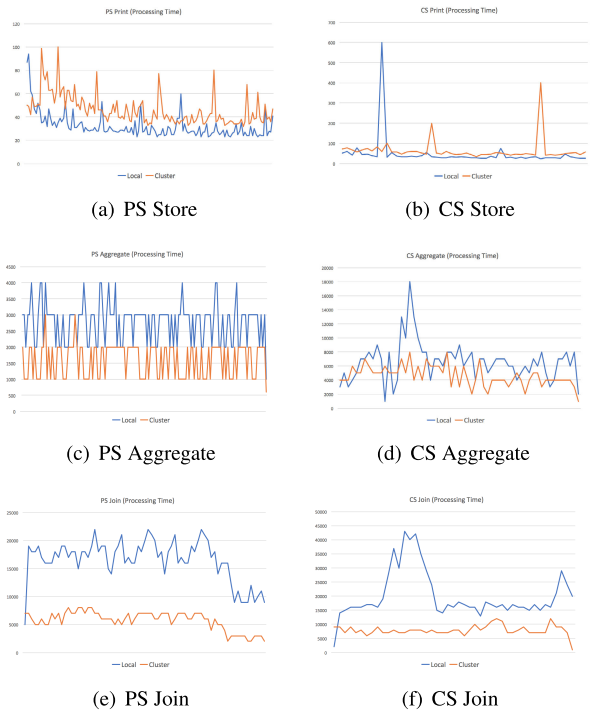


FIGURE 18. Average number of generated tuples per second.



(a) PS Store (b) CS Store

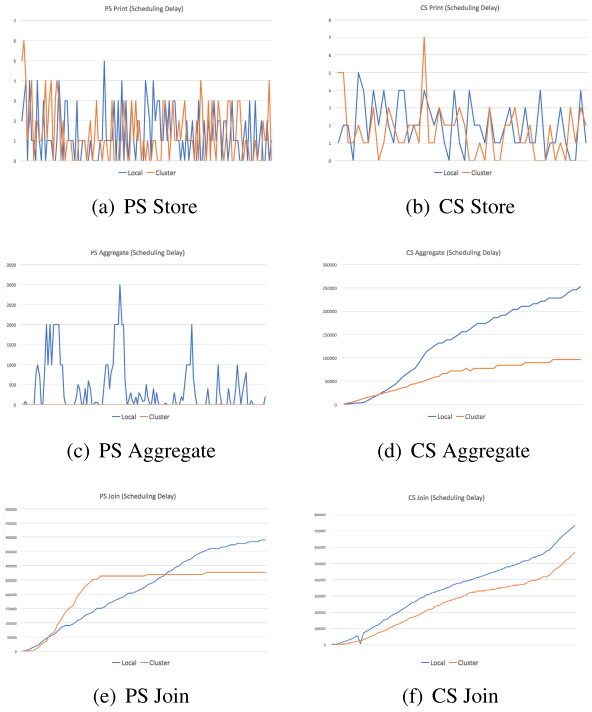
(c) PS Aggregate (d) CS Aggregate

(e) PS Join (f) CS Join

FIGURE 19. Processing Times of the considered DAPs.

Figures 19(a) and 19(b) show the performance of the PS and CS approach for the `store` experiment. The interesting result is that the cluster execution has, in both approaches, worse performance than the local execution. Locally, the Processing Time lasts 32 ms in the PS approach and 47 ms in the CS, while in the cluster execution the Processing Times are 46 ms for PS and 63 ms for CS. Spark script executes a simple operation. The time it takes to route the execution of the operation through the machines of the cluster worsens the performances. In this case we can see that in the CS execution the Processing Time are higher than the PS, but if we take the batch size presented in Figure 18 into account, we can see that the amount of data to process is more than the double.

In the `aggregate` and `join` operations (blocking operators) we have better Processing Times. If, in case of aggregation (Figure 19(c) and Figure 19(d)) we have an improvement of more than a second (from 2,8 to 1,5 sec. in the PS approach and form 6,6 to 4,6 sec. in the CS approach), in case of join (Figure 19(e) and Figure 19(f)) we have a remarkable



(a) PS Store (b) CS Store

(c) PS Aggregate (d) CS Aggregate

(e) PS Join (f) CS Join

FIGURE 20. Scheduling Delay of the considered DAPs.

improvement (from 16,5 to 5,70 sec. in the PS approach and from 20,1 to 7,9 sec. in the CS approach).

2) SCHEDULING DELAY

In the `store` experiments (Figures 20(a) and 20(b)) there are no remarkable differences both for PS and CS approaches in local and cluster execution. The average Scheduling Delay is around 1,5 ms for the PS approach and 1,8 ms for the CS approach. The main differences are shown in the other two tests. In case of aggregation, Scheduling Delay in the PS approach is around 405 ms locally and 1,5 ms in the cluster while in the CS approach the local Scheduling Delay is more than 2 min. and the local is close to 1 min. If we take a look at the graph represented in Figures 20(c) and 20(d) we can see how the trend of the Scheduling Delay in the PS approach is similar to a constant in the cluster case and in the local case is more floating, while in the CS approach the graph tends to grow constantly, especially in the local case.

The interesting point is given with the execution of the `join` experiments. As shown in Figures 20(e) and 20(f), the Scheduling Delay of the local execution in the PS is smaller that the cluster execution in the CS approach (the highest value in our experiments are 390 sec. for the PS local approach and 580 seconds for the CS cluster approach). Another feature to notice is that in the cluster execution of the PS approach the Scheduling Delay is almost constant while in all the other cases the Scheduling Delay is represented as an increasing function.

3) TOTAL DELAY

Figure 21 provides details about the Total Delay for every experiments we conducted. Total Delay is composed of

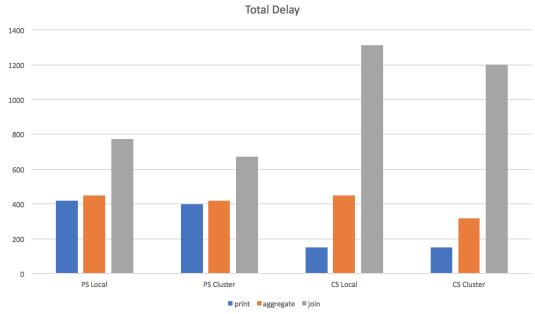


FIGURE 21. Total Delay for every experiment and for the approaches.

Processing Time and Scheduling Delay and as we can see the `store` experiment gives the best results in the CS approach by taking more the half of the time of the PS approach either in the local and cluster mode. For what concern the `aggregation` Total Delay is quite similar to every experiment we have conducted. The local execution takes the same time in both approaches (7 min. and 30 sec.) while in the cluster execution there is more than 1 min. of improvement of CS approach w.r.t. the PS one.

The `join` experiment is the one that introduces the main differences between the two approaches. If we consider the difference of the local and cluster execution, in both approaches the cluster execution gives the better results. For what concern the difference between the PS and CS approach we can see that also the local execution of the join is sensibly better than the execution in cluster mode of the CS approach (770 sec. compared to 1200 sec.). The best performances are the ones provided by the execution in a Cluster with the PS approach. It takes 670 sec. to join 20 million tuples.

C. MODELING OF ONTOLOGIES FOR SPECIFIC DOMAINS AND VERIFICATION OF DAPS

In order to test how much is easy to apply the developed approach in other contexts, we have considered two other scenarios (behinds the one described in the paper) in which we wish to develop DAPs that are controlled by means of a Domain Ontology. This requires from one side to extend the Domain Ontology with the concepts adopted in the considered scenario and to develop DAPs for the considered scenario guided by the corresponding Domain Ontology. In the reminder we report the lesson learned in these activities.

We have considered scenarios of water consumption and monitoring of the level of pollution in the zones of Milano.

1) SCENARIO 1: WATER CONSUMPTION

A clerk of the environment office of the Milano municipality wants to monitor the water consumption in Milano. If it is greater than 100 cubic metre, and the average value of the temperature is greater 25, then a warning is sent to the mayor. The operator selects the 20 sensors spread in Milano for monitoring the water consumption and 10 sensors for checking the temperature. Each sensor takes data every 3 hours. The operator creates a DAP in which he has to:

- convert each stream at a greater spatial granularity,

- unify the streams of each sensor that monitors the water consumption/temperature,
- join the obtained two streams in a single flow,
- trigger the notification to the Mayor only when the water consumption is greater than 100 cubic meter, and the temperature is greater 25 Celsius degree.

2) SCENARIO 2: MONITORING OF THE LEVEL OF POLLUTION

An operator of the municipality wishes to monitor the pollution level in a specific area (z2) of Milano in order to warn population of possible risks, for example related to the possibility to cause an asthma attack to children with respiratory problems.

The operator selects three sensors of a specific area z_2 to gather the PM10 value, the nitrogen dioxide (NO2) value, and the ozone (O3) value according to the following spatio-temporal granularities:

- the PM10 sensor retrieves data every hour,
- the NO2 sensor retrieves data every 20 minutes,
- the O3 sensor retrieves data every 30 minutes,
- the spatial dimension is missing in all three sensors.

The index of the air quality is calculated as the average of the two worst polluting substances. The following steps should be considered in the developed DAP:

- apply a convert operator on each sensor for calculating the average value for each hour;
- enrich the three stream in order to add the spatial dimension;
- join the three streams for generating a single flow and at the same time granularity, include a virtual property ''air quality'' that is calculated as the average of the two worst values that appear at the same time in the previous three streams;
- Trigger a notification only when the maximal the air quality index is greater than 150.

In these scenarios, the Domain Ontology (presented in the paper) needs to be modified for dealing with their peculiar thematics. For the application of the modifications we have asked to two Knowledge Engineers experts in the considered domains to modify the ontology in order to accommodate the new additional requirements. We have monitored the activities of the two engineers in order to assess the level of difficulty to create the required concepts (classes – if required, object property, data property, individuals and their relationship) in the Domain Ontology using Protégé and the clarity of each of the processes to create these use cases.

Both experts found quite easy the creation of individuals of the STT dimensions and inking with their respective concepts in the two scenario. However, linking each individual with its property, in case of thematic granularity, requires some time and a little bit of effort because different objects and data properties should be linked with the Unit and Meta-Data. Expert ''A'' mentioned that instance of ''amount'' is not intuitive and required elaboration. However, both experts agree on the intuitiveness of the overall organization of the Ontologies.

Finally, we have evaluated the content and structure of the obtained Domain Ontologies in terms of clarity and accomplishment to the intended goals. The results obtained by the two experts are quite equivalent and cover all the main requirements in the considered scenarios.

Once the developed Ontologies were ready, we asked three volunteers to develop the aforementioned DAPs with our interface. They were instructed on how to use our interface and on the specific domain. They were able to easily identify the required sensors that are disseminated in the different zones of Milano and correctly applying the services for changing the granularities of the produced streams and aggregating the data and applying the corresponding functions. Actually they found the information about the spatio-temporal granularities and the invalidity information (due to use of streams at different spatio-temporal granularities) provided by the interface particular useful for modifying their DAPs and make them workable in lesser time. Moreover, the presence of the constraints on the thematic of the final stream particularly useful for double checking the correctness of the entire DAP.

IX. RELATED WORK

Recent projects and tools in sensor network research area have focused their efforts on how to integrate events generated in raw and heterogeneous formats by means of a common semantics able to describe their meanings, the lack of which imposes barriers to interoperability among heterogeneous sensors. In the IoT domain, users are primarily interested in understanding the meaning of combined streams that can lead to the detection of significant events instead of raw data flows. Nevertheless, sensors provide raw data that do not contain any additional description or metadata and require specialized knowledge and manual effort for their meaningful combination.

To tackle this problem, several solutions [14], [30]–[32] using Linked Data Principles have been proposed [33]. They use strategies that aim at integrating sensor-data through Semantic Web technologies in order to publish data streams in an enriched and standardized way, so that they can be accessed and consumed by external applications. One of the key factors of these strategies is the possibility to describe the semantics of the sensor data according to three dimensions: Spatial, Temporal and Thematic (STT). Spatial metadata provide information about the sensor location, in terms of either a geographic reference system, local reference, or named location. Temporal metadata provide information about the time instant or interval when the sensor data are captured. Thematic metadata describe the meaning of the data-streams captured by the sensors (temperature, humidity, wind speed, and so on). All these metadata play an essential role in managing sensor data and provide more meaningful descriptions and enhanced access to sensor data.

In our work we do not require that sensors produce events that adhere to a pre-defined Ontology. By contrast, we allow sensors to produce events according to their own properties and schema. Then, when they are included in our system, individuals are included in the Domain Ontology for their semantic description.

Key point of our solution is that we do not force to have a complete description of the sensor schema w.r.t. to the Domain Ontology in order to be able to treat also events that are not totally described by the Domain Ontology. In these cases the user is informed by means of our graphical interfaces and can decide whether to maintain the selected sensors or choose other kinds of sensors.

In the remainder, we discuss other related work that are organized according to the following topics: IoT Ontologies and semantic interoperability, projects in the IoT Domain, and graphical interfaces for ETL operations.

A. IOT ONTOLOGIES AND SEMANTIC INTEROPERABILITY

With the development of semantic sensor networks, a number of ontologies describing the streams coming from the sensors have been brought forth in the past years. These ontologies rely on two different methodologies that it is possible to adopt for describing the semantics of the sensor data. The first one aims at describing sensor data through a mapping between the data-schema of the sensors with concepts and relations specified in the ontology. Instead, in the second methodology the ontology is used for annotating the data generated by sensors. An important example of the first type of semantic description regarding sensor networks ontologies, is provided by the Semantic Sensor Network (SSN) Incubator Group from W3C [17]. The SSN Incubator Group's purpose is to develop ontologies for sensor networks and search for appropriate methods for enhancing available standards with semantic technologies. As a result of the efforts of this group, the Semantic Sensor Network (SSN) ontology [19] was defined in order to describe sensor data according to their capabilities, the measurement processes and the resultant observations. The SSN ontology facilitates the enrichment and the semantic fusion of the sensor data, since it allows us to publish the streaming data and also integrate them with other related data sets. Sometimes, SSN ontology is not able to provide all the semantics needed by a specific domain context, and thus additional ontologies are often required such as IoT-Lite [18] and DOLCE-UltraLite [24] ontology. On the other side, the SSN Incubator Group also worked for developing a methodology to perform semantic annotations over the data generated by sensors following the standards defined by the Open Geospatial Consortium (OGC). These standards help describe observed phenomena such as space, time, and theme. In addition to the SSN Incubator Group' work, other works investigated methods and techniques for semantic annotating IoT devices and services, and their messages and data. An example of such approaches is described in [34] that proposes a data annotation architecture for semantic applications in virtualized Wireless Sensor Networks (WSN). The paper describes a WSN virtualization architecture based on data annotation and an ontology that extends the SSN ontology. In [34] the authors develop a Domain Ontology based on the SSN ontology for enabling a fire monitoring semantic application to receive annotated data and, along with a reasoner, to infer knowledge. Another framework, named Semantic Sensor Web [35], annotates sensor data and provides situational awareness. The annotation is done using spatial, temporal and thematic metadata. In [32], the authors use their own SenMESO ontology for annotation, which is a combination of various Domain Ontologies covering the sensor data and features of interest. Finally, in [14], the Annotation Ontology (SAO) has been specifically thought for handling real-time semantic annotations of data streams in dynamic environments. By using the SAO, it is possible to describe a data stream and a timeline instance to link the segment description with the time extent of a temporal entity representing the data stream. One flaw in adopting a semantic annotation for describing sensor data relies on its limited interoperability that aims at focusing on sensors one by one, rather than specifying a common semantics about their data. When a user annotates a data stream, he/she loses the focus on how to synchronize the schema of the flow under examination with the other data to integrate according a common knowledge base. Moreover, these solutions are too much domain-specific oriented, and their use of protocols such as Sensor Web Enablement (SWE) [36] that is difficult to setup. A better idea is to adopt a shared ontology to use for mapping each sensors schema according to their spatial, temporal and thematic dimensions. To do it, for ensuring interoperability, instead to create a Domain Ontology from scratch it is better to reuse efforts and background knowledge acquired during the design of existing ontologies already in IoT domain. This solution allows users to focus on semantic interoperability between the data flows coming from sensors by specifying the three dimensions according to which to discover significant events, instead of focusing to find the concepts that better describe the knowledge at the base of the ontology.

In our work we do not propose a new Ontology but we rely on the SSN and IoT-lite Ontologies. As discussed in [2] the current attempts to develop new Ontologies for the IoT use as backbone the SSN Ontology and provide specific extensions for their context of use. In our Ontology specific concepts have been introduced for the representation of the different spatio-temporal granularities and their relationships. Moreover, further concepts can be included in the Ontology depending on the context in which it is used. For addressing the requirements of our running scenario, we have included concepts for representing meteorological events (e.g. temperature, humidity, winds) and information extracted from Tweets. Then, in order to facilitate the interoperability among cross-domain IoT platform, sensors made available by the platforms go through a Semantic Virtualization process that allows to clarify the meaning of their properties. A ''soft bridge'' is thus created among the sensors offered by dif-

ferent platforms for the definition of Data Acquisition Plans required by the user. This means that if the user wishes to develop another analysis in the context of another Domain Ontology a new ''soft bridge'' can be easily specified.

B. PROJECTS IN THE IOT DOMAIN

Recent projects in smart city area (VITAL project, X-GSN, Xively, FIT [37], Hi Reply and OpenIoT [15], Spitfire [38] and iCore [39]) have focused their efforts on developing solutions for dealing with: (i) data acquisition (ii) semantic interoperability, and (iii) real-time data analysis and event detection. To mitigate the heterogeneity of data acquired by sensors, these projects use semantic technologies able to provide a uniform access to IoT data, through the use of semantic models such as the SSN Ontology. Also other recent projects such as CityPulse [40] and The Amsterdam Smart City (ASC) [41] or the prototype proposed in [42] aim at providing a resource data mapping via a linking semantic layer that use or extend the SSN Ontology. Specifically, they provide a set of real-time data analytics tools to provide strategies for data federation, data aggregation, event detection, quality analysis and decision support. These projects offer a set of services that facilitates easy access to data and operations that can be applied over the collected data. All cited projects aim at investigating how to use Semantic Web principles and available technologies for integrating sensor data in order to: *i*) extract domain knowledge (ontologies, data sets and rules) in specific contexts; *ii*) combine domain knowledge; *iii*) align and adapt IoT services; *iv*) make the domain knowledge interoperable by using semantic web methodologies and best practices; and, *v*) support IoT application development by means of semantic web technologies.

In the same direction are moving other projects specifically tailored for addressing the issue of IoT cross platform interoperability (e.g. the recently approved H2020 projects INTER-IoT [2], CREATE-IoT, and ACTIVAGE).

Overcoming such challenges will lead to solutions that provide interoperability among data coming from cross-platforms and a uniform guideline to the domain users who are responsible for monitoring significant events in specific contexts of use.

Despite the big challenges that these projects are addressing, many are the challenges that still need to be faced. First, no specific support is provided for dealing with the spatio-temporal granularities and the different unit of measures with which sensor data are produced. Taking into account these aspects are, from our point of view, really relevant for reasoning with streams of events produced by heterogeneous sensors. Moreover, the proposed environments can be exploited only by ICT-specialists because require the development of codes for the acquisition and manipulation of sensor data streams. By contrast, the GUI environment that we propose along with the services that are made available and the controls on the soundness and consistency of the developed Data Acquisition Plans allow also domain experts to develop plans and monitor their execution.

C. ETL OPERATIONS AND INTERACTIVE VISUAL INTERFACES

Dataflow design systems supporting a wide range of operations have been proposed in different contexts depending on the kinds of data to handle (structured and semi-structured). In [43]–[45] there is a good treatment of ETL (Extract, Transform, Load) operations at the conceptual level for feeding a Data warehouse. Moreover, approaches for the semi-automatic generation of ETL operations depending on the user needs and context of use are proposed in [46]. In data management, ETL operations are usually coupled with graphical visual dataflows for helping the user in the identification of the original data sources, the application of the operations for extracting, cleaning, transforming and combining their data. Once the ETL specification is completed, some strategies are proposed for the optimization of the dataflow and for the efficient execution of the loading schedule. These approaches have been mainly developed for producing relational data to feed conventional Data Warehouse System. In [47] an approach is presented for feeding arbitrary target sources (either relational or based on a NoSQL system). Commercial systems like for example Talend Studio, StreamBase Studio, Waylay.io offer graphical interfaces for designing workflows and dataflows as graphs of connected nodes representing tasks and data-sources. While Talend works on static data coming from fixed data-sources, StreamBase and WayLay can receive and analyse continuous data streams and are specifically designed for IoT. These environments provide rich user interface support for the full application life cycle, spanning feed integration, application modeling, development, data streams recording/playback, testing, and debugging. Anyway these are mainly desktop-based systems and in some cases conditions can be only created by adopting strategies based on programming languages paradigms (as for StreamSQL in StreamBase Studio) or by personalizing existing templates with well-defined trigger policy (as in Waylay.io). Visual queries are other examples of solutions for assisting users to extract information from databases in an intuitive, visual and natural approach, making information systems comprehensive and efficient for a wide range of applications. SmartVortex Visual Query System [48] is a typical application where the visual strategy of querying takes the form of drawings or graph. Also in this case the expected users have limited competence for managing data streams but they wish to express basic queries with little efforts and no competence in coding.

What is missing in these visual approaches is a strategy for specifying the semantic of the gathered data. In order to combine and handle heterogeneous data we need to understand the data type, when they are acquired and where. In order words, we need to know the data before handling them. In our idea, the final environment has to allow users to acquire data according to their STT dimensions by exploiting a Domain Ontology able to check the correspondence between the dimensions adopted in the streams with those specified by the domain experts. The system has to be designed for

simplifying the developing of data extraction and integration facilities. For this reason, the environment has to be simply used for the specification of the dataflow but also for monitoring its execution and for achieving or maintaining the semantic consistency of the dataflow in presence of on-line modifications. In this way, semantic poor data can be extended and integrated during the design of the dataflow to monitor, in order to adhere to a semantic model able to effectively describe sensor data according to their spatial, temporal and thematic dimensions.

X. CONCLUSION AND FUTURE WORK

In this paper we have addressed the issue of the Semantic Virtualization of sensors belonging to cross-domain IoT platforms and to generate Data Acquisition Plans that can be soundly executed and that adhere to the constraints specified within a Domain Ontology. For this purpose, we have identified a flexible multi-granular STT data model according to which the schema of heterogeneous sensors belonging to cross-domain IoT platforms can be easily transformed by using standard wrapping tools.

Then, the user can apply a Semantic Virtualization process according to which the sensors and the schema of the events they generate are semantically described at the ontological level by means of instances of the Domain Ontology the user wishes to use for conducting a given kind of analysis. This process is carried out by taking into account the STT dimensions according to which the events generated by the sensors are produced and has the purpose to move towards the adoption of a common semantics of the sensor event streams. However, the description can be partial to deal with situations in which sensors are not equipped with the facilities for associating the STT dimensions specified in the model. Conditions are specified for guaranteeing the consistency of a sensor relying on the consistency of the STT dimensions of the produced events. At this point the user can generate Data Acquisition Plans in which new information is associated with the events produced by the sensors and filter, join aggregation and transformation services can be specified. The result of the application of each service produces a new stream whose consistency can be evaluated w.r.t. the Domain Ontology for its semantic characterization.

The proposed solution allows the creation of ''soft bridges'' among the sensors belonging to cross-domain IoT platforms for the definition of Data Acquisition Plans required by the user. This means that if the user wishes to develop another analysis in the context of another Domain Ontology a new ''soft bridge'' can be specified. The main characteristics of the StreamLoader prototype system are finally discussed. To be best of our knowledge this is the first system for the specification of sound and consistent Data Acquisition Plans w.r.t. a given Domain Ontology specifically tailored for the IoT context. We are currently working on the evaluation of the usability of the proposed interfaces.

Another research direction concerns the analysis of non-consistent Data Acquisition Plans in order to identify similarities and manually or automatically propose Ontology evolution in order to adhere to new requirements emerged in the domain. Finally, we wish to combine approaches for the semi-automatic Semantic Virtualization of sensors with our GUI environment.

REFERENCES

[1] (2016). *Gartner*. [Online]. Available: http://www.gartner.com/newsroom/id/3165317
[2] M. Ganzhaa, M. Paprzyckia, W. Pawlowskib, P. Szmejaa, and K. Wasielewska, ''Semantic interoperability in the Internet of Things: An overview from the inter-IoT perspective,'' *J. Netw. Comput. Appl.*, vol. 81, pp. 111–124, Mar. 2017, doi: 10.1016/j.jnca.2016.08.007.
[3] (2018). *List of 640+ Enterprise IoT Projects*. [Online]. Available: https://iot-analytics.com/product/list-of-640-iot-projects/
[4] Open Mobile Alliance (OMA) Specification. (2010). *Ngsi Context Management Specifications Defined by OMA*. [Online]. Available: http://technical.openmobilealliance.org/Technical/release_program/docs/NGSI/V1_0-20120529-A/OMA-TS-NGSI_Context_Management-V1_0-20120529-A.pdf
[5] ETSI. (2013). *Machine-to-Machine Communications (m2m); Mia, DIA and Midinterfaces*. [Online]. Available: http://www.etsi.org/deliver/etsi_ts/102900_102999/102921/02.01.01_60/ts_102921v020101p.pdf
[6] D. Pavithra and R. Balakrishnan, ''IoT based monitoring and control system for home automation,'' in *Proc. Global Conf. Commun. Technol. (GCCT)*, 2015, p. 169–173, doi: 10.1109/GCCT.2015.7342646.
[7] O. Bibani, C. Mouradian, S. Yangui, R. H. Glitho, W. Gaaloul, N. Ben Hadj-Alouane, M. Morrow, and P. Polakos, ''A demo of IoT healthcare application provisioning in hybrid cloud/fog environment,'' in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2016, pp. 472–475, doi: 10.1109/CloudCom.2016.0081.
[8] S. Jabbar, M. Khan, B. N. Silva, and K. Han, ''A REST-based industrial Web of things' framework for smart warehousing,'' *J. Supercomput.*, vol. 74, pp. 1–15, Dec. 2016, doi: 10.1007/s11227-016-1937-y.
[9] V. Rajaraman, P. Misra, K. Dhotrad, and J. Warrior, ''Enabling plug-n-play for the Internet of Things with self describing devices,'' in *Proc. Int. Conf. Inf. Process. Sensor Netw.*, 2015, pp. 374–375, doi: 10.1145/2737095.2742927.
[10] M. Khan, S. Din, S. Jabbar, M. Gohar, H. Ghayvat, and S. C. Mukhopadhyay, ''Context-aware low power intelligent smarthome based on the Internet of Things'' *Comput. Electr. Eng.*, vol. 52, pp. 208–222, May 2016, doi: 10.1016/j.compeleceng.2016.04.014.
[11] A. Bröring, S. Schmid, C.-K. Schindhelm, A. Khelil, S. Käbisch, D. Kramer, D. Le Phuoc, J. Mitic, D. Anicic, and E. Teniente, ''Enabling IoT ecosystems through platform interoperability,'' *IEEE Softw.*, vol. 34, no. 1, pp. 54–61, Jan./Feb. 2017, doi: 10.1109/MS.2017.2.
[12] J. Manyika Ed., *The Internet of Things: Mapping the Value Beyond the Hype*. New York, NY, USA: McKinsey Global Institute, 2015.
[13] E. Bertino, S. Nepal, and R. Ranjan, ''Building sensor-based big data cyberinfrastructures,'' *IEEE Cloud Comput.*, vol. 2, no. 5, pp. 64–69, Sep./Oct. 2015, doi: 10.1109/MCC.2015.106.
[14] S. Kolozali, M. Bermudez-Edo, D. Puschmann, F. Ganz, and P. Barnaghi, ''A knowledge-based approach for real-time IoT data stream annotation and processing,'' in *Proc. IEEE Int. Conf. Internet Things*, Sep. 2014, pp. 215–222.
[15] J. Soldatos, N. Kefalakis, M. Hauswirth, M. Serrano, J.-P. Calbimonte, M. Riahi, K. Aberer, P. P. Jayaraman, A. Zaslavsky, I. P. Žarko, L. Skorin-Kapov, and R. Herzog, ''OpenIoT: Open source Internet-of-Things in the cloud,'' in *Proc. Int. Workshop, FP7 OpenIoT Project, Held Conjunct. SoftCOM*, vol. 9001. Split, Croatia: Springer, 2015, ch. 1, pp. 13–25, doi: 10.1007/978-3-319-16546-2_3.
[16] T. Zahariadis, A. Papadakis, F. Alvarez, J. Gonzalez, F. Lopez, F. Facca, and Y. Al-Hazmi, ''FIWARE lab: Managing resources and services in a cloud federation supporting future Internet applications,'' in *Proc. IEEE/ACM 7th Int. Conf. Utility Cloud Comput. (UCC)*, Dec. 2014, pp. 792–799.
[17] W3C Semantic Sensor Network Incubator Group. (2005). *Semantic Sensor Network Ontology*. [Online]. Available: https://www.w3.org/2005/Incubator/ssn/ssnx/ssn
[18] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi, and K. Taylor. (2015). *IoT-Lite Ontology*. [Online]. Available: http://www.w3.org/Submission/iot-lite/
[19] M. Compton, P. Barnaghi, L. Bermudez, R. García-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, and A. Herzog, ''The SSN ontology of the W3C semantic sensor network incubator group'' *Web Semantics, Sci., Services Agents World Wide Web*, vol. 17, pp. 25–32, Dec. 2012.

[20] J. Masterton and F. Richardson, "Humidex: A method of quantifying human discomfort due to excessive heat and humidity," in *Environment Canada, Atmospheric Environment* (Atmospheric Environment Service). Canada: CLI, 1979, p. 45
[21] E. Camossi, M. Bertolotto, and E. Bertino, "A multigranular object-oriented framework supporting spatio-temporal granularity conversions," *Int. J. Geograph. Inf. Sci.*, vol. 20, no. 5, pp. 511–534, 2006.
[22] E. Camossi, E. Bertino, M. Mesiti, and G. Guerrini, "Handling expiration of multigranular temporal objects," *J. Log. Comput.*, vol. 14, no. 1, pp. 23–50, 2004.
[23] C. Bettini, S. Jajodia, and X. Wang, *Time Granularities in Databases, Data Mining, and Temporal Reasoning*. Berlin, Germany: Springer-Verlag, 2000.
[24] S. Borgo and C. Masolo, "Ontological foundations of dolce," in *Theory and Applications of Ontology: Computer Applications*. New York, NY, USA: Springer, 2010, pp. 279–295.
[25] Y. Raimond and S. Abdallah. (2007). *The Timeline Ontology*. [Online]. Available: http://motools.sourceforge.net/timeline/timeline.html
[26] L. Lefort. (2011). *Ontology for Quantity Kinds and Units: Units and Quantities Definitions*. [Online]. Available: http://purl.oclc.org/NET/ssnx/qu/qu-rec20
[27] L. Ferrari, S. Valtolina, and M. Mesiti, "Developing IoT spark-streaming applications bymeans of streamloader" in Proc. *4th Int. Symp. Ubiquitous Netw.* Cham, Switzerland: Springer, May 2018, pp. 116–127.
[28] D. Schuler and A. Namioka, Eds., *Participatory Design: Principles and Practices*, Hillsdale, NJ, USA: L. Erlbaum Associates, 1993.
[29] S. Valtolina, B. R. Barricelli, and Y. Dittrich, "Participatory knowledge-management design: A semiotic approach," *J. Vis. Lang. Comput.*, vol. 23, no. 2, pp. 103–115, 2012.
[30] M.-S. Dao, S. Pongpaichet, L. Jalali, K. Kim, R. Jain, and K. Zettsu, "A real-time complex event discovery platform for cyber-physical-social systems," in *Proc. Int. Conf. Multimedia Retr. (ICMR)*, New York, NY, USA: ACM, 2014, pp. 201–208.
[31] M.-S. Dao, K. Zettsu, S. Pongpaichet, L. Jalali, and R. Jain, "Exploring spatio-temporal-theme correlation between physical and social streaming data for event detection and pattern interpretation from heterogeneous sensors," in *Proc. IEEE Int. Conf. Big Data*, Oct./Nov. 2015, pp. 2690–2699.
[32] A. Sheth, C. Henson, and S. S. Sahoo, "Semantic sensor Web," *IEEE Internet Comput.*, vol. 12, no. 4, pp. 78–83, Jul./Aug. 2008.
[33] T. Berners-Lee, (2016). *Linked Data—Design Issues*. [Online]. Available: http://www.w3.org/DesignIssues/LinkedData.html
[34] I. Khan, R. Jafrin, F. Z. Errounda, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, "A data annotation architecture for semantic applications in virtualized wireless sensor networks," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 27–35, doi: 10.1109/INM.2015.7140273.
[35] A. Gyrard, "A machine-to-machine architecture to merge semantic sensor measurements," in *Proc. Int. Conf. World Wide Web*, 2013, pp. 371–376, doi: 10.1145/2487788.2487945.
[36] M. Botts, G. Percivall, C. Reed, and J. Davidson, *OGC Sensor Web Enablement: Overview and High Level Architecture*. Berlin, Germany: Springer, 2006, ch. 4, pp. 175–190, doi: 10.1007/978-3-540-79996-2_10.
[37] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, and T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne, "FIT IoT-LAB: A large scale open experimental IoT testbed," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 459–464, doi: 10.1109/WF-IoT.2015.7389098.
[38] D. Pfisterer, K. Romer, D. Bimschas, O. Kleine, R. Mietz, C. Truong, H. Hasemann, A. Kröller, M. Pagel, M. Hauswirth, M. Karnstedt, M. Leggieri, A. Passant, and R. Richardson, "Spitfire: Toward a semantic Web of things," *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 40–48, Nov. 2011, doi: 10.1109/MCOM.2011.6069708.
[39] R. Giaffreda, *iCore: A Cognitive Management Framework for the Internet of Things*. Berlin, Germany: Springer, 2013, ch. 5, pp. 350–352, doi: 10.1007/978-3-642.
[40] D. Puiu, P. Barnaghi, R. Tönjes, D. Kümper, M. I. Ali, A. Mileo, J. X. Parreira, M. Fischer, S. Kolozali, N. Farajidavar, F. Gao, T. Iggena, T.-L. Pham, C.-S. Nechifor, D. Puschmann, and J. Fernandes, "Citypulse: Large scale data analytics framework for smart cities," *IEEE Access*, vol. 4, pp. 1086–1108, 2016, doi: 10.1109/ACCESS.2016.2541999.
[41] (2015). *Amsterdam Smart City*. [Online]. Available: http://amsterdamsmartcity.com/#/nl/home
[42] Z. Khan, A. Anjum, K. Soomro, and M. A. Tahir, "Towards cloud based big data analytics for smart future cities," *J. Cloud Comput.*, vol. 4, no. 1, p. 2, 2015, doi: 10.1186/s13677-015-0026-8.
[43] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos, "Conceptual modeling for ETL processes," in *Proc. 5th ACM Int. Workshop Data Warehousing OLAP (DOLAP)*, New York, NY, USA: ACM, 2002, pp. 14–21.
[44] M. Gorawski and A. Gorawska, "Research on the stream ETL process," in *Proc. Int. Conf. Beyond Databases, Archit., Struct.*, 2014, pp. 61–71.
[45] H. Zhou, D. Yang, and Y. Xu, "An ETL strategy for real-time data warehouse," in *Practical Applications of Intelligent Systems*, (Advances in Intelligent and Soft Computing), vol. 124. Berlin, Germany: Springer, 2012, pp. 329–336.
[46] V. Theodorou, A. Abelló, M. Thiele, and W. Lehner, "A framework for user-centered declarative ETL," in *Proc. 17th Int. Workshop Data Warehousing OLAP (DOLAP)*. New York, NY, USA: ACM, 2014, pp. 67–70.
[47] M. Mesiti and S. Valtolina, "Towards a user-friendly loading system for the analysis of big data in the Internet of Things," in *Proc. IEEE 38th Annu. Comput. Softw. Appl. Conf. (COMPSAC) Workshops*, Vasteras, Sweden, Jul. 2014, pp. 312–317.
[48] E. Bauleo, S. Carnevale, T. Catarci, S. Kimani, M. Leva, and M. Mecella, "Design, realization and user evaluation of the smartvortex visual query system for accessing data streams in industrial engineering applications," *J. Vis. Lang. Comput.*, vol. 25, no. 5, pp. 577–601, 2014, doi: 10.1016/j.jvlc.2014.08.002.

STEFANO VALTOLINA received the M.Sc. degree in computer science from the Università degli Studi di Milano, and the Ph.D. degree in informatics. He is currently an Assistant Professor with the Department of Computer Science, Università degli Studi di Milano, Italy. His research interests include the intersection between end-user development, human–computer interaction, interaction and participatory design, human-work interaction design, and computer semiotics. In this context, he has taken part in several national and European Projects in different applicative domains. Specifically, he deals with the study and development of methods for the design and evaluation of interactive systems with the involvement of domain experts in specific application domains, such as the e-government, cultural heritage, the IoT, and SmartCity.

LUCA FERRARI received the Ph.D. degree in computer science from the University of Milan. The research activities of his PhD work have been mainly focused on the development of a system for the specification of data acquisition plans for streams of real-time heterogeneous data and for their semantic virtualization. He is currently a Computer Science Technician with the Information System Division, University of Milan, Italy. He was involved, during his master's degree thesis in computer science for communication, in a European Project named "SandS." The main topic of this work has been the development of an advanced domotic system for household appliances. The main results of his work have been reported in this paper.

MARCO MESITI received the Laurea degree *(cum laude)* and the Ph.D. degree in computer science from the University of Genova, Italy. He has been a Visiting Researcher at the Applied Research Center, Telcordia Technologies, Morristown, NJ, USA. He is currently an Associate Professor with the Department of Computer Science, University of Milan, Italy. His research activity has been carried out in the database area, and has focused on semistructured and XML data handling, with specific interests in access control models, data integration, approximate retrieval, and schema evolution. He is currently working on approaches for facing the heterogeneity of the IoT data streams and thus making easy the development of cross-platform IoT applications. He has been recently involved in a precommercial-procurement for the development of the "Snap4city" IoT Platform in the context of the SELECT4CITIES European Project.