A Framework for Acquiring Semantic Sensor Descriptions (Short Paper)

Luka Bradeško, Alexandra Moraru, Blaž Fortuna, Carolina Fortuna, Dunja Mladenić

Jožef Stefan Institute, Ljubljana Slovenia
{luka.bradesko, alexandra.moraru, blaz.fortuna, carolina.fortuna, dunja.mladenic}@ijs.si

Abstract. There has been great effort in developing ontologies for modeling sensor networks, describing various types of sensors and their context. However, when faced with a large scale deployment, the process of acquiring and managing semantic sensor metadata is challenging. This paper focuses on acquiring contextual metadata of sensors, such as location and surrounding environment, as opposed to technical metadata which can be derived from sensor's firmware. More specifically, the paper proposes a framework for collecting contextual metadata information with help of the mobile devices, which allows usage on the deployment site and as such lowers the cost.

Keywords. Semantic Sensor Web, Semantic Sensor Networks, Knowledge Acquisition, Mobile Applications

1 Introduction

With the rapid development and increasing number of real world applications of large scale sensor networks it became obvious that there is a need for software solutions supporting communication, sensor data retrieval and storage. In parallel to that, there is a need for an infrastructure to cover sensor descriptions, deployment and maintenance data. This paper focuses on the infrastructure supporting sensor metadata acquisition and management based on semantic technologies.

Semantic technologies have been identified as one of the key enabling technologies for sensor networks [1], contributing to understanding and managing of the sensors and measurements. One of the advantages of applying semantic technologies to sensor networks is the interoperability support, which in terms of comparison and data merging of different sensor networks, enables new solutions in solving problems.

Existing systems that semantically annotate data require it to be inserted manually via xml configuration files or wiki. Attempts to use meta-data freely inserted by users haven't proved too successful [3]. More recent approaches use custom network protocols such as the Device Identification Protocol (DIP) to automatically obtain the technical meta-data [4], or they manually annotate a small number of sensors and then, based on the similarity of measurements they label other sensors [5]. The existing applications mostly focus on the measurements and insufficient attention has been

paid to the sensor context information. The Open Geospatial Consortium leads efforts to define the Sensor Web Enablement standards i.e. SensorML and Observation-and-Measurement [1]. More recently, the need to more expressivity has been recognized, therefore the SWE standards have also been mapped into the Semantic Sensor Network (SSN) ontology by the W3C Semantic Sensor Network Incubator Group [2].

The contextual specifications depend on the concrete placement of the sensor, are more difficult to obtain and are not covered well within existing solutions. For example, an important piece of information is the geo positioning [6]. Fixed sensor platforms typically do not feature a GPS positioning module for it increases the size and cost of the platform and consumes power without providing much added value over time. Knowing the GPS coordinates enables the acquisition of the following (non-exhaustive list of) contextual information: geographical context (i.e. position on a map, plain or hills, proximity to river, etc.), details about the region such as population density, level of industrialization. All this contextual information can be automatically collected using Linked Data[1]. Besides the geographical information there is other meta-data which is best acquired at the time of the deployment: the actual time of installation, the configuration of the sensor box i.e. (Does it have external sensors, antenna? Is it waterproof?), the placement of a box, surroundings (Are there trees in the immediate vicinity, interferences on the same frequency?), etc.

2 Framework for Acquiring Semantic Sensor Descriptions

To start the KA (knowledge acquisition) process, the mobile terminal collects the sensor node ID directly from the sensor at its location site. Next, the mobile terminal sends an initialization message containing User ID, sensor node ID, time stamp and the GPS coordinates to the server. This is marked as the initialization step in Fig. 1.

The server validates the incoming data and adds it to the knowledge base. This starts an iterative process, in which the server uses predefined KA rules and the domain ontologies to generate KA forms and sends them to the mobile terminal. Next, the user fills forms via mobile terminal and submits the results back to the server. The server adds new data to the knowledge base and uses it to generate additional KA form. The KA loop then continues as described above and depicted in Fig. 1.

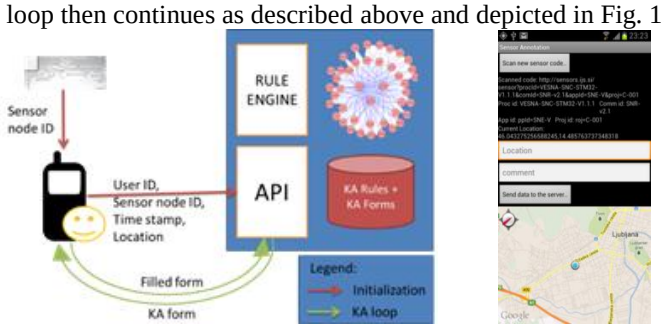

Fig. 1. Architecture of the system (left) and mobile application (right)

[1] http://linkeddata.org/

The proposed framework is generic as its components can be implemented using a variety of existing tools and technologies. For instance, the acquisition of the sensor node ID can be done using barcodes, QR codes, RFID, NFC, etc. With respect to the vocabulary, the Semantic Sensor Network (SSN) ontology, geospatial ontologies such as Basic GeoWGS84 Vocabulary, GeoNames, time representation with W3C time ontology and observed properties using SWEET ontologies can be used. Further, application specific extensions can be added or generated when using the system.

3 Knowledge Acquisition

For the knowledge acquisition we took similar approach as is used in crowdsourcing the ontology learning process. We present to the user a variety of forms/questions, which are dynamically generated based on previous answers and based on the sensor instance. In order to define when and which forms to show, we introduced special vocabulary [7] in the ontology, to mark which knowledge to elicit. When the particular property is marked for KA, then the appropriate form gets generated.

We had to create a set of rules which trigger knowledge elicitation. For example if we want to get the bounding for property <*ssn:observes*> on all instances of class <*ssn:Sensor*>, we define the rule as follows:

[IF <?concept><rdf:type> <ssn:Sensor> THEN
<?concept> <ijs:generateForms> <ssn:observes>] (1)

This means that the system will ask for all the sensors what physical quantity is measured by it. These rules can be stacked to form a follow up conversation. When the first form is filled, the next one responds based on the previous answer, e.g.:

[IF <?concept> <rdf : type> <ssn:Sensor> AND
<?concept> <ssn: observes> <_:observationProperty1> AND
<_:observationProperty1> <rdf: value> <"temperature"> THEN
<?concept> <ijs:generateForms> <ijs:hasTemperatureSensorType>] (2)

This rule is triggered only when the assertion based on the previous answer states that the particular sensor measures temperature.

Theoretically it is possible that the user is presented with the RDF property and part of the ontology which he/she has to fill with the missing data. But this would be tedious work and requires the user of the system to be skilled in ontology engineering. To make the knowledge acquisition as easy and errorless as possible, the user is presented with the forms where he can enter knowledge without the RDF burden. These forms can be anything from HTML templates, to natural language questions and can be stored inside the ontology using special vocabulary or in an external database.

These forms are hooked up on the <*rdf:property*> resources and the knowledge about domain and the range is used to propose the predefined values which the user can enter. The example of the form which was issued by rule (1) can be seen in Fig. .

```
<form subject = "?$1" object ="?$2" predicate = "ssn:observes">
        "This sensor can measure ?$2"
</form>
```

Fig. 2. Example knowledge acquisition form.

Before presented to the user, the form is enriched with the constraints and possible answers, inferred from the <*rdf:range*> part of the ontology. In this example, the options would be "temperature", "mass", "current", "voltage", etc. If the user enters something new, the system informs him that he is creating a new concept, which is then added to the ontology, so everything stays consistent.

4 Implementation and Case Studies

As an example we look at an application around a plant care scenario, which shows the variety of data that can be acquired. There are numerous factors influencing the development of house plants, such as temperature, humidity, soil moisture, etc. Most of the information about the conditions required by house plants is available on the web; however, connecting it to the actual setting of the environment can be facilitated using a KA system.

Let us consider a scenario where three sensors are available for measuring the living conditions of our plant: a temperature sensor, a humidity sensor and soil moisture sensor. Further, we have defined a set of rules using concepts from SSN and SWEET ontologies and few extensions for sensor types and properties observed by sensors.

A simple example of a rule in this scenario would be "*If a sensor is attached to the flower pot, then ask what plant is in the pot?*" The rule can be represented with the <*sweet:hasOrganism*> predicate from SWEET ontology.

```
[IF <?concept> <rdf:type> <ssn:Sensor> AND
<?pot> <ssn:attachedSystem> <?concept> AND
<?pot> <rdf:type> <ijs:PlantPot> THEN
<?pot> <ijs:generateForms> <sweet:hasOrganism>]                    (3)
```

A more elaborate example is the acquisition of lighting conditions. Possible answers are fixed to the following list: direct, indirect sunlight, shadow, artificial light. Depending on the answer, further information can be inferred or asked. If the answer is one of the natural lights, the time interval can be calculated based on the geographical location and date. If the answer is not natural light, then a question about the exposure interval is issued. The above example translates to the following two KA rules:

```
[IF <?plant> <rdf:type> <sweet:Organism> THEN
<?plant> <ijs:generateForms> <sweet:assimilate>]

[IF <?plant> <sweet:assimilate> <?light> AND
<?light> <rdf:type> <ijs:ArtificialLight> THEN
<?light> <ijs:generateForms> <sweet:hasDuration>]
```

Other relevant meta-data can refer to the growing stage of the plant (germination, growth), last fertilization, size of the plant and pot, etc. The purpose for collecting all this metadata is to provide external applications the information needed to automatically detect when watering or fertilization is needed, etc.

The framework was also tested in an ongoing real world sensor deployment. In the CREW[2] project we installed 50 boxes with ISM and TV band energy detection sensors on public infrastructure in the town of Logatec. We manually prepared a list of technical configurations for each of the boxes. These configurations include: processing module ID, communication module ID, application module ID and project specific label. Next, QR codes with box-specific URIs were attached to their corresponding sensor boxes. In the field, the technician installing the sensor box read the QR code using his mobile terminal, and used the KA application (shown in Fig. 1) to geo tag the sensor and provide additional meta-data such as installation timestamp, infrastructure ID, infrastructure type (light pole, wall, and roof), particularities of the vicinity (tress, obstacles) and line of sight to other sensor boxes.

5 Conclusions

Comparison with the other solutions shows that our framework lowers the price of the meta-data acquisition. It allows users to collect data on the fly, using mobile devices and provides a controlled acquisition environment. The collected data is automatically validated and linked with LOD datasets. The proposed framework was compared feature wise with other similar systems, and the results are presented in Table 1. It was further validated in two case studies, demonstrating its usability for gathering technical data from the common users (plant caring) and in the real world deployment of sensors.

GSN[3] system uses XML files to manually label and describe each sensor with arbitrary data. The configuration process is easy for non-experts, but the disadvantage is that its terminology will differ across deployments, as it is hard to align different keywords used across deployments. Recently, work on using SSN, DOLCE and others as standardized vocabulary has been done, however, the focus is search rather than knowledge acquisition, therefore the process is not as automated and controlled as in the proposed framework.

Cosm[4] (the former Pachube) uses a predefined schema, therefore non-experts users can insert information such as title of the sensor setup, feed, id, location, etc. There's no standard vocabulary and no extension by the user seems possible. The scope of the platform is federating data from sensor deployment and eventually monetizing on it, rather than building an interoperable data management system.

In the SPIRFIRE project [5], work on automatically labeling streams of measurements is ongoing. Standard vocabulary is used, but the approach is not geared towards meta-data acquisition.

---

[2] http://www.crew-project.eu/

[3] http://sourceforge.net/apps/trac/gsn/

[4] https://cosm.com/docs/v2/

Linked Stream Middleware-LSM [8] is a platform where users register their sensors and annotate the data stream. It uses standard ontologies, with the possibility of importing new ones. The drawback of this system is that it can be only used by expert ontologists and the annotation has to be done after the sensor is already deployed.

As part of future work, we plan to use the system in a large sensor deployment scenario, as described in Section 4. We also plan to further abstract the system by making the definition of KA rules and forms simpler and more automated.

Table 1. Feature wise comparison of existing sensor meta-data systems

Solution	Schema	Vocabulary	Ontology	Non-expert use
GSN	No schema	Custom	Fixed	Yes
COSM	Predefined	Custom	Fixed	Yes
SPITFIRE	Flexible	Standard	Fixed	No
LSM	Flexible	Standard	Extendible	No
Our KA system	Flexible	Standard	Extendible	Yes

Acknowledgments. This work was partially supported by the Slovenian Research Agency and the ICT Programme of the EC under, ENVISION (ICT-2009-249120) and PlanetData (ICT-NoE-257641). Thanks to our colleagues in SensorLab.

References

1. A. Sheth, C. Henson, S. Sahoo: Semantic Sensor Web. IEEE Internet Computing, 2009.
2. M. Compton, et al.: The SSN Ontology of the W3C Semantic Sensor Network Incubator Group. Journal of Web Semantics, Elsevier, 2012.
3. A. Moraru, C. Fortuna, D. Mladenić: Using Semantic Annotation for Knowledge Extraction from Geographically Distributed and Heterogeneous Sensor Data. SensorKDD (colocated with KDD), July 2010.
4. A. Moraru, M. Vucnik, M. Porcius, C. Fortuna, M. Mohorcic, D. Mladenic: Exposing Real World Information for the Web of Things. IIWeb (co-located with WWW), March 2011
5. M. Hauswirth, I. Chatzigiannakis: Provisioning Semantic IoT Services, Project: SPITFIRE.FIRE Hands-on FIA, Aalborg, Denmark.
6. C. Fortuna, B. Fortuna, K. Kenda, M. Vucnik, A. Moraru, D. Mladenic: Towards Building a Global Oracle: a Physical Mashup Using Artificial Intelligence Technology. Third International Workshop on the Web of Things, June 2012.
7. M. Witbrock, C. Matuszek, A. Brusseau, R.C Kahlert, C.B. Fraser, D.B. Lenat: Knowledge Begets Knowledge: Steps towards Assisted Knowledge Acquisition in Cyc. AAAI Spring Symposium on Knowledge Collection from Volunteer Contributors (KCVC), pp. 99-105. Stanford, California, March 2005.
8. D. Le Phuoc, H. N. Mau Quoc, J. X. Parreira, M. Hauswirth: The Linked Sensor Middleware - Connecting the real world and the Semantic Web. In Semantic Web Challenge 2011, ISWC 2011, http://challenge.semanticweb.org