Linked open data in sensor data mashups

Danh Le-Phuoc and Manfred Hauswirth

Digital Enterprise Research Institute,
National University of Ireland, Galway, Ireland

Abstract. Sensors and the real-time data they produce are novel sources of information which need to be integrated into the Semantic Web at very large scale. Most of the time such data is locked inside specific applications and only accessible within organizational boundaries. Publishing and integrating sensor data across these islands is difficult and labor-intensive. In this paper we present an approach and an infrastructure which makes sensor data available following the linked open data principle and enables the seamless integration of such data into mashups. SensorMasher publishes sensor data as Web data sources which can then easily be integrated with other (linked) data sources and sensor data. Raw sensor readings and sensors can be semantically described and annotated by the user. These descriptions can then be exploited in mashups and in linked open data scenarios and enable the discovery and integration of sensors and sensor data at large scale. The user-generated mashups of sensor data and linked open data can in turn be published as linked open data sources and be used by others.

Key words: Mashup, sensor web, semantic sensor data

1 Introduction

If we define a sensor as a data source which produces a sequence of data items over time – a data stream – then we already have a very large number of sensors deployed all over the planet. Following this unifying definition beyond the "classical" definition of sensors, a very large number of sensors already exist and have been deployed: Each mobile phone already hosts a number of sensors, such as GPS, camera, Bluetooth, WLAN, accelerometer, etc., standard computers and CPUs carry a multitude of sensing devices which are already used for remote administration at large scales, large deployments of sensors in the environmental domain, for monitoring traffic, for logistics, supply chain management, etc. exist already and this is not the end of the development.

Gartner predicts that *"By 2015, wirelessly networked sensors in everything we own will form a new Web. But it will only be of value if the 'terabyte torrent' of data it generates can be collected, analyzed and interpreted."*[1] Making sensor-generated information usable as a new and key source of knowledge will require its integration into the existing information space of the Web.

[1] Mark Raskino, Jackie Fenn, and Alexander Linden. Extracting Value From the Massively Connected World of 2015. Gartner Research, 1 April 2005. http://www.gartner.com/resources/125900/125949/extracting_valu.pdf

Besides the problem of coping with the vast amounts of data which will be produced, most of these data sources are not easily accessible to enable integration with other data. One of main reasons for this may be the fact that research in the sensor network area has focused on routing, data aggregation and energy consumption inside a single sensor network while the integration of multiple sensor networks has only been studied to a limited extent. On top of that, as the price of commodity sensors is rapidly decreasing we can soon expect large numbers of autonomous sensor networks being deployed. Hence, there is the need for platforms to publish and share sensor data in an easy way and to reduce the cost and complexity of sensor data access and integration.

Several attempts has been made to remedy the problems of collecting and publishing sensor data online, for example, SensorMap[2], EarthScope[3], and Earthcam[4], etc. However, these data sources are difficult to integrate and to make them accessible to other applications. Moreover, sensor discovery, which is vital in the large scales, is impossible with most providers or rarely supported. Additionally, to the best of our knowledge, no research is trying to target support for the casual user who does not have full programming skills while still wants to have access to sensor information and integrate it into Web applications. This gap may prohibit very creative and useful applications of sensor data by non-technical users.

Sensed data is often archived or streamed as raw data, but rarely associated with enough metadata describing its meaning. Meaning of sensor data includes the feature of interest, the specification of measuring devices, accuracy, measuring condition, scenario of measurements, location, etc. Such metadata is essential when the user is confronted with large numbers of sensors and gigabytes of sensor data. Especially, when the user does not have clear ideas about what he/she is looking for, he/she can start a quite general search of relevant concepts and narrow it down based on semantic descriptions and their relations. For example, a city planner may want to assess and monitor quality of life in certain area. To do so, he/she can start to navigate from his/her own domain of knowledge,then finds out that quality of life depends on noise, sunlight, humidity, air pollution, traffic condition, etc. Then he can filter out which sensor sources can provide such data in his area of interest.

To enable easy access to sensor data to non-technical users, we propose our SensorMashup platform as a step towards the vision of "The Web of Thing". Our platform will enable the publishing of sensors and the associated data sources as Web citizens under URIs. These published entities, called Sensor Mashups, will be created and then linked to ontological concepts, other (virtual) sensors and other Web citizens under URIs through a visual composer. This phase will create useful linked data for sensor discovery. Hence, the visual composer is able to provide an intuitive GUI which allows users to navigate and explore sensor data sources by following semantic links and using faceted-browsing techniques. After having found the relevant sensor data sources, the user is able to combine

[2] SensorMap : http://atom.research.microsoft.com/sensewebv3/sensormap/
[3] EarthScope - An Earth Science Program : http://www.earthscope.org/
[4] EarthCam - Webcam Network : http://www.earthcam.com/

them visually in a workflow editor and then connect them to data processing operators to create a new Sensor Mashup. Sensor data published from our platform can be accessed through SPARQL endpoints and RESTful web services under JSON, XML, RDF formats which can be easily used by various applications. Triple-based query support enables semantic-based traversal over sensor data, thus makes the data filtering and correlating more sophisticated than those of previous approaches. Triple-based query support is only used to enable access via standard mechanisms while internally we use compressed data representations for efficiency reasons.

2 Motivating scenarios

A large number of use case scenarios have been proposed already but to the best of our knowledge none takes into account linked open data as a principle for the seamless integration of physical and virtual data. For our motivating scenario we use the notion of "integrated presence management". The goal of integrated presence management is to combine sources of virtual presence, for example, calendar information, online status in Skype and chats, in IP telephone systems, collaborative environments, etc., with information about physical presence, for instance, location determined via GPS, WLAN, Bluetooth, motion detection sensors, RFID, noise sensors, etc., to build up an integrated view of an entity's presence and availability. This information can then be used in simple yet technically quite sophisticated scenarios. For example, if information from the user' calendar is combined with physical location, then appointments can be automatically changed if the system somehow can use users' current physical locations to infer that all participants are not able to get to meeting location in time, or, despite being seen available on Skype, on the phone, and in a chat client, a user may have an ongoing meeting which can easily be determined via audio sensors and/or the physical proximity of RFID tags in the user's office. In more sophisticated scenarios, someone may leave a location-based notification for a user who is not in his office and which will be delivered as soon as the addressee gets close to this location again, e.g., his office. Besides these user-centric scenarios, a plethora of scenarios also exists for the management of things, e.g., for a meeting not only the participants need to be there but also resources like projectors, etc. which are movable and have a usage context.

It is quite simple and straight-forward to come up with further more complex scenarios based on these basic scenarios, for example, if also profiles, policies, and privacy are taken into account. However, the underlying requirement in all these scenarios is the need for flexible, transparent integration of data from diverse sources and sensors. It can be accomplished very efficiently with a linked data approach for sensor data sources.

3 Linking sensor data to the Web

In order to link real world data captured by sensors to the Web, it has to become a form of Web resource. Sensors and their data elements can become Web re-

sources by publishing their data on the Web using linked data techniques[5]. When adopting such an approach, sensors and their data are assigned a URI which enables the creation of links to other Web entities. Using a similar method to *Stream Feeds* [10], each sensor has a URL pattern for streaming real-time data as well as historical data. This pattern also includes semantic descriptions of the sensor and sensor data model as described in the following.

As sensors and their data elements have URI identifiers, we are able to link them into a virtual RDF graph as shown in Figure 1. The graph is called a *virtual* graph, because the whole graph is not materialized and stored in a triple storage. It is constituted by interlinked subgraphs controlled by sensor mashups. A mashup's metadata stored in the metadata repository defines how the raw sensor data in a data stream can be linked to domain knowledge and external linked data. This is driven by a set of ontologies which are used to capture facts and the data model of the sensor information system. This linked metadata will guide the finding of triple patterns needed from the virtual RDF graph of sensor data.

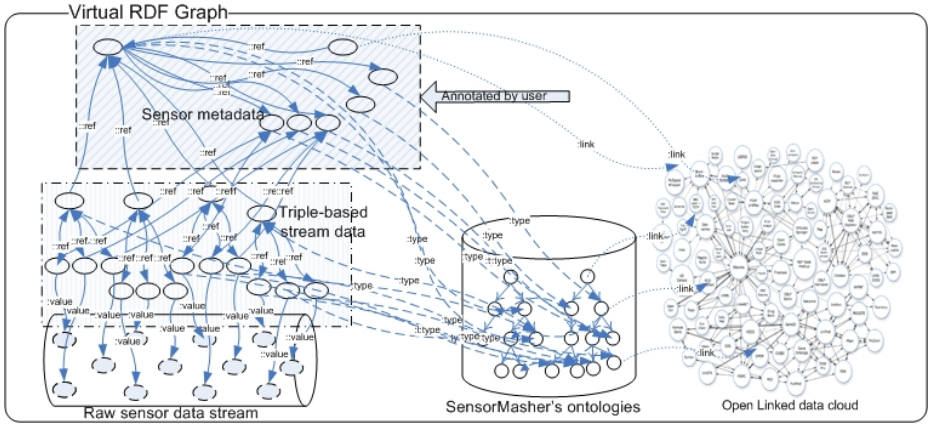

Fig. 1. Linking raw data stream to virtual RDF graph

Sensor data from multiple sources can be fused to create derived sensor data sources. We are later able to use these fused sensor data sources in the same way as the original sources. The process of composing a new sensor data source from existing ones is facilitated by the sensor mashup composer of SensorMasher which enables the user to rapidly create new sensor data sources in a visual way. The composing process generates the mashup's configuration used to controls the data flows from sensors to data streams and from data streams to data processing operators. A mashup's configuration is also stored in the metadata repository. With the annotated sensor data streams, the composing operation can exploit existing approaches proposed, such as [7] and [21]. These approaches employ reasoning to enable the provisioning of interfaces for automatically composing a sensor data source meeting a user's goals.

[5] http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/

The most important operation when integrating data is exploring existing data sources to find out which ones are of interest. The triple-based query processor will facilitate the semantic-based sensor data exploration. A faceted browsing functionality helps the user to filter sensor data based on relevant facts. For example, a user can filter by sensor type as classified in the sensor taxonomy, or by sensor location by choosing an area in a map, or by sensor specification (sensor type, physical context, calibration parameters, precision, accuracy), etc. The user can navigate from a data set to other data sets by following semantic links. Under the hood, the exploration and navigation actions are translated into complex queries (spatial, temporal and thematic queries) to the query processor as described in [17].

Along with a URL pattern for direct access to each sensor's data elements, SensorMasher also provides a SPARQL endpoint to query all types of sensor data. To identify a sensor, a query over the semantic descriptions of sensors can be posed to this SPARQL endpoint, for instance, "find the nearest webcam from a tourist venue." Queries over sensor data are also supported, for example, "find the latest air temperature reading if the average humidity in last 3 hours was higher than 80%."

Sensor and sensor data are also accessible via Web services which conform to Sensor Web Enablement (SWE) standards [29]. However, to support interoperability among semantic information system, we use RDFa to embed semantic descriptions in WSDL as well as the output data. The descriptions of these web services are dynamically generated from semantic descriptions of the sensors and mashups.

The SPARQL endpoint, web services and streaming protocol allow a SensorMasher node to integrate sensor data from a remote one transparently. Hence, a set of autonomous SensorMasher nodes will create a federated sensor information system from which sensor data can travel and be queried across the borders of single systems.

4 System design

4.1 Architectural view

Figure 2 shows the architecture of the SensorMasher platform. The bottom of the diagram shows the wrappers for collecting sensor data. The sensor data is streamed to the Data Stream Management System (DSMS) and fusion operators in the next layer. The data flows of the streams are coordinated and managed by the Query Processor, the Sensor&Mashup Manager and the User Manager in the Mediator layer. The Frontend components such as Explorer, Composer and Web interfaces on the top use interfaces provided by the Mediator layer to access the sensor data streams. The detailed descriptions of these components are given below.

Wrappers provide interfaces for receiving sensor stream data from physical sensors via interfaces such as USB, Bluetooth, and serial ports. The wrappers are responsible for interfacing with these physical sensors whereby the SensorMasher node plays the role as a gateway between a sensor network and the Web

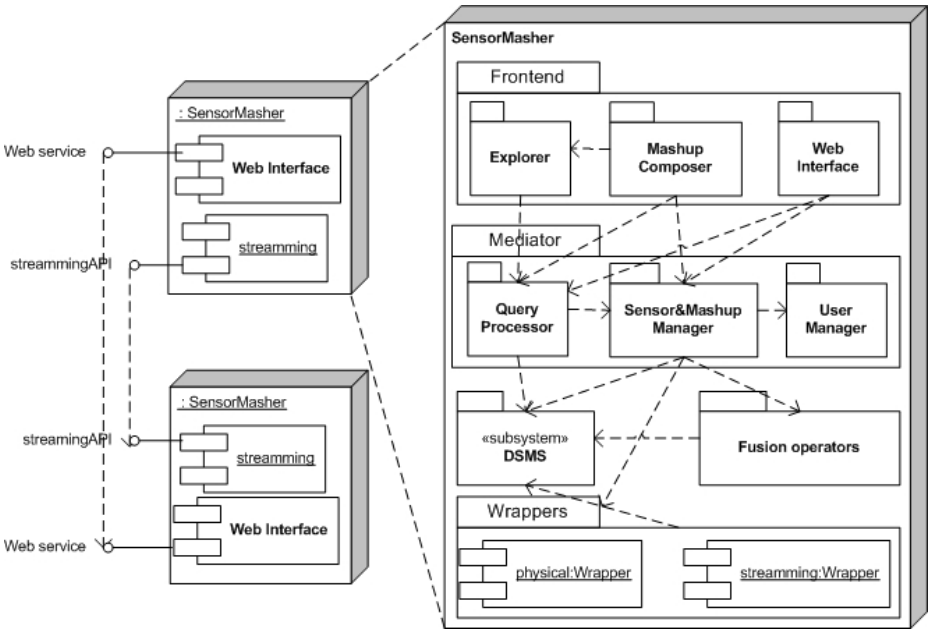

Fig. 2. Architectural view

and vice versa. Sensor data can be streamed from IP-based connections in both push-based and pull-based fashions. Using the push-based paradigm, the wrappers listen and wait for the sensor data being actively streamed from a peer. Alternatively, in the pull-based paradigm, the wrappers periodically pull the data from a remote peer, for instance, by making an HTTP request. As soon as a new reading arrives, the wrapper pushes the data into the appropriate stream in the DSMS component.

Sensor data can arrive in the ways which cannot be directly handled in persistent storage systems such as relational databases. Data streams differ from conventional stored relational models in several aspects. The data streams not only have data elements arriving continuously but also potentially have unbounded size. Additionally, the order of data elements arriving within a data stream or across multiple data streams is unpredictable and not necessarily arranged. Furthermore, as soon as an element in a data stream has been processed it can be discarded or archived, thus it cannot be no longer easily retrieved, unless it is explicitly stored in memory, which typically is small relative to the size of the data streams. To deal with these characteristics, data stream management systems (DSMS) are designed for monitoring, combining and analyzing and correlating streams of data [4] rather than following the design of traditional data management systems. Designing and implementing such a data stream management system, see [2, 8, 16] for examples, is beyond the scope of this paper. In the SensorMasher platform, we use the DSMS as a blackbox in a subsystem which provides interfaces to create sensor data streams as well as provides APIs supporting continuous queries over data streams. As soon as a stream is created, it

can listen to incoming data from the underlying wrappers and/or other streams. When new data arrives to the stream, the DSMS routes the data to further listeners such as fusion operators or listening data streams.

The fusion operators implement typical sensor data processing operations including data filtering, data alignment and association, correlation and classification. They employ algorithms and techniques used in multisensor data fusion process models [12] to allow users to incrementally build data processing workflows that may generate real-world observations. A special type of fusion operator provides the extraction of semantics from sensor readings produced by wrappers. For raw sensor data fed from physical wrappers, a template for annotating semantics to sensor readings will be generated using the semantic description of the physical sensor. For example, MICAz is a well-known wireless measurement system which provides connectors to plug in sensors measuring light, temperature, barometric pressure, etc. With foreseen meanings from output data structure, a template to enrich semantics of the bit stream sending from this system can be generated. For sensor data provided in XML format, the XSLT transformation operators similar to the ones introduced in the Semantic Web Pipes [13] architecture can be used. Semantic descriptions embedded in XML-based sensor data using RDFa or microformats can be easily extracted in the same manner as done in Semantic Web Pipes.

The Sensor & Mashup manager component controls the data flow from sensors to the DSMS and from the DSMS to the fusion operators and vice versa. The Sensor & Mashup manager also provides interfaces to the metadata repository such as editing and querying metadata under a triple-based model. It also enables the deployment of sensor mashups by initializing associated wrappers to feed data from sensors. It is also responsible for providing access control via the User Manager component. The User Manager manages profiles of the mashups' authors as well as the mashups' sharing policy.

The Query Processor component supports triple-based queries over sensor data from both the DSMS component and the metadata repository. To process such queries, it has to follow the mapping rules that are generated from the metadata repository. These rules guide it in delegating continuous queries to the DSMS component as well as triple-based queries to the metadata repository.

The Explorer component provides a GUI enabling the user to explore sensor data on a map or along semantic links. This Explorer also includes a facet-based filter over triple-based sensor data. This semantic exploration is empowered by APIs, the Query Processor and Sensor & Mashup Manager components and Web services provided from remote SensorMasher nodes. The Web interfaces provided include a Web services interface, a SPARQL endpoint and HTTP requests from URL-based Sensor Web sources. The Mashup composer is used for visually composing sensor mashups by creating a workflow connecting sensor data sources with fusion operators. In composing processes, the user has to employ the Explorer component to search for the input data sources required.

The SensorMasher platform can be deployed in a federated fashion whereby autonomous nodes are able to query each other via Web interfaces. The stream-

ing APIs provided by wrappers allow a local node to stream data to a remote one and vice versa.

4.2 Ontology-driven component design

Using the Sensor model of the SensorML[6], we abstract the sensor model, sensor data model and data processing components in Figure 3, and represent this model in OWL-DL. In this data model, the sensor mashup is the key abstraction of the SensorMasher. We define a sensor mashup as an observation (Observation class). An observation is conducted to observe an feature of interest of a real world object. For example, "finding vacant parking spots in an underground parking lot" is an observation which targets the real world object "underground parking lot" and is interested in the feature "vacant parking spots". A feature of interest may have several properties which can be measured by sensing devices, called ObservableProperty. In the previous example, the observable property is "car in parking lot or not". The ObservableProperty class is the basic class that describes the meaning of sensor outputs. For example, Temperature and Humidity are sub-concepts of ObservableProperty, that represent the meanings of temperature and humidity readings.

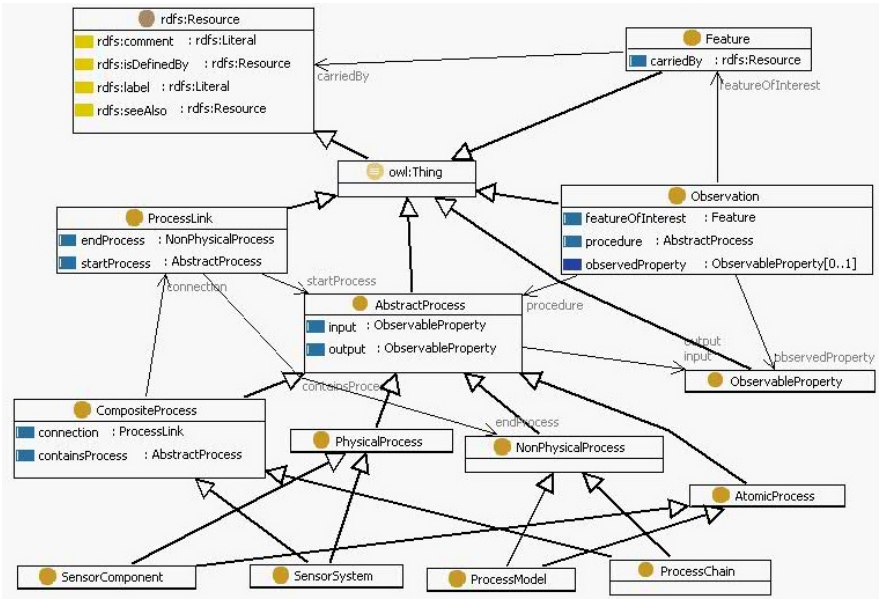

Fig. 3. Ontology-driven system design diagram

The observation is facilitated by a sensing process represented as AbstractProcess. The subclasses of AbstractProcess describe 4 types of sensing processes. The first one, SensorComponent, is for representing an atomic sensor to detect a single sensing signal. The second one is for presenting a sensing system (SensorSystem) containing a set of atomic sensors. For instance, a weather station[7]

[6] Sensor Model Language :http://www.opengeospatial.org/standards/sensorml
[7] http://en.wikipedia.org/wiki/Weather_station

is a sensing system which has autonomic sensors such as thermometer, barometer, anemometer, etc. The third one (ProcessModel) is for representing fusion operators as software processes that are used to process sensor data. The last one is the composition of sensing processes, called ProcessChain. This process also contains a list of links to describe the flow of sensor data streams from constituted processes. The sensor mashup often uses instances of ProcessChain to define how the data from sensors is to be connected to the fusion operators along the predefined workflow to generate a new output data stream. This instance plays the role of the mashup's configuration.

As mentioned earlier, we use the DSMS for managing sensor data streams and do not expose all sensor readings as materialized triples. The triple-based query over sensor data linked into the virtual RDF graph mentioned in Section 3 is parsed to create a query plan consisting of a set of subqueries. There exist two types of subqueries: continuous queries over stream data and triple-based queries over sensor metadata. The query planner will delegate the corresponding subqueries to the DSMS and the triple-based query processors of the metadata repository. The query aggregator later composes the individual partial query results into the final result.

The user is able to explore all facets of sensor data by traversing virtual RDF graphs of the federated SensorMasher nodes. After identifying the relevant streams, the user can connect them to fusion operators to generate the final sensor mashup with the expected output streams. The mashup will be deployed by creating data streams associated with workflow fusion operators into the DSMS component and storing metadata as triples in the metadata repository.

As the sensor mashup's configuration is represented as triples, the sensor mashup composing component is therefore an RDF graph composer. The final component required is a tool for dragging RDF nodes to a canvas and connecting them to create triples. As such, the user is able to drag sensors, mashups and fusion operators (identified by URIs) into the composer's canvas, then connect them into a workflow, and finally, save the workflow as a graph. Using this methodology, adding facts to sensors and sensor data is simply a process of creating triples.

5 Implementation

Based on the above design, we have implemented SensorMasher. The system is available at ***sensormasher.deri.org***. Similar to some related DSMSs such as TelegraphCQ [8] and STREAM [16], in this prototype, we extended SQL to support declarative continuous queries. These declarative queries will be translated into physical query plans over the underlying relational data storage. Because there are only some minor modifications compared to SQL, querying data streams with this DSMS is basically similar to querying relations. Hence, following the D2RQ [5] approach we extended Jena ARQ [26] to build the SPARQL query processor on top of the DSMS and the metadata repository. The mapping rules, which are similar to the D2RQ mapping language, are automatically generated from the configurations of the mashups.

We use two main ontologies to control the SensorMasher: the core ontology and the extended ontology. The core ontology represents core concepts and properties which are the same for every SensorMasher deployment. The extended ontology contains subclasses of the core ontology which can be customized based on specific requirements. In our prototype, the extended ontology was built by using concepts and properties from the Physical Property taxonomy in the SWEET Property ontology[8] and the SANY Sensor Taxonomy[9]. To support implicit properties inferred from ontological relationships such as subClassOf, subPropertyOf, inverseOf, etc. in answering SPARQL queries, we use the Jena in-memory reasoner to reason about these two ontologies at class level to generate query mapping rules. For example, if we query for sensors that can measure meteorological measurement, the result will contain sensors that can measure temperature, humidity, wind speed, etc.

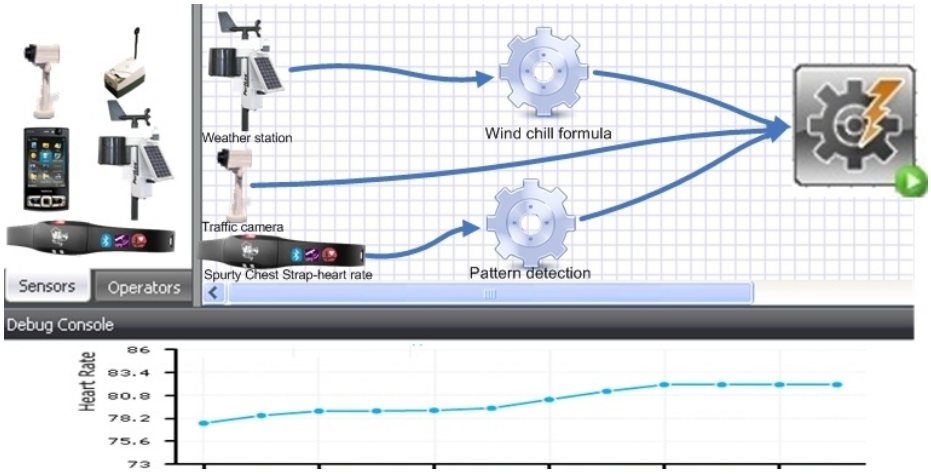

Fig. 4. Mashup Composer

The Mashup Composer and Explorer are implemented as Ajax-based web applications. Screen shots are shown in Figure 4 and Figure 5. The user is able to find and explore sensor data sources by drawing a polygon on the map to locate the area of interest. The user is also able to drill down into the result set by using facet filters on types and properties. Furthermore, from an result item, it is possible to follow its relationships to discover other items which which have

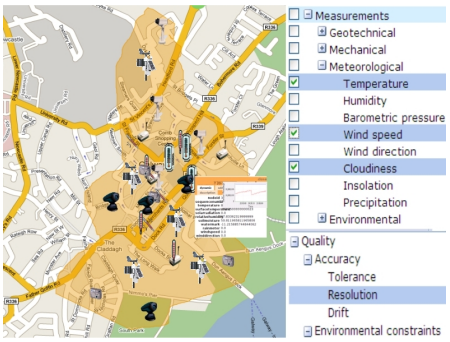

Fig. 5. Exploring sensor data sources

[8] http://sweet.jpl.nasa.gov/1.1/property.owl
[9] http://sany-ip.eu/publications/1954

semantic links to the previous one. If the user has identified a sensor data source of interest he can drag it into Mashup Composer to later wire it into his mashup.

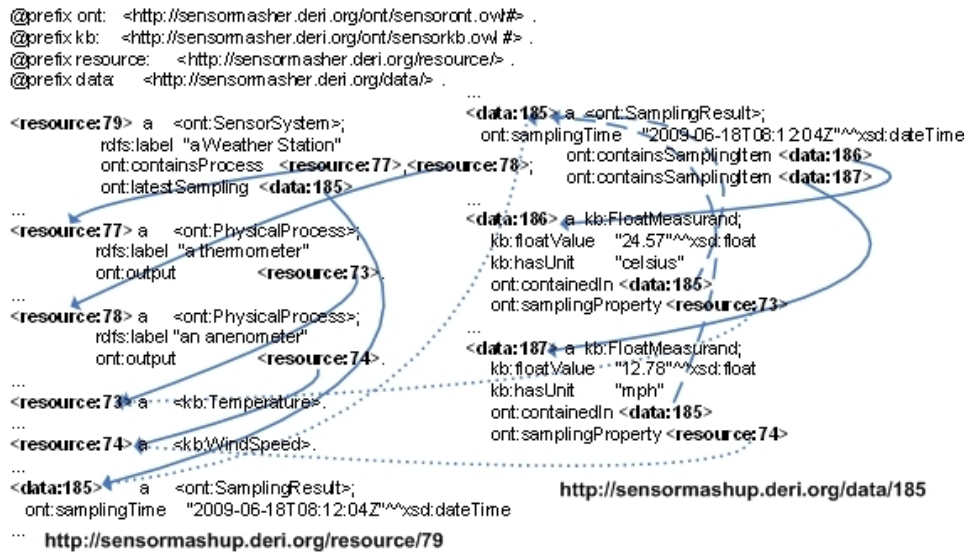

Fig. 6. Examples of sensor description and output data in RDF

After a mashup has been deployed in a SensorMasher node, it has an associated URL pattern to request different types and formats of data. For instance, URL *http://sensormasher.deri.org/resource/79* contains semantic description of a weather observation using the weather station mentioned in an example of Section 4.2 . This weather station uses a *thermometer* to measure temperate and an *anemometer* to measure wind speed. The left hand side of Figure 6 shows some RDF snippets of this weather station. In this description, along with the specification of the weather station, we see that the URL *http://sensormasher.deri.org/data/185* is URL its latest reading. The right hand side of Figure 6 shows the RDF output of this URL with some links to the aforementioned sensor description. Via this URL, the corresponding sensor reading also can be retrieved in other formats such as RDF, JSON, XML, RSS, CSV, etc. by using HTTP negotiation, so that it can be consumed by various third-party applications such as mobile applications, scripting web applications, etc.

Figure 7 shows a sample query over real-time sensor data. This query requests the "the temperature and location where the wind speed has reached 30 miles per hour around the Cliff of Moher". We also extended the 52°North[23] source code to support the OpenGIS Sensor Observation Web Service[10] with RDF triples embedded in WSDL and output data using RDFa.

[10] http://www.opengeospatial.org/standards/sos

```
@prefix ont:   <http://sensormasher.deri.org/ont/sensoront.owl#> .
@prefix kb:    <http://sensormasher.deri.org/ont/sensorkb.owl#> .
@prefix spatial:    <http://sensormasher.deri.org/ont/spatialont.owl#> .

SELECT  ?tempReading, ?lat, ?long
WHERE{
?sensorSys rdf:type ont:SensorSystem. ?sensorSys ont:locatedAt ?aFixLoc.
?aFixLoc rdf:type  spatial:FixedLocation.?aFixLoc spatial:lat ?lat. ?aFixLoc spatial:long ?long.
?aFixLoc  spatial:within ?area. ?area  spatial:geometryObjName "Cliff of Moher".
?sensorSys ont:output ?temperature. ?temperature rdf:type kb:Temperature.
?sensorSys ont:output ?winspeed. ?windspeed rdf:type kb:Windspeed.
?sensorSys ont:latestSampling ?sampling.
?tempReading ont:containedIn ?sampling.  ?tempReading ont:samplingProperty ?temperature.
?wsReading ont:containedIn ?sampling. ?wsReading ont:samplingProperty ?windspeed.
FILTER(?wsReading kbxfloatValue>30)
}
```

Fig. 7. An example of querying sensor data using SPARQL

6 Related Work

Traditionally, sensor data fusion applications require prior knowledge about the sensor systems as well as knowledge about the environment for which the system is built. However, sensor data sources as well as fusion components can be used in unpredictable ways beyond the original designer's intent. To support flexible reuse to create unpredictably creative results, specific interoperability and integration issues among sensor information system have to be overcome. Preliminary efforts to cope with these issues are the definition of standards such as standards for transducers (IEEE 1451), the Radiation Detection Standards (ANSI N42), the OGC Sensor Web Enablement [29], Extended Environments Markup Language (EEML) [25], etc. Most of these standards are over-simplified or too general or too domain-specific, so, there exist only a few systems conforming to them. The standard with the broadest impact among them seems to be the Sensor Web Enablement standard which has been implemented by the 52°North [23] project, the NASA/JPL Sensor Webs Project [27], the European Space Agency [24], etc.

Some efforts to improve accessibility to sensors are on the way as. Usually the proposed approaches try to provide simple interfaces to publish and retrieve sensor data via centralized portals. SensorBase[11] enables the publication of sensor data via HTTP POST and provides web services to query sensor data from its relational database. In a similar fashion, Pachube [12] provides a RESTful interface to stream real-time sensor data. Historical data and real-time data can be retrieved via feeds in XML (EEML) or JSON formats. Along with web services to publish and request historical and real-time sensor data, SensorMap[15] also provides an explorer for sensor data on maps. Another approach employing the HTTP protocol to make sensor data accessible is proposed by Dickerson et

[11] http://sensorbase.org/
[12] http://pachube.com/

al. [10]. It combines the advantages of the Web feed and multimedia streaming paradigms. Each sensor stream has a URL with associated parameters to query and filter historical and real-time data. The Global Sensor Networks (GSN) [1] provides a zero-programming, declarative middleware to publish sensor data directly from physical sensors connected to a PC via interfaces like USB, Bluetooth, UDP, TCP/IP, etc. or from virtual sensors. Sensor data is accessible through HTTP with querying parameters. GSN provides a wide range of supported sensors and a nice development environment with Web interfaces and visualization functionalities.

Typically, the user is interested not only in sensor readings but also in facts and statements about them. Sensor data streams have some common properties such as deployment, design, sensor type, sensor capabilities, targeted phenomena, etc. On top of that, without the associated meaning, the user cannot decide whether a sensor reading is suitable for an observation. For example, *ambient temperature* readings would be inappropriate for a weather analysis which looks for *air temperature* readings. This is a general shortcoming of the above approaches. From a user's perspective, such facts and statements about data are significant in exploring and collecting information sources to meet the user's demands of monitoring and observing the proper real-world phenomena. In a real-life setting, the most important concern are the observed object and the observable properties of interest. Only then, a user would be concerned about which sensors are able to provide the required quality and quantity data. Hence, metadata about sensors and sensor data which supports this data using fashion is essential.

In response to this demand, the Open Geospatial Consortium (OGC) proposed sensor and observation models based on XSD schema to specify interoperability interfaces and metadata encodings that enable the real-time integration of heterogeneous sensor webs into an information infrastructure. However, there is a gap between the syntactic XSD of OCG's Sensor Web Enablement (SWE) and the RDF/OWL-based metadata which is commonly used for representing domain knowledge. To bridge this gap, Shet et al. [19] proposed to use RDFa to annotate ontological concepts and properties to SWE by using XLink [30]. To encode sensor data in this form, the underlying sensor information system should have such semantic data ready. This system has to capture sufficient semantic data along with raw sensor readings and must be able to manage and process sensor stream data as well as support semantic functionalities on top. While the concepts have been proposed, no such system is in existence yet.

Some efforts make use of semantic descriptions of sensor data streams to automatically compose sensor applications and services. In 2006, Whitehouse et al. proposed Semantic Streams [21] which allows the user to pose queries based on the semantics of sensor data against the system. The approach describes sensor data stream semantics by using Prolog-based logic rules. Due to issues of scalability and decidability of that model, Booillet et al. [7] proposed to use OWL instead to represent sensor data stream as well as processing elements for composing applications from input data streams. However, these approaches also

assume that semantic descriptions of sensor data are already available and have sufficient quality for their systems.

Along these efforts of coping with sensor data interoperability and integration issues, we propose SensorMasher as the first platform for publishing sensor data as linked data. Its mashup engine enables the user to visually explore and combine sensor data resources in a homogeneous triple-based data model. As sensor data is published as linked data sources, every single data element can not only can be easily accessed but also can it be annotated with meaningful linked data. Using linked metadata as semantic description of sensor data leverages the dynamic discovery, querying, exploration, navigation and combination of sensor data sources.

7 Discussion

There are several challenging data management issues in Sensor Web environments such as data ingestion, temporal and spatial data management, data exploration, analysis and visualization, statistical modeling, data uncertainty management, data interoperability and distributed, large-scale data processing [3]. In the following we will discuss how SensorMasher tries to address some of them in the context of the triple-based data model.

The current version of SensorMasher uses SPARQL to query sensor data streams. However, SPARQL is not expressive enough to address specific data stream issues such as window slicing, sampling operators, join and correlation operators over streams, etc. Additionally, aggregating functions such as COUNT, MAX, MIN, AVERAGE, etc. which are vital in stream data processing, are not natively supported in the current version of the SPARQL working draft [18]. Hence, an extended version of SPARQL which addresses these issues and supports continuous queries over a triple-based data model is essential. At the time of this writing, there is only one paper published on extending the SPARQL grammar to process data streams [6]. The problem of the approach proposed in this paper, however, is the usage of RDF as the underlying data model which defeats the efficiency of stream query processing. While externally in the context of linking data this is desirable, internally more efficient formats should be used as we do in SensorMasher. Further research is required in order to design an efficient query language which supports triple-based queries over data streams. As mentioned in Section 5, the first prototype of SensorMasher is only able to support inferred subsumption results of SPARQL queries with small ontologies. While supporting reasoning on SPARQL queries is a difficult issue in the context of large amounts of instances and big ontologies, dealing with data instances "arriving and leaving" in a stream at high rates is even more complicated to handle.

Due to the lack of declarative query languages over triple-based data streams, there exist no approaches to map this type of query to physical query plans. In SensorMasher, we try to map SPARQL queries to query plans over the continuous queries of the underlying data stream management system (DSMS). However, additional optimization of query plans is required. Since a SensorMasher node is accessible as an autonomous SPARQL endpoint, federating SPARQL

queries over distributed SensorMasher nodes is an inevitable need. DARQ [20] and the Semantic Discovery System [28] are just a few proposals for distributed SPARQL query processing whose suitability to stream queries requires further research.

The scalability of a SensorMasher node heavily depends on the scalability of the stream query processor on top of the DSMS used by the platform. Medusa [9], and Borealis [2] are two examples of distributed DSMS designed to support large-scale data processing. Practically, employing these distributed DSMS to solve the scalability issues concerning the processing of millions of sensor data streams has not been well investigated. In this context, building the query planner to deploy and schedule query plans translated from triple-based queries over such underlying distributed DSMS is a incredibly challenging problem.

As the sensor data processing and integration in SensorMasher platforms is carried out in an autonomous and distributed fashion, data provenance is an issue which must be dealt with. Even though there have been various efforts investigating data provenance, inferring and computing origins of a derived sensor data source after it has gone through many processing phases such as summarization, aggregation, correlation, etc. are still open questions. Data provenance is especially complicated when dealing with federated queries where sensor data is transparently streamed from one autonomous node to another. By recording as much relevant sensor metadata as possible we can use semantic links to support the tracing of a particular piece of information to its origins and deciding how much to trust it.

The above discussion shows that this area requires significant research efforts in various aspects. We believe that SensorMasher can be used as an experimental platform to support practical experiments with the coming approaches and can integrate the research results produced in a straight-forward way.

8 Conclusion

This paper is the first attempt to publish sensor data as linked data to enable dynamic discovery, integration, and querying of heterogeneous sensor data sources. The ontology-driven data model and related system enable the user to easily deploy, combine and annotate sensor data sources by using a visual composer and explorer without requiring expertise in sensor programing and sensor data processing. As a linked data citizen, SensorMasher enables linking real world sensor data to The Linked Open Data dataset cloud[13].

9 Acknowledgments

This work has been supported by Science Foundation Ireland under the Lion project (SFI/02/CE1/I131)and by the European FP7 project PECES (FP7-224342-ICT-2007-2).

References

1. Aberer, K. Hauswirth, M. Salehi, A. : Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks. In : MDM'07. (2007)

[13] http://linkeddata.org/

2. Abadi, D.J. et al.: The Design of the Borealis Stream Processing Engine.CIDR' 05.
3. Balazinska, M., Deshpande, A., Franklin, M. J., and Gibbons, P. B., Gray, J,. Hansen, M., Liebhold, M., Nath, S., Szalay, A,.Tao, V. :Data Management in the Worldwide Sensor Web. In : IEEE Pervasive Computing. pp. 30–40 (2007)
4. Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. 2002. Models and issues in data stream systems. In PODS'02. (2002)
5. Bizer, C., Cyganiak, R.: D2R Server Publishing Relational Databases on the Semantic Web. In: ISWC'06.(2006)
6. Bolles, A. , Grawunder, M., Jacob, J. : Streaming SPARQL - Extending SPARQL to Process Data Streams. In: ESWC'08. (2008)
7. Bouilet, E., Feblowitz, M., Liu, Z., Ranganathan, A.,Riabov, A.,Ye, F.: A Semantics-Based Middleware for Utilizing Heterogeneous Sensor Networks. In: DCOSS'07.
8. Chandrasekaran, S., et al.,: TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In CIDR' 03.(2003).
9. Cherniack,M., Balakrishnan,H., Balazinska,M.,Carney, D. , Cetintemel, U. , Xing,Y., Zdonik, S.:Scalable Distributed Stream Processing . In CIDR'03. (2003)
10. Dickerson,R. , Lu, J., Lu, J. , Whitehouse, K. : Stream feeds - an abstraction for the world wide sensor web. In IOT'08). (2008)
11. Hall, D. L. and Llinas, J. : An introduction to multisensor data fusion. In: Proceedings of the IEEE J. (1997)
12. Hall, D. L. and Llinas, J.: Multisensor Data Fusion, Second Edition . CRC (2001)
13. Le-Phuoc, D.,Polleres, A., Hauswirth, M., Tummarello, G., Morbidoni, C. : Rapid prototyping of semantic mash-ups through semantic web pipes . In: WWW'09, pp 581–590. ACM, Madrid,Spain (2009)
14. Li, L., Taylor, K.: A Framework for Semantic Sensor Network Services. In: ISSOC' 08.(2008)
15. Luo, L., Kansal, A., Nath, S., and Zhao, F. 2008. Sharing and exploring sensor streams over geocentric interfaces. In ACM GIS '08. (2008)
16. Motwani,R., et al: Query Processing, Resource Management, and Approximation in a Data Stream Management System. In CIDR' 03.(2003).
17. Perry,M., Sheth,A., Hakimpour,F., Jain,P. : Supporting Complex Thematic, Spatial and Temporal Queries over Semantic Web Data. In: GeoS '07. (2007)
18. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. In : W3C Working Draft. (2006)
19. Sheth, A.,Henson, C.,Sahoo, S. S. : Semantic Sensor Web. IEEE Internet Computing. Vol. 12. (2008)
20. Quilitz, B., Leser, U. : Querying Distributed RDF Data Sources with SPARQL: In : ESWC 2008. (2008)
21. Whitehouse, K., Zhao, F., Liu, J.: Semantic streams: A framework for composable semantic interpretation of sensor data. In: EWSN'06, pp. 5–20. (2008)
22. Worthen, B.: 'Mashup' Sew Data Together: Software Tools Can Cut Costs, Time for linking Information Sources. In : The Wall Street J. Tuesday, July 31, 2007. p.B4
23. 52°North, http://52north.org
24. European Space Agency, http://www.esa.int/esaCP/index.html
25. Extended Environments Markup Language (EEML), http://www.eeml.org
26. Jena Semantic Web Framework, http://jena.sourceforge.net/
27. NASA/JPL Sensor Webs Project, http://sensorwebs.jpl.nasa.gov
28. Semantic discovery system, http://www.insilicodiscovery.com
29. Sensor Web Enablement WG, http://www.opengeospatial.org/projects/groups/sensorweb
30. XML Linking Language (XLink) , http://www.w3.org/TR/xlink/